ElectroCraft

EtherCAT CoE Programming

For PRO Series Programmable Servo Drives

ElectroCraft Document Number
A11358 Rev 1

User Manual

ELECTROCRAFT

CAN application protocol over EtherCAT® (CoE) Programming

User Manual

ElectroCraft Document Number
A11358 Revision 1

ElectroCraft
1 Progress Drive
Dover, NH 03820

www.electrocraft.com

Read This First

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

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

About This Manual

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

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

- ❑ **Step 1 Hardware installation**
- ❑ **Step 2 Drive commissioning** using initially the ElectroCraft **PRO Config** or **MotionPRO Developer** software platforms and later via an EtherCAT® master
- ❑ **Step 3 Motion programming** using one of the options:
  A. An **EtherCAT® master**
  B. The drive **built-in motion controller** executing a ElectroCraft's Motion Programming Language (**MPL**) program developed using ElectroCraft **MotionPRO Developer** software
  C. A **distributed control** approach which combines the above options, like for example a master calling motion functions programmed on the drives in MPL

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

Scope of This Manual

This manual applies to all ElectroCraft programmable drives having an EtherCAT® communication interface.

Notational Conventions

This document uses the following conventions:

- ❑ **AL –** Application Layer
- ❑ **CoE -** CAN application protocol over EtherCAT®
- ❑ **ControlWord.5** – bit 5 of ControlWord data
- ❑ **ESM** – EtherCAT® State Machine
- ❑ **IP -** interpolated
- ❑ **IU** – drive/motor internal units
- ❑ **MPL** – ElectroCraft's Motion Programming Language
- ❑ **CSP** – Cyclic Synchronous Position
- ❑ **CSV** – Cyclic Synchronous Velocity
- ❑ **CST** – Cyclic Synchronous Torque
- ❑ **X1 –** Subindex 1 of object $60C1_h$ - Interpolation data record
- ❑ **X2 –** Subindex 2 of object $60C1_h$ - Interpolation data record

Trademarks

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

Related Documentation

Technical Reference Manual of each programmable drive with EtherCAT® interface – provides information needed for hardware installation: technical data, connectors pin assignment, wiring diagrams plus a detailed presentation of drive setup procedure.

Help of the PRO Config software – describes how to use **PRO Config** to quickly setup any ElectroCraft drive for your application using only 2 dialogues. The output of PRO Config 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 PRO Config it is also possible to retrieve the complete setup information from a previously programmed drive.

Motion Programming using ElectroCraft MotionPRO Suite (Document* No. A11229*) – describes how to use the MotionPRO Suite to create motion programs using the ElectroCraft Motion PROgramming Language (MPL). The MotionPRO Suite platform includes **PROconfig** 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 MPL instructions. *With MotionPRO Suite you can fully benefit from a key advantage of ElectroCraft drives – their capability to execute complex moves without requiring an external motion controller, thanks to their built-in motion controller.* Motion PRO Suite is available as part of a PRO Series Drive Evaluation Kit. Please contact ElectroCraft or your local ElectroCraft sales representative for more information on obtaining MotionPRO Suite or an evaluation kit.

If you Need Assistance …

If you want to …	Contact ElectroCraft at …
Visit ElectroCraft online	World Wide Web: www.electrocraft.com
Receive general information or assistance (see Note)	World Wide Web: www.electrocraft.com Email: drivesupport@electrocraft.com
Ask questions about product operation or report suspected problems (see Note)	Tel : +1 734.662-7771 Email: drivesupport@electrocraft.com
Make suggestions about, or report errors in documentation (see Note)	Mail: ElectroCraft 1 Progress Drive Dover, NH 03820 USA

This page is empty.

Contents

Read This First .... **III**
About This Manual III
Scope of This Manual III
Notational Conventions III
Trademarks IV
Related Documentation IV
Contents .... **VII**
1 Getting Started **14**
1.1 Setting up the drive using PRO Config or MotionPRO Developer 14
1.1.1 What are PRO Config and MotionPRO Developer? 14
1.1.2 Installing PRO Config or MotionPRO Developer 15
1.1.3 Establishing serial communication with the drive 15
1.1.4 Choosing the drive, motor and feedback configuration 16
1.1.5 Commissioning the drive 19
1.1.6 Downloading setup data to drive/motor 20
1.1.7 Saving setup data in a file 20
1.1.8 Creating a .sw file with the setup data 21
1.1.9 Checking and updating setup data via .sw files with an EtherCAT® master 21
1.1.10 Testing and monitoring the drive behavior 21
1.2 Setting up EtherCAT® communication. Example with TwinCAT2 21
1.2.1 Adding the XML file 22
1.2.2 Understanding EtherCAT® addressing modes supported 22
1.2.3 Detecting the drive 22
1.2.4 Configuring ElectroCraft EtherCAT drives for NC PTP compatibility 25
1.2.5 Running the NC-PTP interface 29
1.2.6 Checking and updating the XML file stored in the drive 32
1.2.7 Setting the free run communication cycle 35
1.3 Controlling the drive using CoE commands. Examples 37
1.3.1 Starting a position profile with TwinCAT System Manager 37
1.3.2 Starting a cyclic synchronous position mode 41
1.3.3 Mapping objects to TxPDOs and RxPDOs in TwinCAT System Manager 42
1.3.4 Running an Interpolated Position Time motion in TwinCAT 44
2 CAN application protocol over EtherCAT® **48**
2.1 EtherCAT® Architecture 48
2.2 Accessing EtherCAT® devices 49
2.2.1 CoE elements 49
2.2.2 Object dictionary 50
2.2.3 Object access using index and sub-index 50
2.2.4 Service Data Objects (SDO) 50
2.2.5 Process Data Objects (PDO) 52
2.3 Objects that define SDOs and PDOs 52
2.3.1 Object 1600h: Receive PDO1 Mapping Parameters 52
2.3.2 Object 1602h: Receive PDO2 Mapping Parameters 54

2.3.3 Object 1602h: Receive PDO3 Mapping Parameters 54
2.3.4 Object 1603h: Receive PDO4 Mapping Parameters 55
2.3.5 Object 1A00h: Transmit PDO1 Mapping Parameters 56
2.3.6 Object 1A01h: Transmit PDO2 Mapping Parameters 56
2.3.7 Object 1A02h: Transmit PDO3 Mapping Parameters 57
2.3.8 Object 1A03h: Transmit PDO4 Mapping Parameters 58
2.3.9 Object 1C00h: Sync Manager Communication type 59
2.3.10 Object 1C12h: Sync Manager Channel 2 (Process Data Output) 60
2.3.11 Object 1C13h: Sync Manager Channel 3 (Process Data Input) 61
2.3.12 Object 207Dh: Dummy 62
2.4 PDOs mapping example 63
3 EtherCAT® State Machine (ESM) 65
3.1 Overview 65
3.2 Device control 65
3.3 EtherCAT® State Machine and CANopen State Machine 67
3.4 Emergency messages 68
3.5 EtherCAT® Synchronization 70
3.5.1 Overview 70
3.5.2 Synchronization signals 70
3.5.3 Object 2089h: Synchronization test config 71
4 Drive control and status 73
4.1 Overview 73
4.2 Drive control and status objects 77
4.2.1 Object 6040h: Control Word 77
4.2.2 Object 6041h: Status Word 81
4.2.3 Object 1002h: Manufacturer Status Register 83
4.2.4 Object 6060h: Modes of Operation 84
4.2.5 Object 6061h: Modes of Operation Display 85
4.3 Error monitoring 86
4.3.1 Object 1001h: Error Register 86
4.3.2 Object 2000h: Motion Error Register 87
4.3.3 Object 2001h: Motion Error Register Mask 89
4.3.4 Object 2002h: Detailed Error Register 89
4.3.5 Object 605Ah: Quick stop option code 90
4.3.6 Object 605Bh: Shutdown option code 91
4.3.7 Object 605Ch: Disable operation option code 92
4.3.8 Object 605Dh: Halt option code 93
4.3.9 Object 605Eh: Fault reaction option code 93
4.3.10 Object 6007h: Abort connection option code 94
4.4 Digital I/O control and status objects 95
4.4.1 Object 60FDh: Digital inputs 95
4.4.2 Object 60FEh: Digital outputs 96
4.4.3 Object 2045h: Digital outputs status 99
4.4.4 Object 2102h: Brake Status 99
4.4.5 Object 2046h: Analogue input: Reference 100
4.4.6 Object 2047h: Analogue input: Feedback 100
4.4.7 Object 2055h: DC-link voltage 101
4.4.8 Object 2058h: Drive Temperature 101
4.4.9 Object 208Bh: Sin AD signal from Sin/Cos encoder 102

4.4.10 Object 208Ch: Cos AD signal from Sin/Cos encoder 102
4.4.11 Object 208Eh: Auxiliary Settings Register 103
4.4.12 Object 2108h: Filter variable 16bit 104

4.5 Protections Setting Objects 105
4.5.1 Object 607Dh: Software position limit 105
4.5.2 Object 2050h: Over-current protection level 106
4.5.3 Object 2051h: Over-current time out 107
4.5.4 Object 2052h: Motor nominal current 107
4.5.5 Object 207Fh: Current limit 108
4.5.6 Object 2053h: I2t protection integrator limit 108
4.5.7 Object 2054h: I2t protection scaling factor 110

4.6 Step Loss Detection for Stepper Open Loop configuration 111
4.6.1 Object 2083h: Encoder Resolution 111
4.6.2 Object 2084h: Stepper Resolution 111
4.6.3 Enabling step loss detection protection 113
4.6.4 Step loss protection setup 114
4.6.5 Recovering from step loss detection fault 114
4.6.6 Remarks about Factor Group settings when using step the loss detection 114

4.7 Drive info objects 115
4.7.1 Object 1000h: Device Type 115
4.7.2 Object 6502h: Supported drive modes 115
4.7.3 Object 1008h: Manufacturer Device Name 116
4.7.4 Object 1009h: Manufacturer Hardware Version 117
4.7.5 Object 100Ah: Manufacturer Software Version 117
4.7.6 Object 2060h: Software version of a MPL application 117
4.7.7 Object 1018h: Identity Object 118

4.8 Miscellaneous Objects 120
4.8.1 Object 2025h: Stepper current in open-loop operation 120
4.8.2 Object 2026h: Stand-by current for stepper in open-loop operation 120
4.8.3 Object 2027h: Timeout for stepper stand-by current 121
4.8.4 Object 2100h: Number of Steps per revolution 121
4.8.5 Object 2101h: Number of microSteps per Step 122
4.8.6 Object 2103h: Number of encoder counts per revolution 122
4.8.7 Object 2075h: Position triggers 123
4.8.8 Object 2076h: Save current configuration 123
4.8.9 Object 208Ah: Save setup status 124
4.8.10 Object 2080h: Reset drive 124
4.8.11 Object 2082h: Sync on fast loop 125
4.8.12 Object 2085h: Position triggered outputs 126

5 Factor group 128

5.1 Factor group objects 128
5.1.1 Object 607Eh: Polarity 128
5.1.2 Object 6089h: Position notation index 129
5.1.3 Object 608Ah: Position dimension index 130
5.1.4 Object 608Bh: Velocity notation index 130
5.1.5 Object 608Ch: Velocity dimension index 131
5.1.6 Object 608Dh: Acceleration notation index 131
5.1.7 Object 608Eh: Acceleration dimension index 132
5.1.8 Object 206Fh: Time notation index 132
5.1.9 Object 2070h: Time dimension index 133

5.1.10 Object 6093h: Position factor 134
5.1.11 Object 6094h: Velocity encoder factor....................134
5.1.12 Object 6097h: Acceleration factor 135
5.1.13 Object 2071h: Time factor 136

6 Homing Mode 138

6.1 Overview 138

6.2 Homing methods....................139
6.2.1 Method 1: Homing on the Negative Limit Switch. 139
6.2.2 Method 2: Homing on the Positive Limit Switch. 140
6.2.3 Methods 3 and 4: Homing on the Positive Home Switch and Index Pulse. 140
6.2.4 Methods 5 and 6: Homing on the Negative Home Switch and Index Pulse. 141
6.2.5 Methods 7 to14: Homing on the Negative Home Switch and Index Pulse. 141
6.2.6 Methods 17 to 30: Homing without an Index Pulse 144
6.2.7 Methods 33 and 34: Homing on the Index Pulse....................145
6.2.8 Method 35: Homing on the Current Position 145
6.2.9 Method -1: Homing on the Negative Mechanical Limit and Index Pulse 146
6.2.10 Method -2: Homing on the Positive Mechanical Limit and Index Pulse....................147
6.2.11 Method -3: Homing on the Negative Mechanical Limit without an Index Pulse. 148
6.2.12 Method -4: Homing on the Positive Mechanical Limit without an Index Pulse. 150

6.3 Homing Mode Objects 151
6.3.1 Controlword in homing mode....................151
6.3.2 Statusword in homing mode 152
6.3.3 Object 607Ch: Home offset 152
6.3.4 Object 6098h: Homing method....................153
6.3.5 Object 6099h: Homing speeds 153
6.3.6 Object 609Ah: Homing acceleration 154
6.3.7 Object 207Bh: Homing current threshold 155
6.3.8 Object 207Ch: Homing current threshold time 155

6.4 Homing example....................157

7 Position Profile Mode 159

7.1 Overview 159
7.1.1 Discrete motion profile (*change set immediately* = 0) 159
7.1.2 Continuous motion profile (*change set immediately* = 1) 160
7.1.3 Controlword in profile position mode 160
7.1.4 Statusword in profile position mode....................161

7.2 Position Profile Mode Objects....................162
7.2.1 Object 607Ah: Target position....................162
7.2.2 Object 6081h: Profile velocity 162
7.2.3 Object 6083h: Profile acceleration....................163
7.2.4 Object 6085h: Quick stop deceleration 163
7.2.5 Object 2023h: Jerk time 164
7.2.6 Object 6086h: Motion profile type 164
7.2.7 Object 6062h: Position demand value....................165
7.2.8 Object 6063h: Position actual internal value 165
7.2.9 Object 6064h: Position actual value 165
7.2.10 Object 6065h: Following error window....................166
7.2.11 Object 6066h: Following error time out....................166
7.2.12 Object 6067h: Position window....................167
7.2.13 Object 6068h: Position window time....................167
7.2.14 Object 60F4h: Following error actual value 168

7.2.15 Object 60FCh: Position demand internal value 168
7.2.16 Object 2022h: Control effort 169
7.2.17 Object 2081h: Set/Change the actual motor position 169
7.2.18 Object 2088h: Actual internal position from sensor on motor 170
7.2.19 Object 208Dh: Auxiliary encoder position 170
7.3 Position Profile Example 171
8 Interpolated Position Mode 175
8.1 Overview 175
8.1.1 Internal States 175
8.1.2 Controlword in interpolated position mode 176
8.1.3 Statusword in interpolated position mode 176
8.2 Interpolated Position Objects 177
8.2.1 Object 60C0h: Interpolation sub mode select 177
8.2.2 Object 60C1h: Interpolation data record 178
8.2.3 Object 60C2h: Interpolation time period 180
8.2.4 Object 2072h: Interpolated position mode status 181
8.2.5 Object 2073h: Interpolated position buffer length 183
8.2.6 Object 2074h: Interpolated position buffer configuration 183
8.2.7 Object 2079h: Interpolated position initial position 184
8.2.8 Object 207Ah: Interpolated position 1st order time 185
8.2.9 Loading the interpolated points 185
8.3 PT absolute movement example 186
8.4 PVT absolute movement example 188
8.5 PVT relative movement example 192
9 Velocity Profile Mode 196
9.1 Overview 196
9.1.1 Controlword in profile velocity mode 196
9.1.2 Statusword in profile velocity mode 196
9.2 Velocity Mode Objects 197
9.2.1 Object 6069h: Velocity sensor actual value 197
9.2.2 Object 606Bh: Velocity demand value 198
9.2.3 Object 606Ch: Velocity actual value 198
9.2.4 Object 606Fh: Velocity threshold 199
9.2.5 Object 60FFh: Target velocity 199
9.2.6 Object 60F8h: Max slippage 200
9.2.7 Object 2005h: Max slippage time out 200
9.2.8 Object 2087h: Actual internal velocity from sensor on motor 202
9.3 Velocity profile example 202
10 Electronic Gearing Position (EGEAR) Mode 204
10.1 Overview 204
10.1.1 Controlword in electronic gearing position mode (slave axis) 205
10.1.2 Statusword in electronic gearing position mode 205
10.2 Gearing Position Mode Objects 206
10.2.1 Object 201Eh: Master position 206
10.2.2 Object 2012h: Master resolution 206
10.2.3 Object 2013h: EGEAR multiplication factor 207
10.2.4 Object 2017h: Master actual position 208
10.2.5 Object 2018h: Master actual speed 208

10.2.6 Object 201Dh: External Reference Type 209
10.3 Electronic gearing through online communication example 210
11 Electronic Camming Position (ECAM) Mode 212
11.1 Overview 212
11.1.1 Controlword in electronic camming position mode 213
11.1.2 Statusword in electronic camming position mode 214
11.2 Electronic Camming Position Mode Objects 214
11.2.1 Object 2019h: CAM table load address 214
11.2.2 Object 201Ah: CAM table run address 215
11.2.3 Object 201Bh: CAM offset 215
11.2.4 Object 206Bh: CAM: input scaling factor 216
11.2.5 Object 206Ch: CAM: output scaling factor 216
11.2.6 Building a CAM profile and saving it as an .sw file example 217
11.3 Electronic gearing through online communication example 224
12 Cyclic synchronous position mode 226
12.1 Overview 226
12.1.1 Controlword in synchronous position mode 226
12.1.2 Statusword in synchronous position mode 227
12.1.3 Object 2086h: Limit speed/acceleration for CSP/CSV 227
12.2 Cyclic synchronous position profile example 228
13 Cyclic synchronous velocity mode 229
13.1 Overview 229
13.1.1 Controlword in synchronous velocity mode 229
13.1.2 Statusword in synchronous velocity mode 230
13.2 Cyclic synchronous velocity profile example 230
14 Cyclic synchronous torque mode 232
14.1 Overview 232
14.1.1 Controlword in cyclic synchronous torque mode 232
14.1.2 Statusword in cyclic synchronous torque mode 233
14.1.3 Object 6071h: Target torque 233
14.1.4 Object 6077h: Torque actual value 234
14.2 Cyclic synchronous velocity profile example 234
15 Touch probe functionality 236
15.1 Overview 236
15.2 Touch probe objects 236
15.2.1 Object 60B8h: Touch probe function 236
15.2.2 Object 60B9h: Touch probe status 238
15.2.3 Object 60BAh: Touch probe 1 positive edge 240
15.2.4 Object 60BBh: Touch probe 1 negative edge 240
15.2.5 Object 60BCh: Touch probe 2 positive edge 241
15.2.6 Object 60BDh: Touch probe 2 negative edge 241
15.2.7 Object 2104h: Auxiliary encoder function 242
15.2.8 Object 2105h: Auxiliary encoder status 243
15.2.9 Object 2106h: Auxiliary encoder captured position positive edge 244
15.2.10 Object 2107h: Auxiliary encoder captured position negative edge 244
16 Data Exchange between EtherCAT® master and drives 245
16.1 Checking Setup Data Consistency 245

16.2 Image Files Format and Creation 245
16.3 Data Exchange Objects 246
16.3.1 Object 2064h: Read/Write Configuration Register 246
16.3.2 Object 2065h: Write 16/32 bits data at address set in Read/Write Configuration Register 247
16.3.3 Object 2066h: Read 16/32 bits data from address set in Read/Write Configuration Register 248
16.3.4 Object 2067h: Write data at specified address 249
16.3.5 Object 2069h: Checksum configuration register 249
16.3.6 Object 206Ah: Checksum read register 250
16.4 Downloading an image file (.sw) to the drive via CoE commands 251
16.5 Checking and loading the drive setup via .sw file and CoE commands example. 253
17 Advanced features 255
17.1 Using MotionPRO Developer 255
17.1.1 Starting a new project 255
17.1.2 Choosing the drive, motor and feedback configuration 256
17.1.3 Downloading setup data to drive/motor 258
17.2 Using MPL Functions to Split Motion between Master and Drives 258
17.2.1 Build the MPL program within MotionPRO Developer 258
17.2.2 Build MPL functions within MotionPRO Developer 259
17.2.3 Downloading MPL program and functions within MotionPRO Developer 260
17.3 MPL Function Objects 261
17.3.1 Executing MPL programs 261
17.3.2 Object 2077h: Execute MPL program 261
17.4 Loading Automatically Cam Tables Defined in MotionPRO Developer 262
17.4.1 CAM table structure 262
17.5 Customizing the Homing Procedures 263
17.6 Customizing the Drive Reaction to Fault Conditions 264
17.7 Saving the program, functions and setup to an image file (.sw) 264
Appendix A Object Dictionary by Index 265

1 Getting Started

1.1 Setting up the drive using PRO Config or MotionPRO Developer

1.1.1 What are PRO Config and MotionPRO Developer?

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

The output of PRO Config 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 ElectroCraft drive. At power-on, the drive is initialized with the *setup data* read from its EEPROM. PRO Config may also be used to retrieve the *setup data* previously stored in a drive EEPROM.

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

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

MotionPRO Developer includes **PRO Config** 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 ElectroCraft's Motion Programming Language (MPL). It automatically generates all the MPL instructions, hence you don't need to learn or write any MPL code. Via MPL 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 MPL 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 MotionPRO Developer 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 *MPL motion program*.

Using MPL, 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 MPL to execute complex tasks, and inform the master when these tasks have been completed.

Important: *You need* ***MotionPRO Developer full version****, only if you use MPL programming.*

1.1.2 Installing PRO Config or MotionPRO Developer

PRO Config and **MotionPRO Developer demo version** can be downloaded ***free of charge*** from ElectroCraft 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.

MotionPRO Developer demo version includes a fully functional version of **PRO Config**, hence you don't need to install both of them.

You can install the MotionPRO Developer full version in 2 ways:

a) Using the CD provided by ElectroCraft. In this case, after installation, use the ***Update via Internet*** tool to check for the latest updates;

b) Transforming MotionPRO Developer demo into a full version, by introducing in the application menu command **Help | Registration Info** the serial number provided by ElectroCraft.

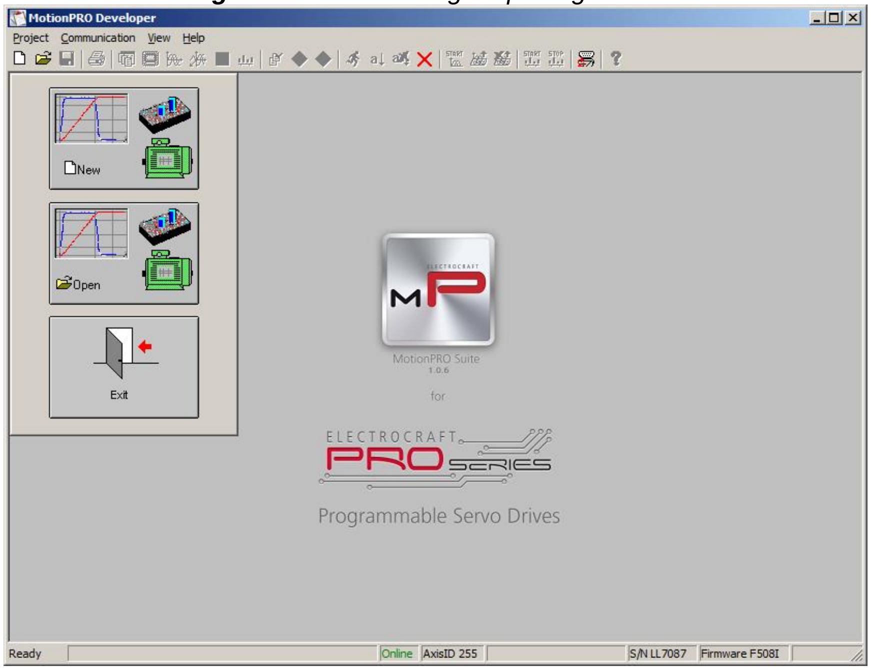
The 2nd option is especially convenient if the MotionPRO Developer demo version is already installed.

Remark*: The next paragraphs present only the drive commissioning with PRO Config. Par. 17.1.1 shows how to perform the same steps with MotionPRO Developer.*

1.1.3 Establishing serial communication with the drive

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

Figure 1.1 *PRO Config - Opening window*



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

Remark: *When first started, PRO Config 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.*

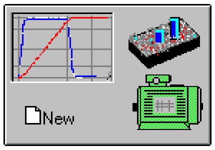
If the communication is established, PRO Config displays in the status bar (the bottom line) the text "**Online**" plus the axis ID of your drive/motor and its firmware version. Otherwise the text displayed is "**Offline**" and a communication error message tells you the error type. In this case, use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

- **Channel Type**: RS232
- **CAN Protocol:** can be any selection. The CANbus is not used
- **Port:** Select the COM port where you have connected the drive
- **Baud rate:** can be any value. The baud rate is automatically detected. For best performance, we recommend to use the highest value: 115200.
  Remark: *Once the communication is established, you can reopen the* **Communication | Setup** dialogue and change the baud rate
- **Axis ID of drive/motor connected to PC is:** autodetected or 255

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

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

1.1.4 Choosing the drive, motor and feedback configuration

Press **New** button  and select your drive category: PRO Series Drives (all drives from the new PRO Series line), Plug In Drives (all plug-in drives, except PRO Series line), Open Frame Drives, (all open-frame drives except PRO Series line), Closed Frame Drives (all close-frame drives except PRO Series line), etc. If you don't know your drive category, you can find it on ElectroCraft 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).

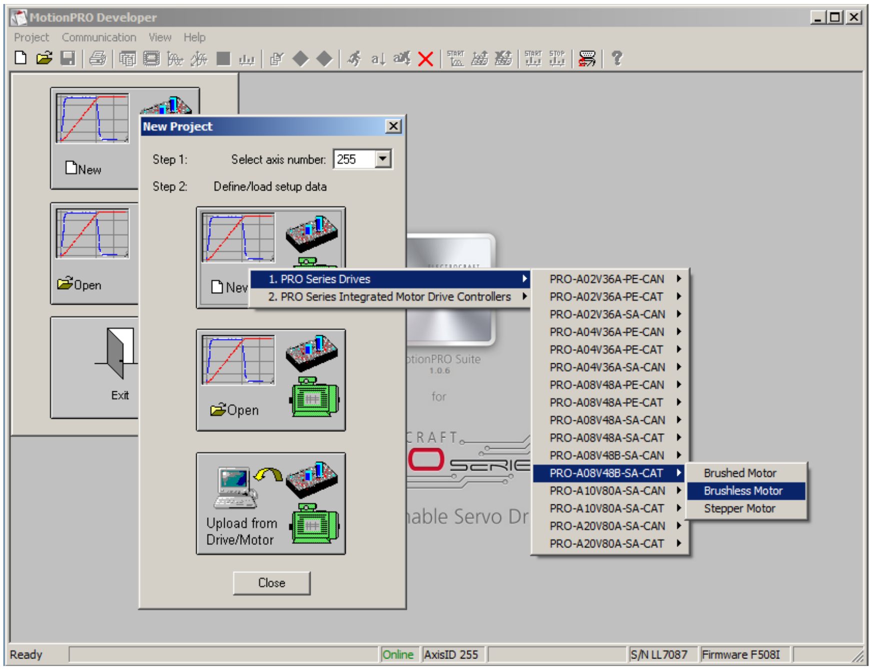
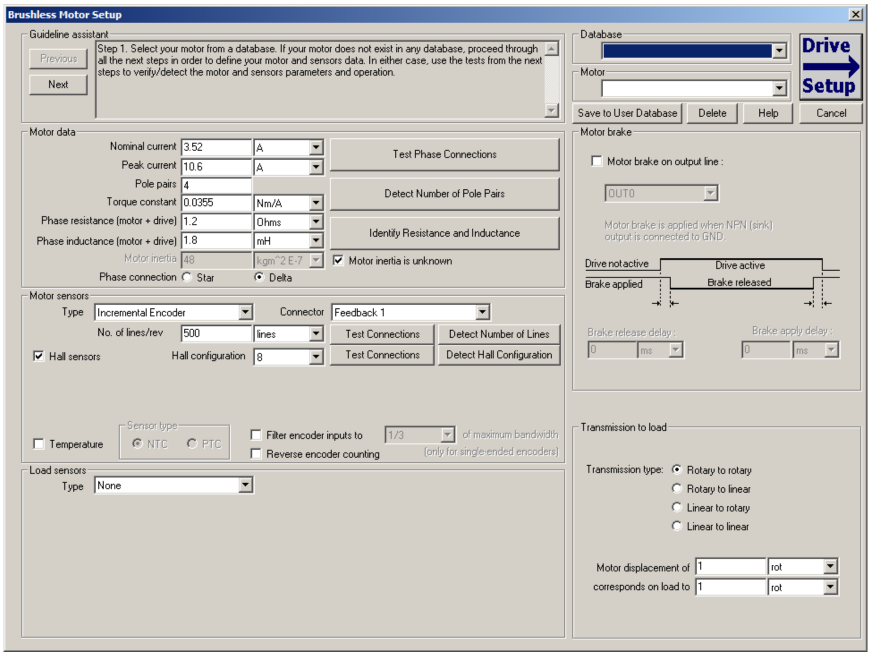


***Figure 1.2** PRO Config – Selecting the drive, motor and feedback*

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.

Introducing motor data



Brushless Motor Setup

Guideline assistant

Step 1. Select your motor from a database. If your motor does not exist in any database, proceed through all the next steps in order to define your motor and sensors data. In either case, use the tests from the next steps to verify/detect the motor and sensors parameters and operation.

Motor data

Nominal current	3.52	A
Peak current	10.6	A
Pole pairs	4	
Torque constant	0.0355	Nm/A
Phase resistance (motor + drive)	1.2	Ohms
Phase inductance (motor + drive)	1.8	mH
Motor inertia	48	kgm^2 E-7

☑ Motor inertia is unknown

Phase connection ○ Star ● Delta

Motor sensors

Type Incremental Encoder Connector Feedback 1

No. of lines/rev 500 lines

☑ Hall sensors Hall configuration 8

□ Temperature Sensor type ● NTC ○ PTC

□ Filter encoder inputs to 1/3 of maximum bandwidth

□ Reverse encoder counting (only for single-ended encoders)

Load sensors

Type None

Motor brake

□ Motor brake on output line : OUT0

Motor brake is applied when NPN (sink) output is connected to GND.

Brake release delay : 0 ms

Brake apply delay : 0 ms

Transmission to load

Transmission type: ● Rotary to rotary ○ Rotary to linear ○ Linear to rotary ○ Linear to linear

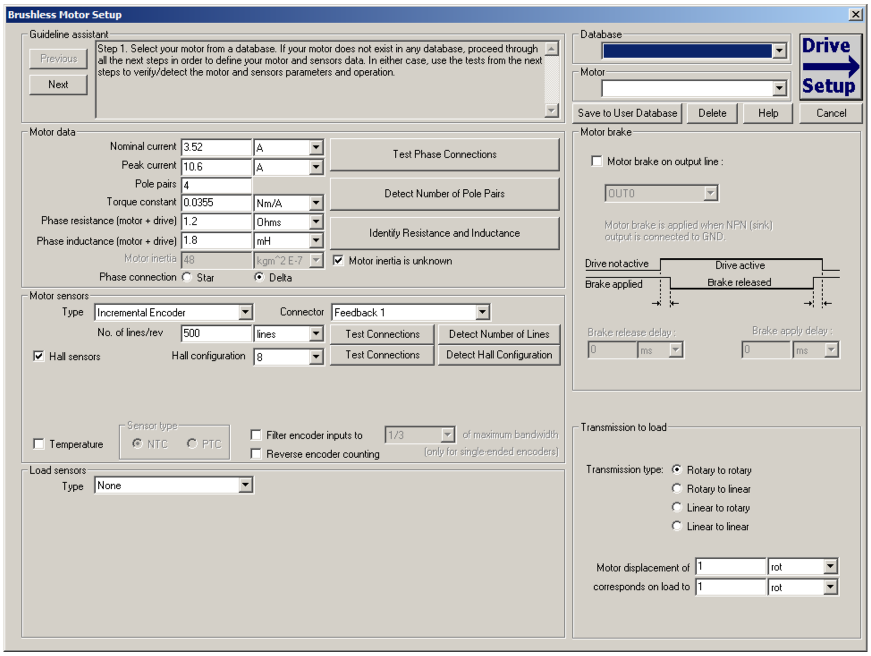
Motor displacement of 1 rot

corresponds on load to 1 rot

Figure 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*



Step 1. Select your motor from a database. If your motor does not exist in any database, proceed through all the next steps in order to define your motor and sensors data. In either case, use the tests from the next steps to verify/detect the motor and sensors parameters and operation.

Nominal current 3.52 A
Peak current 10.6 A
Pole pairs 4
Torque constant 0.0355 Nm/A
Phase resistance (motor + drive) 1.2 Ohms
Phase inductance (motor + drive) 1.8 mH
Motor inertia 48 kgm^2 E-7 ☑ Motor inertia is unknown
Phase connection ○ Star ● Delta

Type Incremental Encoder Connector Feedback 1
No. of lines/rev 500 lines
☑ Hall sensors Hall configuration 8
☐ Temperature Sensor type ● NTC ○ PTC
☐ Filter encoder inputs to 1/3 of maximum bandwidth (only for single-ended encoders)
☐ Reverse encoder counting

Load sensors Type None

☐ Motor brake on output line : OUT0
Motor brake is applied when NPN (sink) output is connected to GND.
Brake release delay : 0 ms Brake apply delay : 0 ms

Transmission type: ● Rotary to rotary ○ Rotary to linear ○ Linear to rotary ○ Linear to linear
Motor displacement of 1 rot
corresponds on load to 1 rot

***Figure 1.3** PRO Config – Introducing motor data*

1.1.5 Commissioning the drive

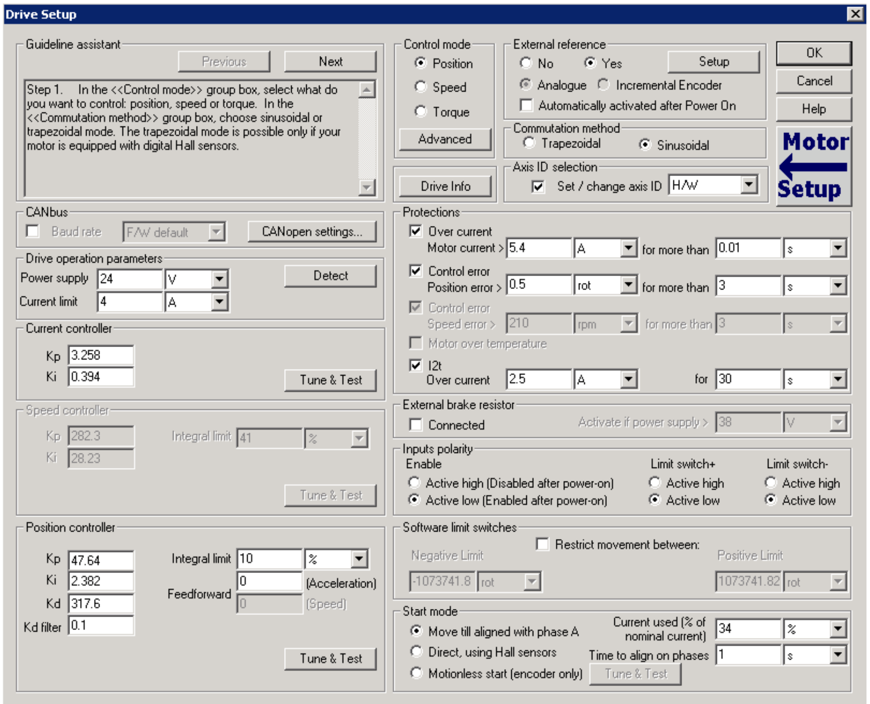
Figure 1.4 shows the **Drive setup** dialogue where you can configure and parameterize the drive for your application. Use the **Guideline Assistant**, and follow the steps described. This will guide you through the whole process of setting up the drive. Use the **Next** button to see the next guideline step and the **Previous** button to return to the previous step.

Close the Drive setup dialogue with **OK** to preserve all the changes done in both motor and drive setup dialogues.

***Remarks**:*

1) *Press the **Help** button from the Drive setup dialogue for detailed information*
2) *Set the motor **Over current** protection level below the motor **Peak current** value. This shall protect the motor against accidental high currents bypassing its **Peak current** value*
3) *When motor I2t protection is enabled, set its **Over current** value over the motor **Nominal current***
4) *Set the drive **Current limit** equal or below the motor **Over current** protection level. During a hard stop homing (no. -1 to -4) you can temporary reduce the Current limit via*

Object 207Fh: Current limit *to avoid triggering the protection during the homing*



Drive Setup

Guideline assistant

Step 1. In the <<Control mode>> group box, select what do you want to control: position, speed or torque. In the <<Commutation method>> group box, choose sinusoidal or trapezoidal mode. The trapezoidal mode is possible only if your motor is equipped with digital Hall sensors.

CANbus: □ Baud rate F/W default

Drive operation parameters: Power supply 24 V; Current limit 4 A

Current controller: Kp 3.258; Ki 0.394

Speed controller: Kp 282.3; Ki 28.23; Integral limit 41 %

Position controller: Kp 47.64; Ki 2.382; Kd 317.6; Kd filter 0.1; Integral limit 10 %; Feedforward 0 (Acceleration), 0 (Speed)

Control mode: ● Position ○ Speed ○ Torque

External reference: ○ No ● Yes; ● Analogue ○ Incremental Encoder; □ Automatically activated after Power On

Commutation method: ○ Trapezoidal ● Sinusoidal

Axis ID selection: ☑ Set / change axis ID H/W

Protections

☑ Over current: Motor current > 5.4 A for more than 0.01 s

☑ Control error: Position error > 0.5 rot for more than 3 s

☑ Control error: Speed error > 210 rpm for more than 3 s

□ Motor over temperature

☑ I2t: Over current 2.5 A for 30 s

External brake resistor: □ Connected; Activate if power supply > 38 V

Inputs polarity

Enable: ○ Active high (Disabled after power-on) ● Active low (Enabled after power-on)

Limit switch+: ○ Active high ● Active low

Limit switch-: ○ Active high ● Active low

Software limit switches: □ Restrict movement between: Negative Limit -1073741.8 rot; Positive Limit 1073741.82 rot

Start mode: ● Move till aligned with phase A ○ Direct, using Hall sensors ○ Motionless start (encoder only)

Current used (% of nominal current) 34 %; Time to align on phases 1 s

Figure 1.4 *PRO Config – Commissioning the drive*

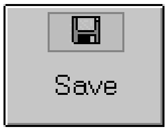
1.1.6 Downloading setup data to drive/motor

Closing the Drive setup dialogue with **OK**, keeps the new settings only in the PRO Config 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.

1.1.7 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

1.1.8 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 an EtherCAT® master, using the communication objects for writing data into the drive EEPROM (see **Chapter 15** for detailed example)
- using the EEPROM Programmer tool, which comes with PRO Config but may also be installed separately. The EEPROM Programmer was specifically designed for repetitive fast and easy programming of **.sw** files into the ElectroCraft drives during production

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

You can program an EtherCAT® master to automatically check after power on if all the ElectroCraft drives connected to the EtherCAT® network have the right setup data stored in their EEPROM. The comparison shall be done with the reference .sw files of each axis. These need to be loaded into the EtherCAT® master. There fastest way to compare a .sw file with a drive EEPROM contents is by comparing the checksums computed on the .sw file data with those computed by the drive on the same address range. In case of mismatch, the reference .sw file has to be reloaded into the drive by the EtherCAT® master. Par 16.4 and 16.5 present examples how to program a .sw file in a drive and how to check its consistency versus a .sw reference file

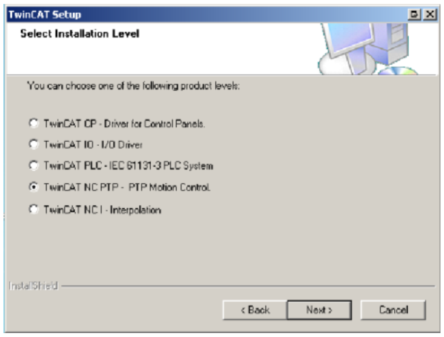
1.1.10 Testing and monitoring the drive behavior

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.

1.2 Setting up EtherCAT® communication. Example with TwinCAT2

This paragraph shows how to set up the EtherCAT® communication using Beckhoff TwinCAT2 software, running on a PC and provides several examples how to program the drive for different modes of operation: position profile, cyclic synchronous position and position-time interpolated. Another example shows how to map objects in TxPDOs and RxPDOs.

The TwinCAT version used in all examples presented is the 30 day free version 2.11 build 2239 downloaded from www.beckhoff.de with TwinCAT NC PTP installation level.



TwinCAT Setup

Select Installation Level

You can choose one of the following product levels:

○ TwinCAT CP - Driver for Control Panels.

○ TwinCAT IO - I/O Driver

○ TwinCAT PLC - IEC 61131-3 PLC System

● TwinCAT NC PTP - PTP Motion Control.

○ TwinCAT NC I - Interpolation

1.2.1 Adding the XML file

You can find on ElectroCraft web page the .xml file of each EtherCAT drive. After TwinCAT installation is complete, copy the appropriate .xml file for your product in TwinCAT installation folder \Io\EtherCAT. The default location is “C:\TwinCAT\Io\EtherCAT”.

1.2.2 Understanding EtherCAT® addressing modes supported

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

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

a) By hardware[1], function of the voltage levels of the axis ID inputs. The AxisID value is computed in the same way as in the case of PRO-CAN drives using the CANopen® protocol.

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

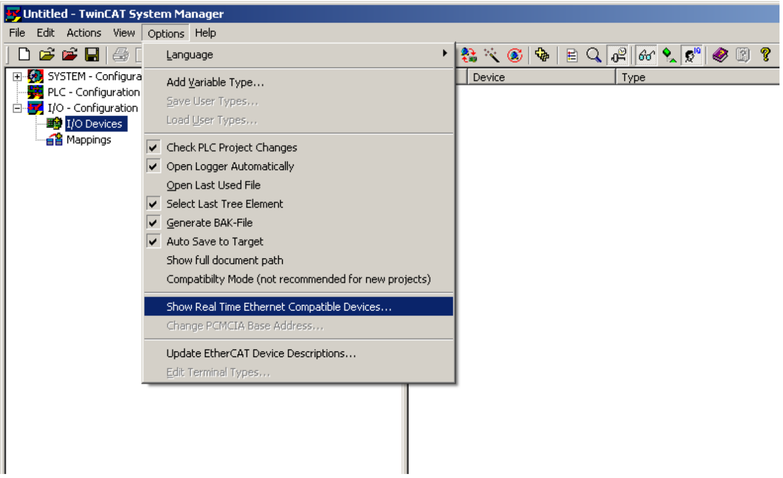
b) By software, imposed via PRO Config a specific AxisID value in the range 1-255.

1.2.3 Detecting the drive

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

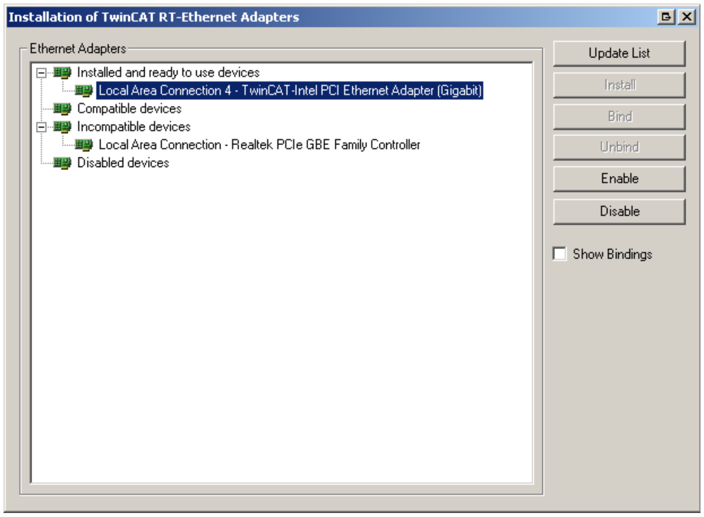
Start TwinCAT System Manager. Select the menu command *Options->Show Real Time Ethernet Compatible Devices…*

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

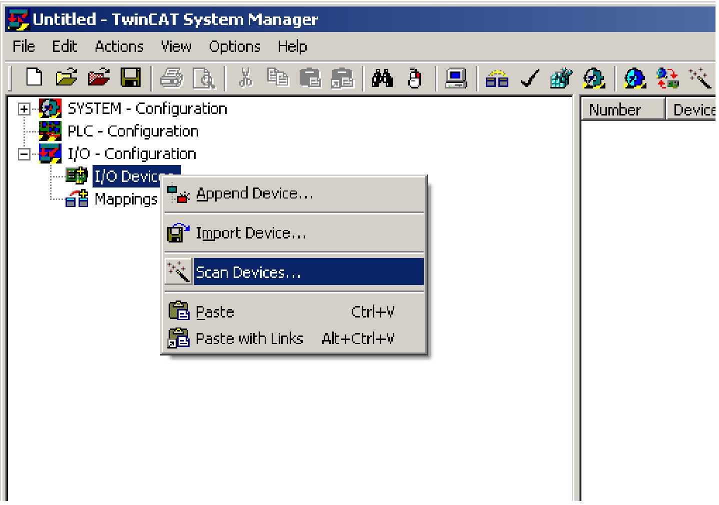

Untitled - TwinCAT System Manager
File Edit Actions View Options Help
SYSTEM - Configura
PLC - Configuration
I/O - Configuration
I/O Devices
Mappings
Device
Type
Language
Add Variable Type...
Save User Types...
Load User Types...
☑ Check PLC Project Changes
☑ Open Logger Automatically
Open Last Used File
☑ Select Last Tree Element
☑ Generate BAK-File
☑ Auto Save to Target
Show full document path
Compatibilty Mode (not recommended for new projects)
Show Real Time Ethernet Compatible Devices...
Change PCMCIA Base Address...
Update EtherCAT Device Descriptions...
Edit Terminal Types...

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

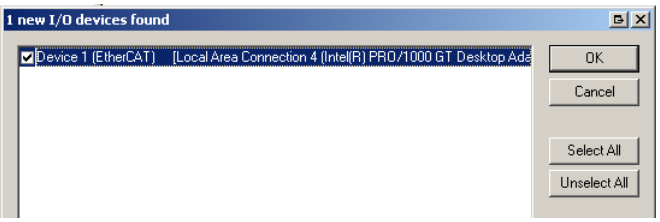
Remark: *In most cases, this operation is done automatically when TwinCAT is installed*


Installation of TwinCAT RT-Ethernet Adapters
Ethernet Adapters
Installed and ready to use devices
Local Area Connection 4 - TwinCAT-Intel PCI Ethernet Adapter (Gigabit)
Compatible devices
Incompatible devices
Local Area Connection - Realtek PCIe GBE Family Controller
Disabled devices
Update List
Install
Bind
Unbind
Enable
Disable
□ Show Bindings

On the left tree of the TwinCAT System Manager, select *I/O Configuration,* then *I/O devices*, then click the right mouse button and choose *Scan Devices...* Confirm the next dialogue with OK.

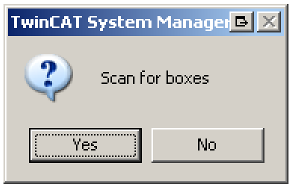


In the list of EtherCAT® devices found, select only the ones you want to add to the list and click OK.


1 new I/O devices found

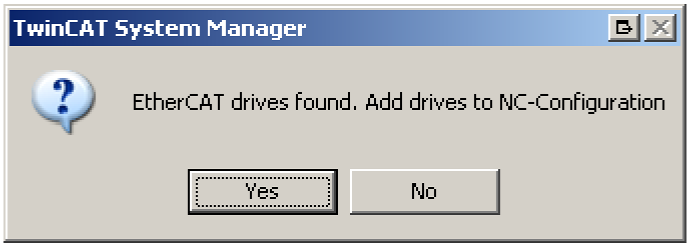
☑ Device 1 (EtherCAT) [Local Area Connection 4 (Intel(R) PRO/1000 GT Desktop Ada

The following dialog appears. Click Yes to confirm:


TwinCAT System Manager

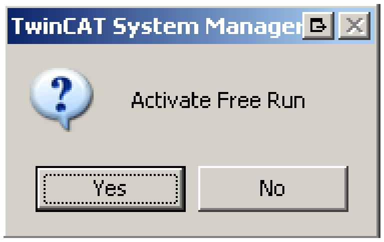
Scan for boxes

Click Yes, to add drives to NC-Configuration.


TwinCAT System Manager

EtherCAT drives found. Add drives to NC-Configuration

Click Yes to activate Free run in the next dialogue.



Activate Free Run

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

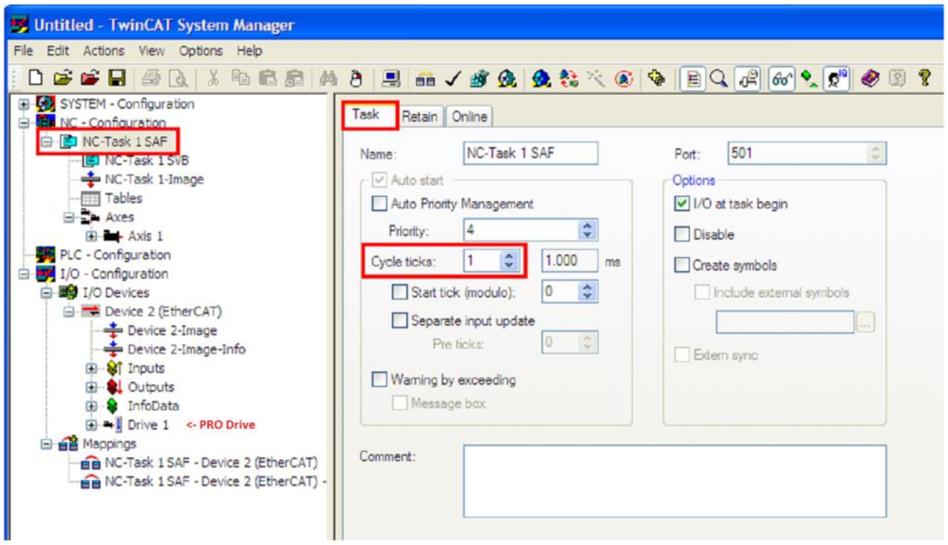
1.2.4 Configuring ElectroCraft EtherCAT drives for NC PTP compatibility

After the instructions from chapter **1.2.3** are complete, do the following:

1.2.4.1 Setting the communication cycle time for RUN mode

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

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



Name: NC-Task 1 SAF

Port: 501

☑ Auto start

☐ Auto Priority Management

Priority: 4

Cycle ticks: 1 1.000 ms

☐ Start tick (modulo): 0

☐ Separate input update

Pre ticks: 0

☐ Warning by exceeding

☐ Message box

☑ I/O at task begin

☐ Disable

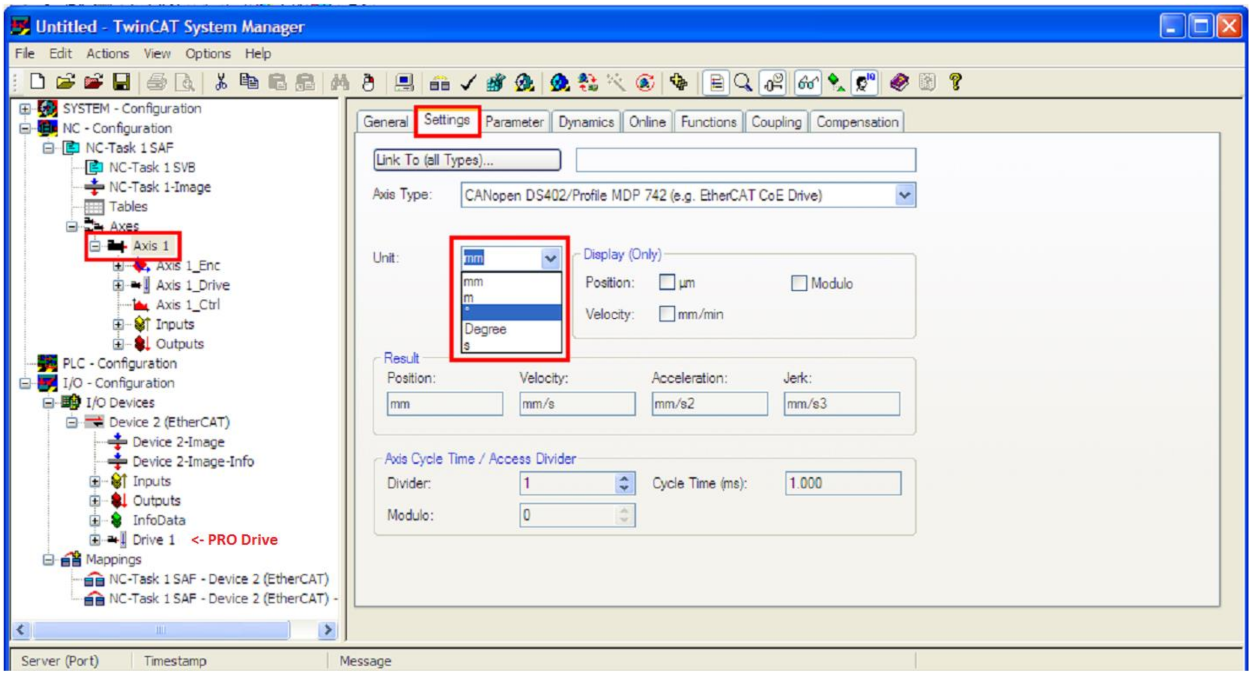
☐ Create symbols

☐ Include external symbols

☐ Extern sync

1.2.4.2 Setting the interface factor group settings

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



Axis Type: CANopen DS402/Profile MDP 742 (e.g. EtherCAT CoE Drive)

Unit: mm

Display (Only)

Position: □ µm □ Modulo

Velocity: □ mm/min

Result

Position:	Velocity:	Acceleration:	Jerk:
mm	mm/s	mm/s2	mm/s3

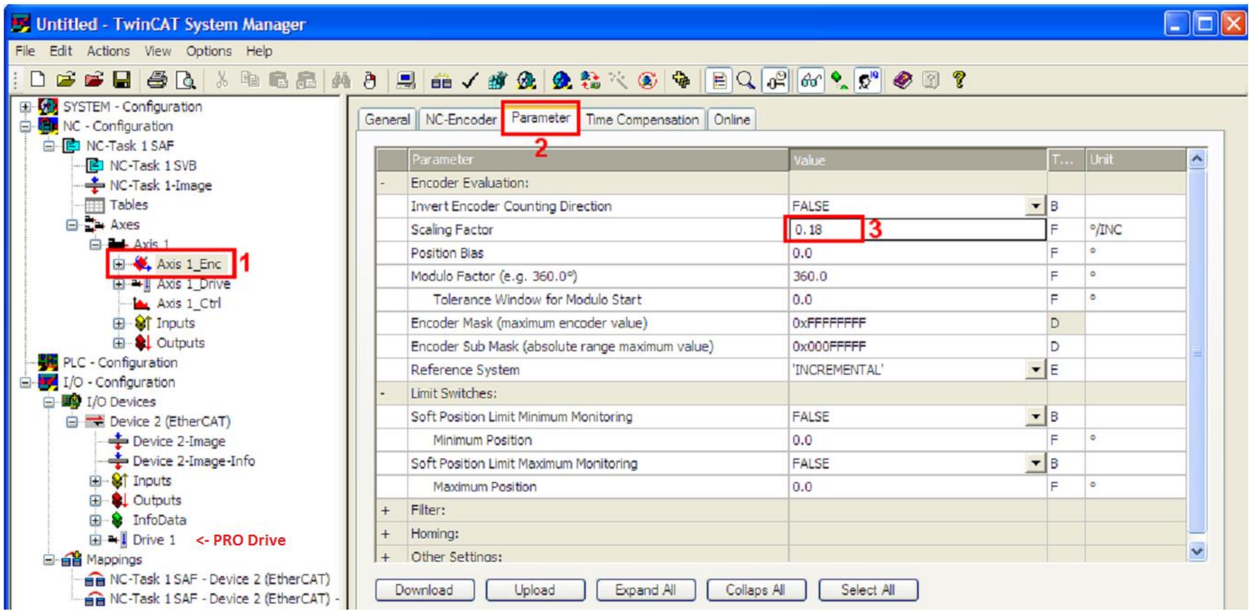
Axis Cycle Time / Access Divider

Divider: 1 Cycle Time (ms): 1.000

Modulo: 0

Setting the scaling factor:

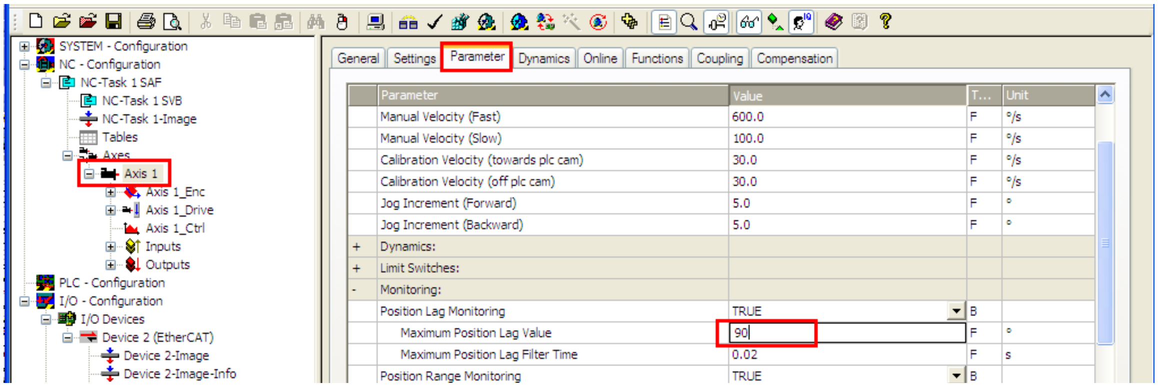
Click Axis 1_Enc(1) and select the Parameter(2) tab. Write the scaling factor for your encoder (3). The formula is 360◦/ Number of encoder counts of one full motor rotation. So, if you have a 500 line quadrature encoder, you will have 2000 encoder counts per rotation. So the scaling factor in this case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



	Parameter	Value	T...	Unit
-	Encoder Evaluation:			
	Invert Encoder Counting Direction	FALSE	B	
	Scaling Factor	0.18	F	°/INC
	Position Bias	0.0	F	°
	Modulo Factor (e.g. 360.0°)	360.0	F	°
	Tolerance Window for Modulo Start	0.0	F	°
	Encoder Mask (maximum encoder value)	0xFFFFFFFF	D	
	Encoder Sub Mask (absolute range maximum value)	0x000FFFFF	D	
	Reference System	'INCREMENTAL'	E	
-	Limit Switches:			
	Soft Position Limit Minimum Monitoring	FALSE	B	
	Minimum Position	0.0	F	°
	Soft Position Limit Maximum Monitoring	FALSE	B	
	Maximum Position	0.0	F	°
+	Filter:			
+	Homing:			
+	Other Settings:			

1.2.4.3 Choosing a position lag value

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

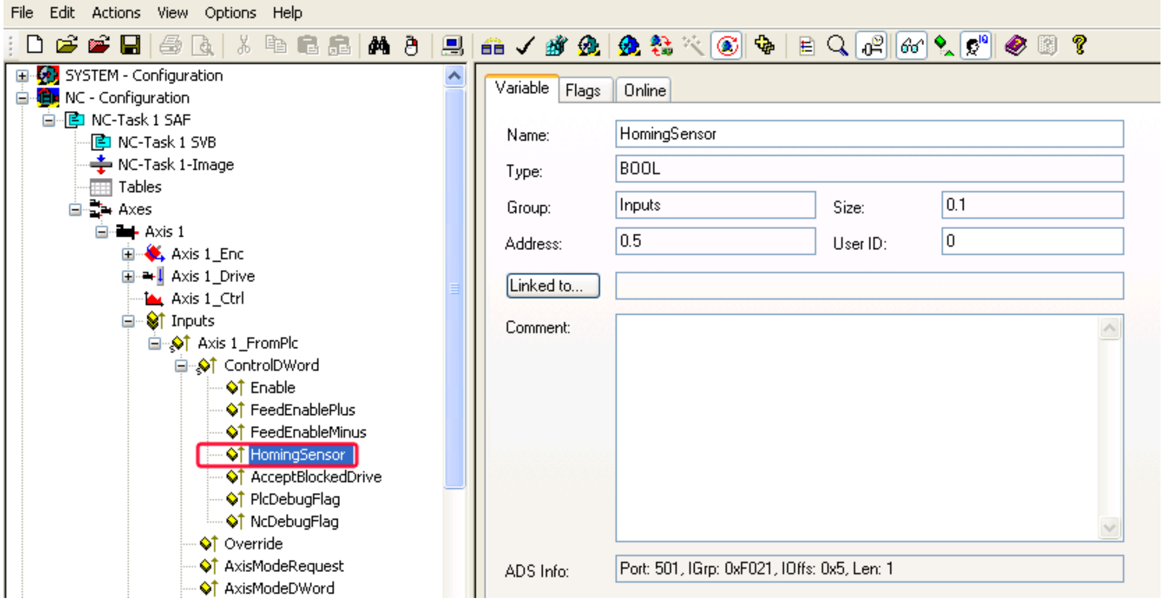


	Parameter	Value	T...	Unit
	Manual Velocity (Fast)	600.0	F	°/s
	Manual Velocity (Slow)	100.0	F	°/s
	Calibration Velocity (towards plc cam)	30.0	F	°/s
	Calibration Velocity (off plc cam)	30.0	F	°/s
	Jog Increment (Forward)	5.0	F	°
	Jog Increment (Backward)	5.0	F	°
+	Dynamics:			
+	Limit Switches:			
-	Monitoring:			
	Position Lag Monitoring	TRUE	B	
	Maximum Position Lag Value	90	F	°
	Maximum Position Lag Filter Time	0.02	F	s
	Position Range Monitoring	TRUE	B	

Remark: the position lag protection also exists on all ElectroCraft drives and can be set with PRO Config under the “Control error/ Position error” fields.

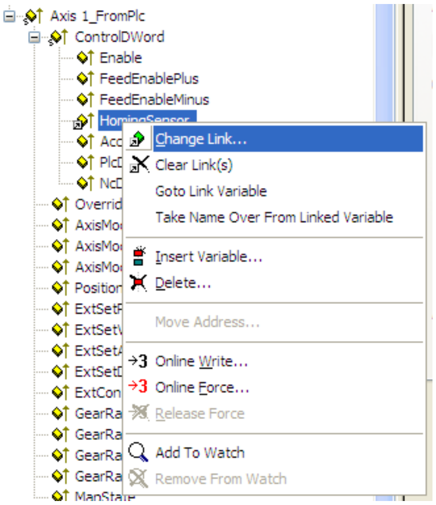
1.2.4.4 Mapping a digital input as the home switch for the NC-PTP interface

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

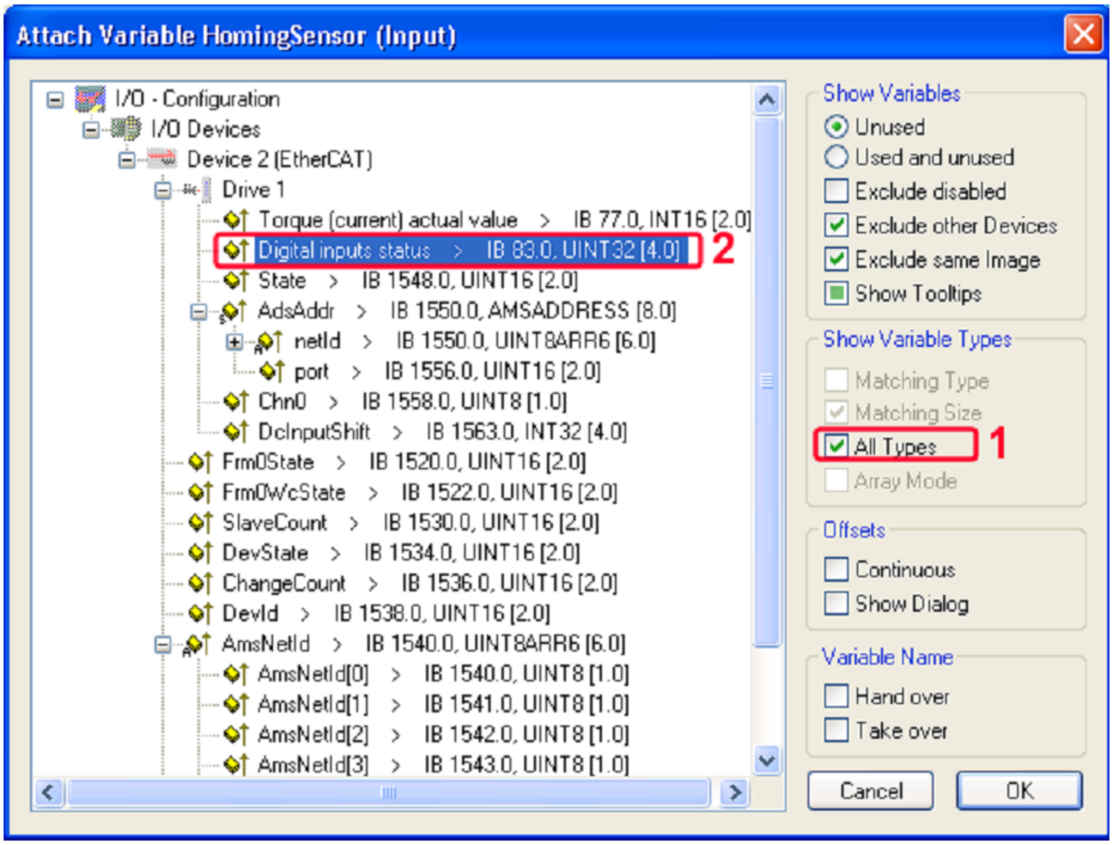


Name:	HomingSensor		
Type:	BOOL		
Group:	Inputs	Size:	0.1
Address:	0.5	User ID:	0
ADS Info:	Port: 501, IGrp: 0xF021, IOffs: 0x5, Len: 1		

Right click the variable, and select change link… .



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


Attach Variable HomingSensor (Input)

```
I/O - Configuration
  I/O Devices
    Device 2 (EtherCAT)
      Drive 1
        Torque (current) actual value  >  IB 77.0, INT16 [2.0]
        Digital inputs status  >  IB 83.0, UINT32 [4.0]
        State  >  IB 1548.0, UINT16 [2.0]
        AdsAddr  >  IB 1550.0, AMSADDRESS [8.0]
          netId  >  IB 1550.0, UINT8ARR6 [6.0]
          port  >  IB 1556.0, UINT16 [2.0]
        Chn0  >  IB 1558.0, UINT8 [1.0]
        DcInputShift  >  IB 1563.0, INT32 [4.0]
      Frm0State  >  IB 1520.0, UINT16 [2.0]
      Frm0WcState  >  IB 1522.0, UINT16 [2.0]
      SlaveCount  >  IB 1530.0, UINT16 [2.0]
      DevState  >  IB 1534.0, UINT16 [2.0]
      ChangeCount  >  IB 1536.0, UINT16 [2.0]
      DevId  >  IB 1538.0, UINT16 [2.0]
      AmsNetId  >  IB 1540.0, UINT8ARR6 [6.0]
        AmsNetId[0]  >  IB 1540.0, UINT8 [1.0]
        AmsNetId[1]  >  IB 1541.0, UINT8 [1.0]
        AmsNetId[2]  >  IB 1542.0, UINT8 [1.0]
        AmsNetId[3]  >  IB 1543.0, UINT8 [1.0]
```

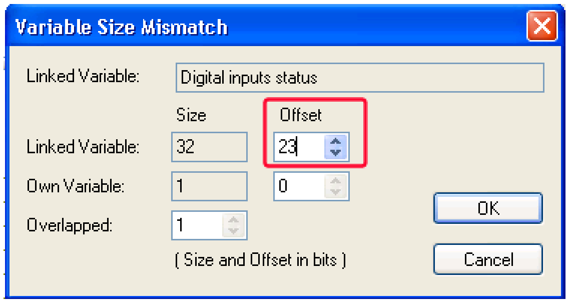
Show Variables
- ● Unused
- ○ Used and unused
- □ Exclude disabled
- ☑ Exclude other Devices
- ☑ Exclude same Image
- ■ Show Tooltips

Show Variable Types
- □ Matching Type
- ☑ Matching Size
- ☑ All Types
- □ Array Mode

Offsets
- □ Continuous
- □ Show Dialog

Variable Name
- □ Hand over
- □ Take over

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


Variable Size Mismatch

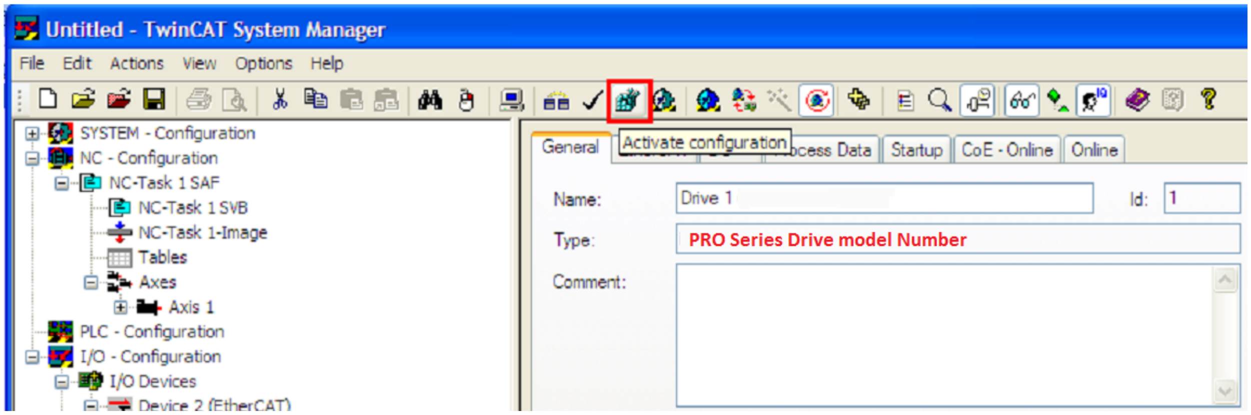
Linked Variable: Digital inputs status

	Size	Offset
Linked Variable:	32	23
Own Variable:	1	0
Overlapped:	1	

(Size and Offset in bits)

1.2.5 Running the NC-PTP interface

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


Untitled - TwinCAT System Manager

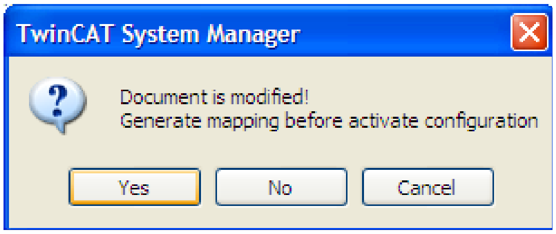
Name: Drive 1

Id: 1

Type: PRO Series Drive model Number

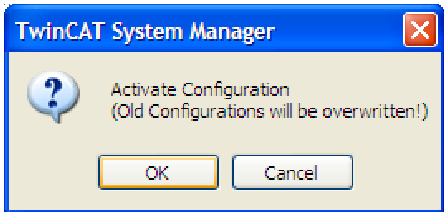
Comment:

Click Yes.


TwinCAT System Manager

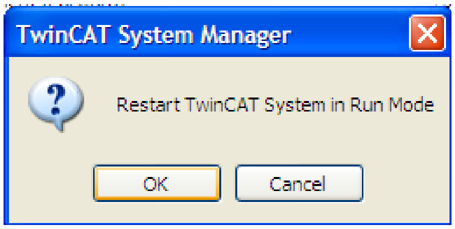
Document is modified!
Generate mapping before activate configuration

Click OK.


TwinCAT System Manager

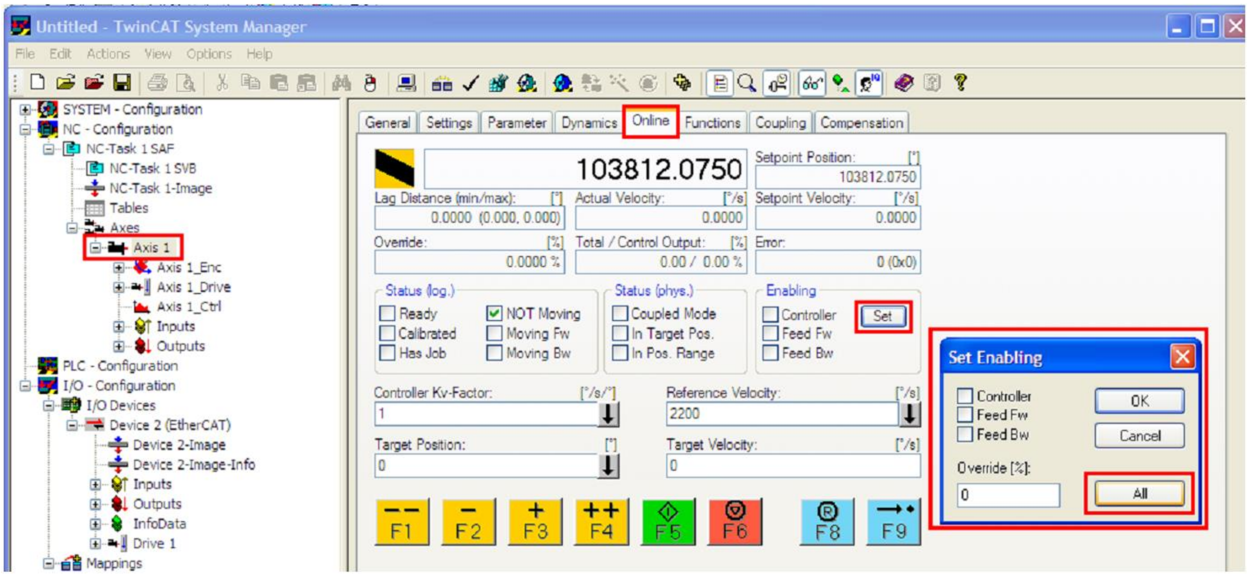
Activate Configuration
(Old Configurations will be overwritten!)

Click OK.


TwinCAT System Manager

Restart TwinCAT System in Run Mode

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


Untitled - TwinCAT System Manager

103812.0750

Setpoint Position: [°] 103812.0750

Lag Distance (min/max): [°] 0.0000 (0.000, 0.000)

Actual Velocity: [°/s] 0.0000

Setpoint Velocity: [°/s] 0.0000

Override: [%] 0.0000 %

Total / Control Output: [%] 0.00 / 0.00 %

Error: 0 (0x0)

Status (log.): □ Ready □ Calibrated □ Has Job ☑ NOT Moving □ Moving Fw □ Moving Bw

Status (phys.): □ Coupled Mode □ In Target Pos. □ In Pos. Range

Enabling: □ Controller □ Feed Fw □ Feed Bw

Controller Kv-Factor: [°/s/°] 1

Reference Velocity: [°/s] 2200

Target Position: [°] 0

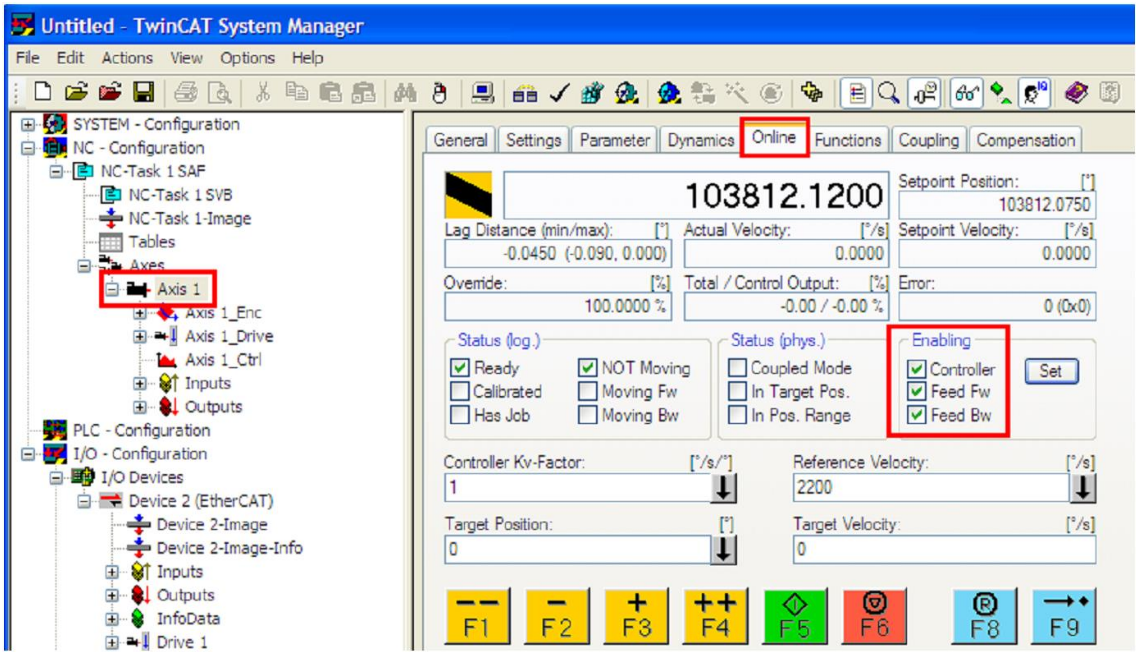
Target Velocity: [°/s] 0

Set Enabling

□ Controller □ Feed Fw □ Feed Bw

Override [%]: 0

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



103812.1200

Setpoint Position: [°] 103812.0750

Lag Distance (min/max): [°] -0.0450 (-0.090, 0.000)

Actual Velocity: [°/s] 0.0000

Setpoint Velocity: [°/s] 0.0000

Override: [%] 100.0000 %

Total / Control Output: [%] -0.00 / -0.00 %

Error: 0 (0x0)

☑ Ready ☑ NOT Moving ☐ Coupled Mode ☑ Controller

☐ Calibrated ☐ Moving Fw ☐ In Target Pos. ☑ Feed Fw

☐ Has Job ☐ Moving Bw ☐ In Pos. Range ☑ Feed Bw

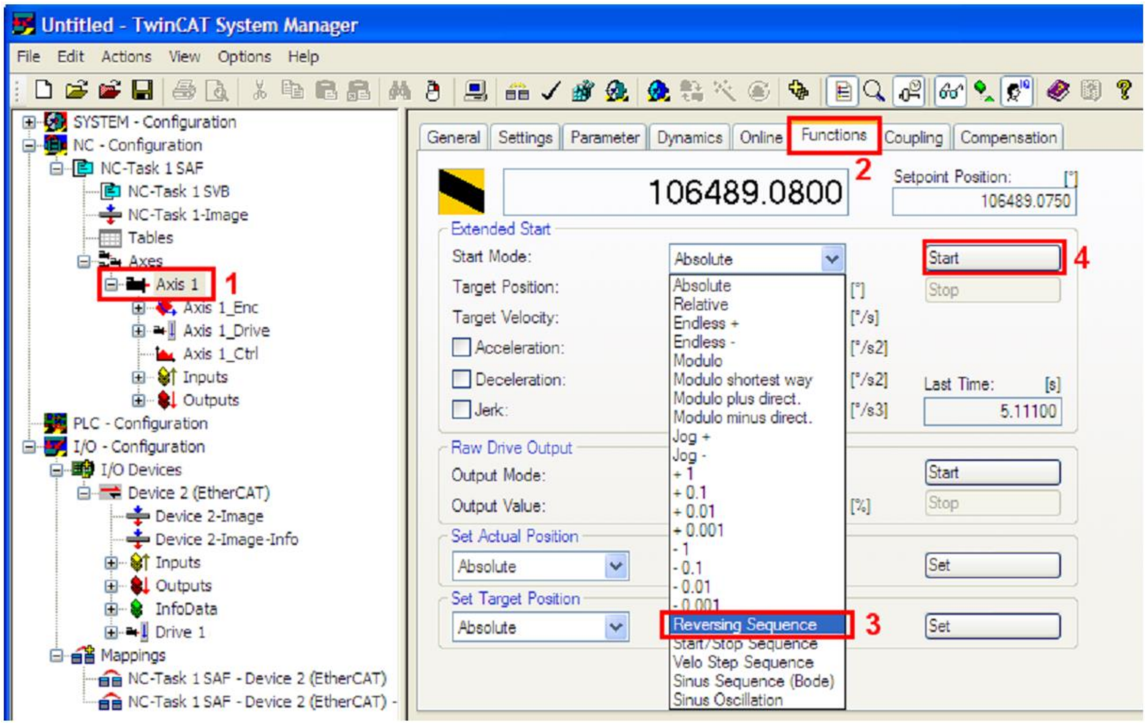
Controller Kv-Factor: [°/s/°] 1

Reference Velocity: [°/s] 2200

Target Position: [°] 0

Target Velocity: [°/s] 0

Click the Functions Tab(2) and choose a Reversing Sequence (3).



106489.0800

Setpoint Position: [°] 106489.0750

Start Mode: Absolute

Target Position: [°]

Target Velocity: [°/s]

☐ Acceleration: [°/s2]

☐ Deceleration: [°/s2]

☐ Jerk: [°/s3]

Last Time: [s] 5.11100

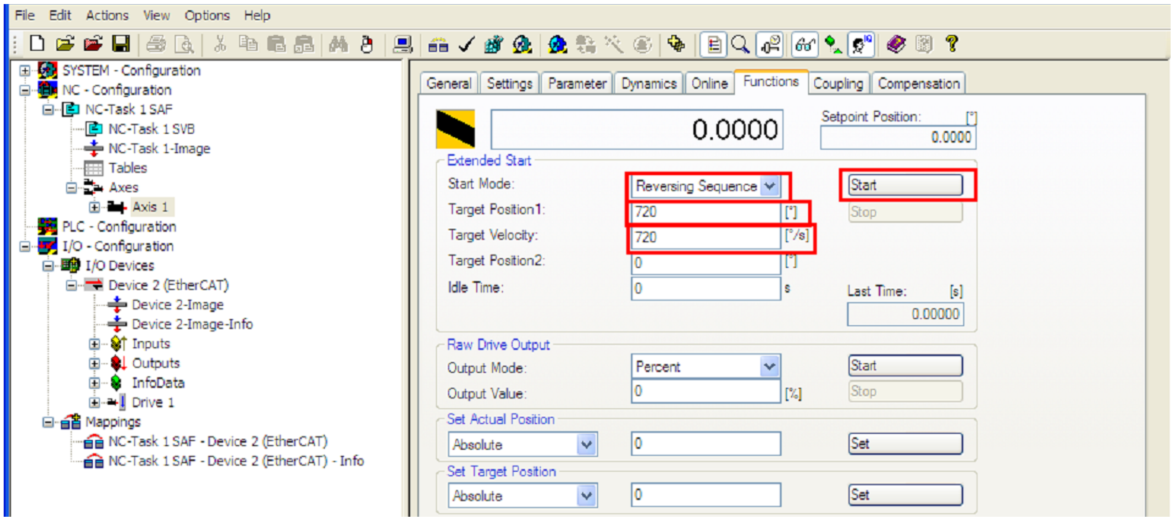
Output Mode:

Output Value: [%]

Set Actual Position: Absolute

Set Target Position: Absolute

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



0.0000

Setpoint Position: [°] 0.0000

Extended Start

Start Mode: Reversing Sequence

Target Position1: 720 [°]

Target Velocity: 720 [°/s]

Target Position2: 0 [°]

Idle Time: 0 s

Last Time: [s] 0.00000

Raw Drive Output

Output Mode: Percent

Output Value: 0 [%]

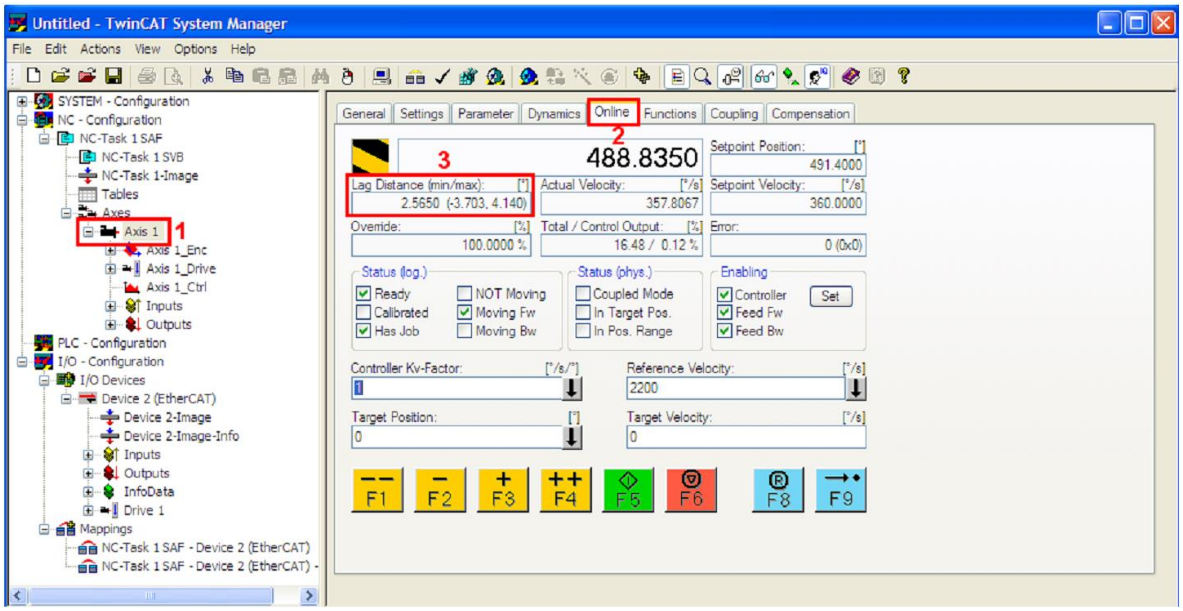
Set Actual Position

Absolute 0

Set Target Position

Absolute 0

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



488.8350

Setpoint Position: [°] 491.4000

Lag Distance (min/max): [°] 2.5650 (-3.703, 4.140)

Actual Velocity: [°/s] 357.8067

Setpoint Velocity: [°/s] 360.0000

Override: [%] 100.0000 %

Total / Control Output: [%] 16.48 / 0.12 %

Error: 0 (0x0)

Status (log.)

☑ Ready
☐ Calibrated
☑ Has Job
☐ NOT Moving
☑ Moving Fw
☐ Moving Bw

Status (phys.)

☐ Coupled Mode
☐ In Target Pos.
☐ In Pos. Range

Enabling

☑ Controller
☑ Feed Fw
☑ Feed Bw

Controller Kv-Factor: [°/s/°] 1

Reference Velocity: [°/s] 2200

Target Position: [°] 0

Target Velocity: [°/s] 0

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

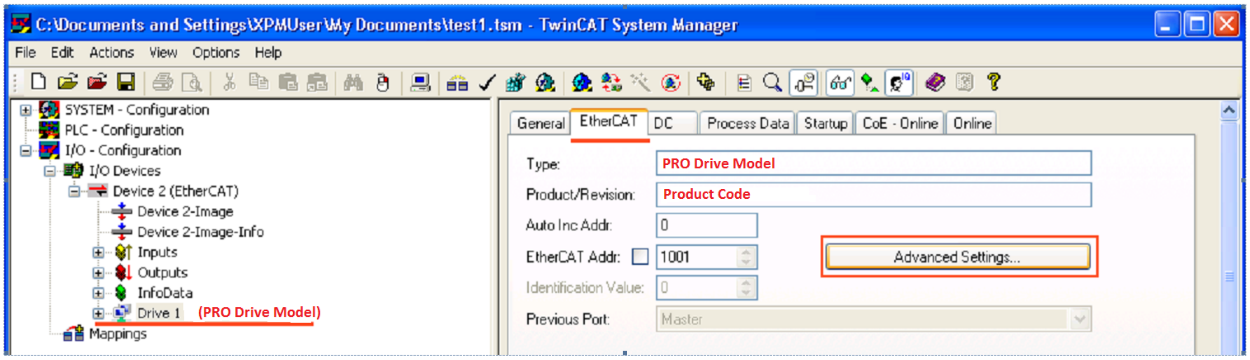
1.2.6 Checking and updating the XML file stored in the drive

In the particular case of the PRO Series Evaulation Kits with EtherCAT®, all PRO Series drives use the same EtherCAT® extension module. As this is interchangeable, it is possible to cross combine by mistake the drives and EtherCAT® interfaces. In this situation an PRO-A04V36 drive may work together with an

EtherCAT® extension having inside it the .xml file for PRO-A02V36 drive and vice versa. This condition will lead to incorrect drive identification when scanning devices in the network. For example the EtherCAT® master will see the PRO-A04V36 as an PRO-A02V36. To check if the drive is correctly identified, read the label on the drive, or read Object 1018_h sub index 2 which shows the correct Product code.

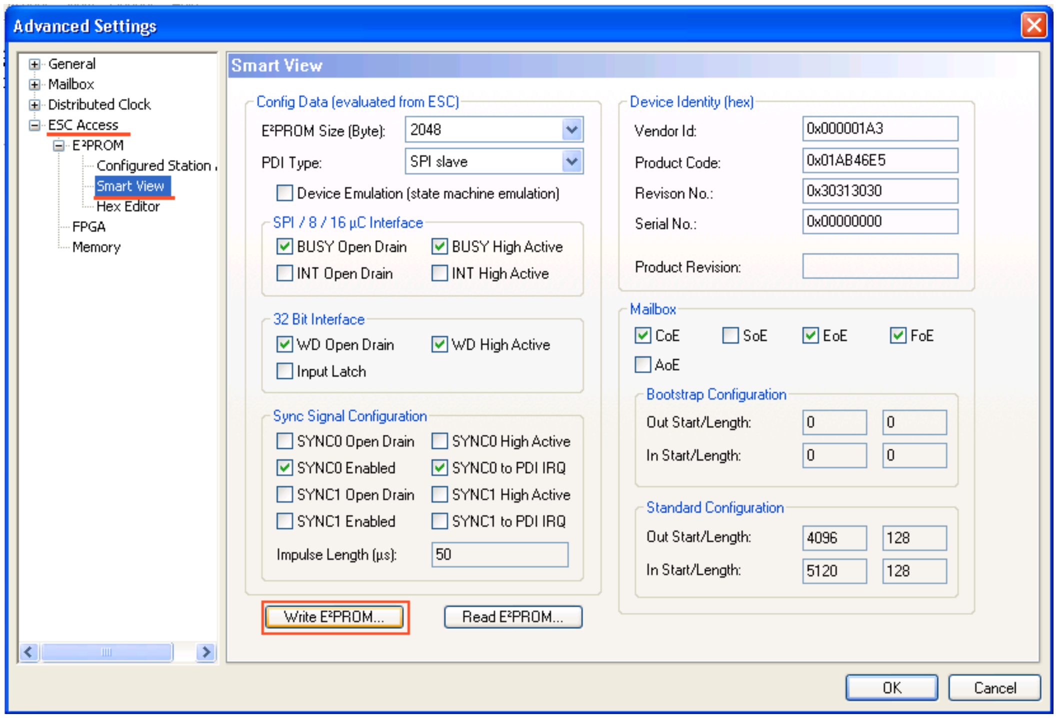
ElectroCraft Drive name	Product code in 1018_h, sub 02_h
PRO-A08V48B-SA-CAT	27214221
PRO-A10V80A-SA-CAT	29025221
PRO-A20V80A-SA-CAT	29026224

To write a new XML file, first make sure that this is present in the right folder i.e. in TwinCAT installation folder \Io\EtherCAT®. The default location is “C:\TwinCAT\Io\EtherCAT”. In TwinCAT System Manager, if you know the product ID of the drive, select the target drive from the left side menu. Select the *EtherCAT®* tab and click *Advanced Settings…*button. See figure below.



Type: PRO Drive Model
Product/Revision: Product Code
Auto Inc Addr: 0
EtherCAT Addr: □ 1001
Identification Value: 0
Previous Port: Master

In the Advanced Settings window, select from the left side tree option *ESC access/E^{2}PROM/ Smart View*. Click the *Write E^{2}PROM…* button.



Advanced Settings

- General
- Mailbox
- Distributed Clock
- ESC Access
  - E²PROM
    - Configured Station ...
    - Smart View
    - Hex Editor
  - FPGA
  - Memory

Smart View

Config Data (evaluated from ESC)

E²PROM Size (Byte): 2048

PDI Type: SPI slave

□ Device Emulation (state machine emulation)

SPI / 8 / 16 µC Interface

☑ BUSY Open Drain ☑ BUSY High Active

□ INT Open Drain □ INT High Active

32 Bit Interface

☑ WD Open Drain ☑ WD High Active

□ Input Latch

Sync Signal Configuration

□ SYNC0 Open Drain □ SYNC0 High Active

☑ SYNC0 Enabled ☑ SYNC0 to PDI IRQ

□ SYNC1 Open Drain □ SYNC1 High Active

□ SYNC1 Enabled □ SYNC1 to PDI IRQ

Impulse Length (µs): 50

Device Identity (hex)

Vendor Id: 0x000001A3

Product Code: 0x01AB46E5

Revison No.: 0x30313030

Serial No.: 0x00000000

Product Revision:

Mailbox

☑ CoE □ SoE ☑ EoE ☑ FoE

□ AoE

Bootstrap Configuration

Out Start/Length: 0 0

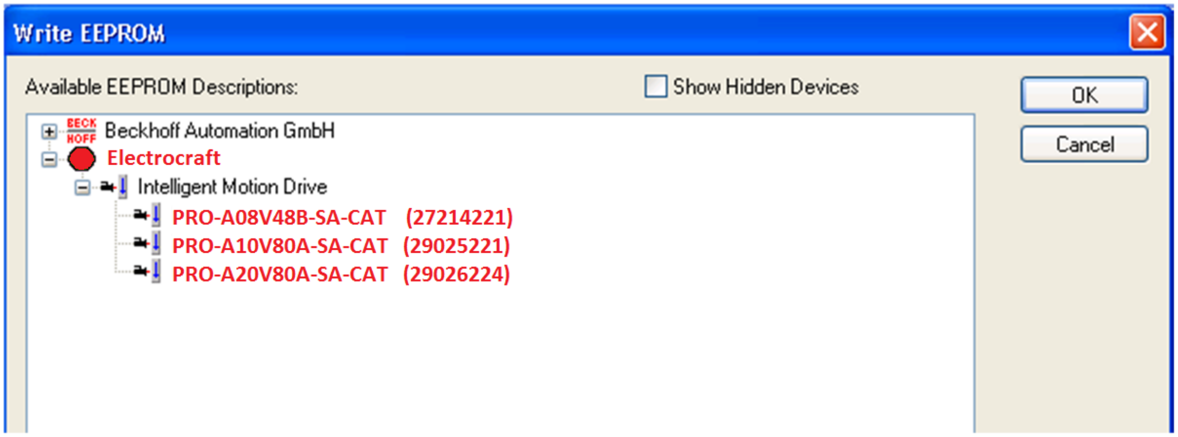
In Start/Length: 0 0

Standard Configuration

Out Start/Length: 4096 128

In Start/Length: 5120 128

If the XML file is present (see 1.2.1 Adding the XML file), a list of available product descriptions will be available. Choose your correct drive information and click OK.



Write EEPROM

Available EEPROM Descriptions: □ Show Hidden Devices

- Beckhoff Automation GmbH
- Electrocraft
  - Intelligent Motion Drive
    - PRO-A08V48B-SA-CAT (27214221)
    - PRO-A10V80A-SA-CAT (29025221)
    - PRO-A20V80A-SA-CAT (29026224)

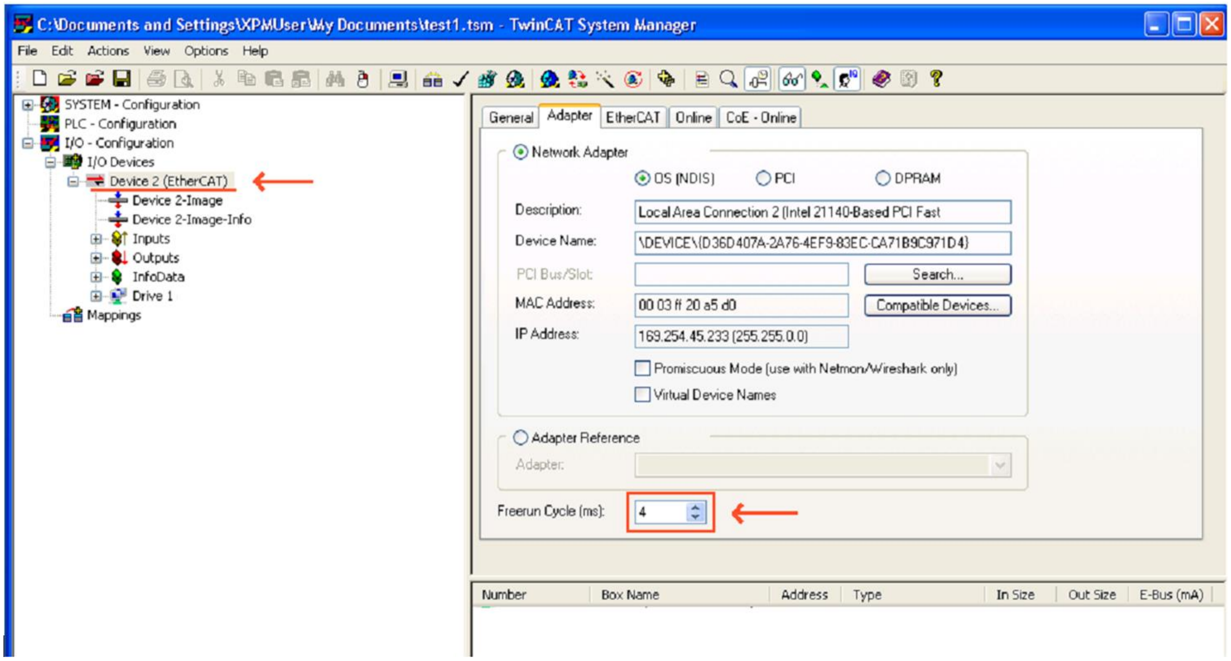
The first number represents the drive Product number and the second one represents the revision number. The revision number represents the current firmware version. It can be converted into hex and later into ASCII characters. So, if the current firmware F510D, the revision number will be 510D (ASCII), 0x35313044 (hex) and 892416068 (decimal).

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

1.2.7 Setting the free run communication cycle

In the left tree, at *I/O devices*, select *Device 2 (EtherCAT)*. On the right side, select *Adapter* tab. Set the *Freerun Cycle* (ms) for example at 1ms.

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



C:\Documents and Settings\XPMUser\My Documents\test1.tsm - TwinCAT System Manager

- SYSTEM - Configuration
- PLC - Configuration
- I/O - Configuration
  - I/O Devices
    - Device 2 (EtherCAT)
      - Device 2-Image
      - Device 2-Image-Info
      - Inputs
      - Outputs
      - InfoData
      - Drive 1
  - Mappings

Network Adapter: ● OS (NDIS) ○ PCI ○ DPRAM

Description: Local Area Connection 2 (Intel 21140-Based PCI Fast

Device Name: \DEVICE\{D36D407A-2A76-4EF9-83EC-CA71B9C971D4}

PCI Bus/Slot:

MAC Address: 00 03 ff 20 a5 d0

IP Address: 169.254.45.233 (255.255.0.0)

□ Promiscuous Mode (use with Netmon/Wireshark only)

□ Virtual Device Names

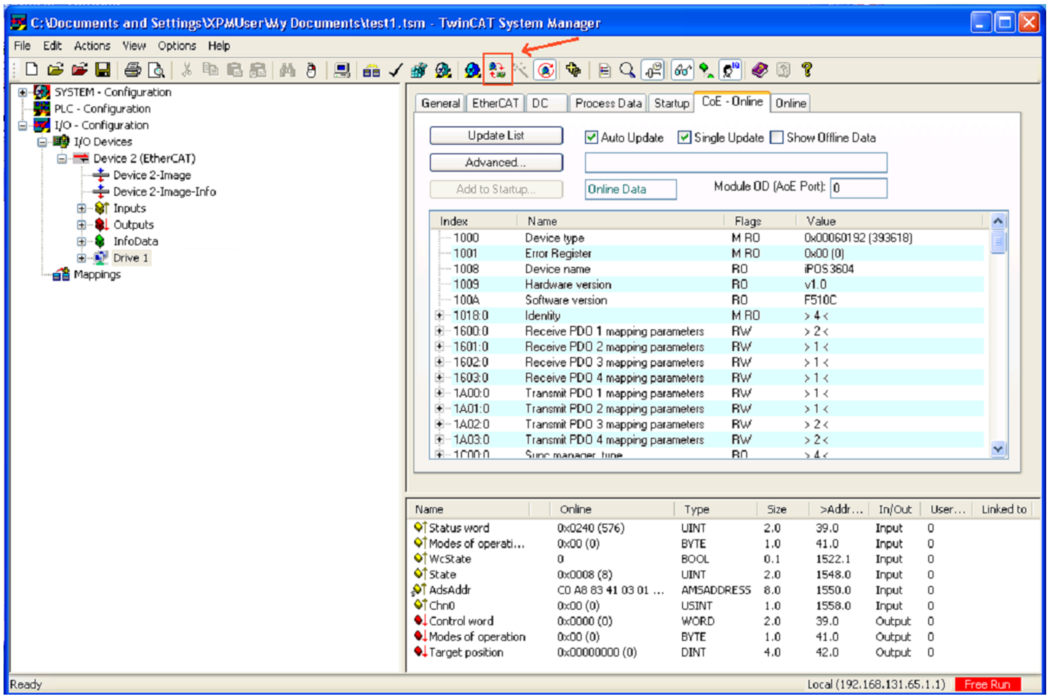
○ Adapter Reference

Adapter:

Freerun Cycle (ms): 4

Number	Box Name	Address	Type	In Size	Out Size	E-Bus (mA)

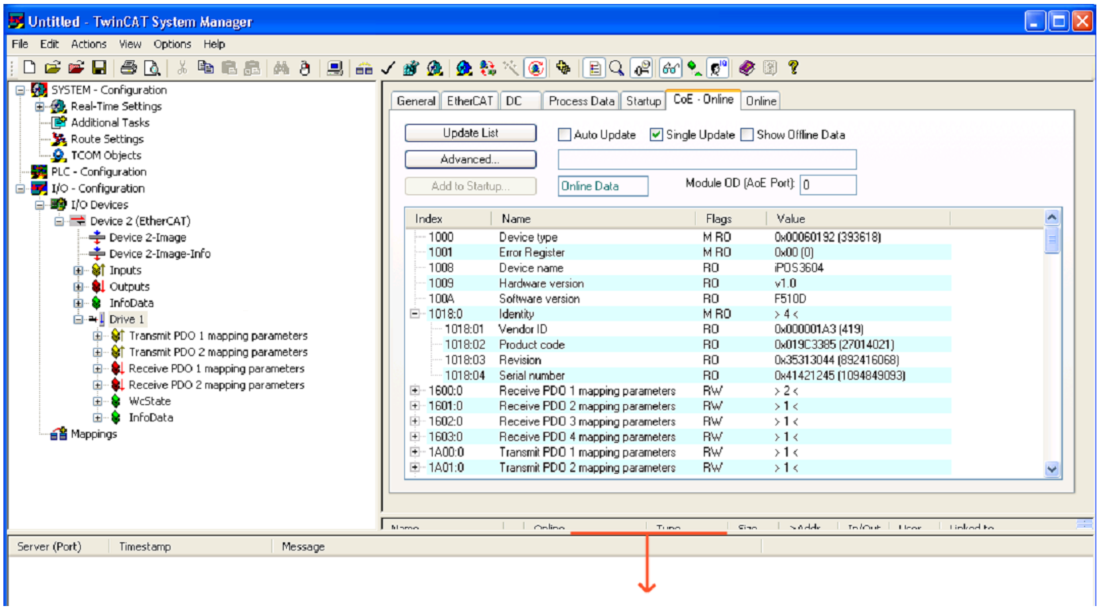
To activate all settings done so far (including PDO mapping changes) click the *Reload I/O Devices (F4)* button shown below. This will also activate SYNC 0 signal if it was enabled.


C:\Documents and Settings\XPMUser\My Documents\test1.tsm - TwinCAT System Manager
File Edit Actions View Options Help
SYSTEM - Configuration
PLC - Configuration
I/O - Configuration
I/O Devices
Device 2 (EtherCAT)
Device 2-Image
Device 2-Image-Info
Inputs
Outputs
InfoData
Drive 1
Mappings
General EtherCAT DC Process Data Startup CoE - Online Online
Update List
Advanced...
Add to Startup...
☑ Auto Update ☑ Single Update ☐ Show Offline Data
Online Data
Module OD (AoE Port): 0
Ready
Local (192.168.131.65.1.1)
Free Run

Index	Name	Flags	Value
1000	Device type	M RO	0x00060192 (393618)
1001	Error Register	M RO	0x00 (0)
1008	Device name	RO	iPOS3604
1009	Hardware version	RO	v1.0
100A	Software version	RO	F510C
1018:0	Identity	M RO	> 4 <
1600:0	Receive PDO 1 mapping parameters	RW	> 2 <
1601:0	Receive PDO 2 mapping parameters	RW	> 1 <
1602:0	Receive PDO 3 mapping parameters	RW	> 1 <
1603:0	Receive PDO 4 mapping parameters	RW	> 1 <
1A00:0	Transmit PDO 1 mapping parameters	RW	> 1 <
1A01:0	Transmit PDO 2 mapping parameters	RW	> 1 <
1A02:0	Transmit PDO 3 mapping parameters	RW	> 2 <
1A03:0	Transmit PDO 4 mapping parameters	RW	> 2 <
1C00:0	Sync manager type	RO	> 4 <

Name	Online	Type	Size	>Addr...	In/Out	User...	Linked to
Status word	0x0240 (576)	UINT	2.0	39.0	Input	0	
Modes of operati...	0x00 (0)	BYTE	1.0	41.0	Input	0	
WcState	0	BOOL	0.1	1522.1	Input	0	
State	0x0008 (8)	UINT	2.0	1548.0	Input	0	
AdsAddr	C0 A8 83 41 03 01 ...	AMSADDRESS	8.0	1550.0	Input	0	
Chn0	0x00 (0)	USINT	1.0	1558.0	Input	0	
Control word	0x0000 (0)	WORD	2.0	39.0	Output	0	
Modes of operation	0x00 (0)	BYTE	1.0	41.0	Output	0	
Target position	0x00000000 (0)	DINT	4.0	42.0	Output	0	

Remark: *On a lower resolution display, TwinCAT may not show the window with the mapped PDO data. Drag the line shown in the picture below, to see this information.*


Untitled - TwinCAT System Manager
File Edit Actions View Options Help
SYSTEM - Configuration
Real-Time Settings
Additional Tasks
Route Settings
TCOM Objects
PLC - Configuration
I/O - Configuration
I/O Devices
Device 2 (EtherCAT)
Device 2-Image
Device 2-Image-Info
Inputs
Outputs
InfoData
Drive 1
Transmit PDO 1 mapping parameters
Transmit PDO 2 mapping parameters
Receive PDO 1 mapping parameters
Receive PDO 2 mapping parameters
WcState
InfoData
Mappings
General EtherCAT DC Process Data Startup CoE - Online Online
Update List
Advanced...
Add to Startup...
☐ Auto Update ☑ Single Update ☐ Show Offline Data
Online Data
Module OD (AoE Port): 0
Name Online Type Size >Addr In/Out User Linked to
Server (Port) Timestamp Message

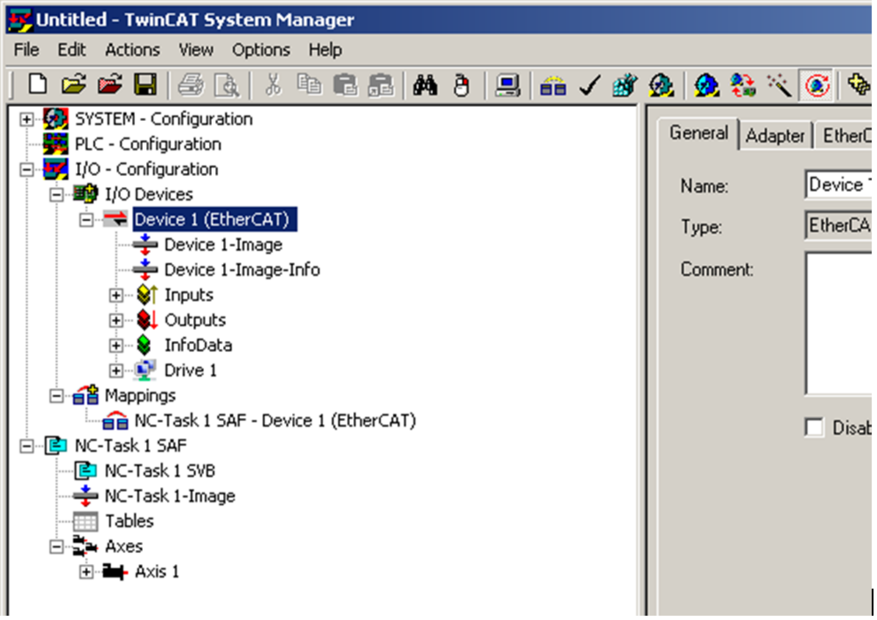
Index	Name	Flags	Value
1000	Device type	M RO	0x00060192 (393618)
1001	Error Register	M RO	0x00 (0)
1008	Device name	RO	iPOS3604
1009	Hardware version	RO	v1.0
100A	Software version	RO	F510D
1018:0	Identity	M RO	> 4 <
1018:01	Vendor ID	RO	0x000001A3 (419)
1018:02	Product code	RO	0x019C3385 (27014021)
1018:03	Revision	RO	0x35313044 (892416068)
1018:04	Serial number	RO	0x41421245 (1094849093)
1600:0	Receive PDO 1 mapping parameters	RW	> 2 <
1601:0	Receive PDO 2 mapping parameters	RW	> 1 <
1602:0	Receive PDO 3 mapping parameters	RW	> 1 <
1603:0	Receive PDO 4 mapping parameters	RW	> 1 <
1A00:0	Transmit PDO 1 mapping parameters	RW	> 1 <
1A01:0	Transmit PDO 2 mapping parameters	RW	> 1 <

1.3 Controlling the drive using CoE commands. Examples

1.3.1 Starting a position profile with TwinCAT System Manager

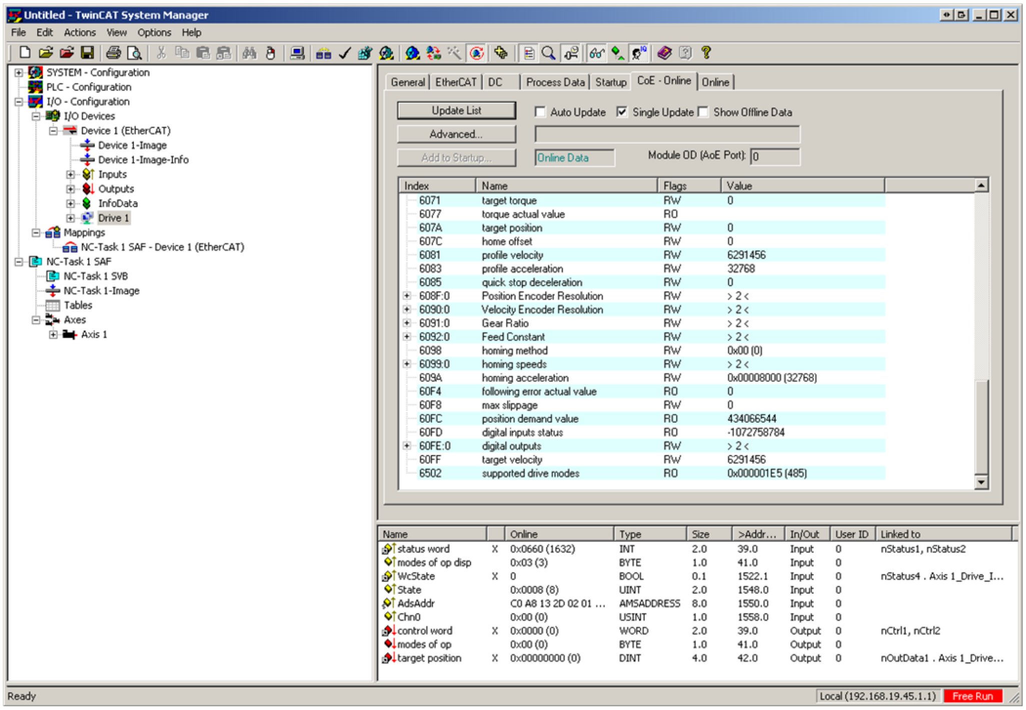
Assuming the motor has been connected to the drive and it has been set up, by following the next steps, a positive trapezoidal motion shall be executed:

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



Name: Device
Type: EtherCA
Comment:

2. Click on *CoE – Online* tab on the right side. If the provided .xml file matches the characteristics of the drive, the CoE objects shall have an associated name, like object 0x1018 Identity. Else, all CoE objects will be read directly from the drive without anything in the name column.



□ Auto Update ☑ Single Update □ Show Offline Data

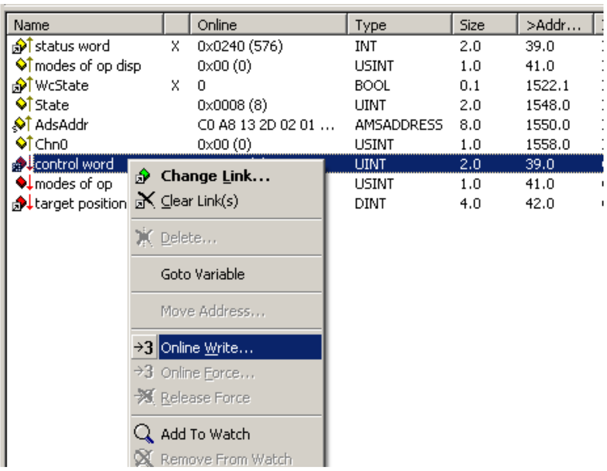
Online Data   Module OD (AoE Port): 0

Index	Name	Flags	Value
6071	target torque	RW	0
6077	torque actual value	RO	
607A	target position	RW	0
607C	home offset	RW	0
6081	profile velocity	RW	6291456
6083	profile acceleration	RW	32768
6085	quick stop deceleration	RW	0
608F:0	Position Encoder Resolution	RW	> 2 <
6090:0	Velocity Encoder Resolution	RW	> 2 <
6091:0	Gear Ratio	RW	> 2 <
6092:0	Feed Constant	RW	> 2 <
6098	homing method	RW	0x00 (0)
6099:0	homing speeds	RW	> 2 <
609A	homing acceleration	RW	0x00008000 (32768)
60F4	following error actual value	RO	0
60F8	max slippage	RW	0
60FC	position demand value	RO	434066544
60FD	digital inputs status	RO	-1072758784
60FE:0	digital outputs	RW	> 2 <
60FF	target velocity	RW	6291456
6502	supported drive modes	RO	0x000001E5 (485)

Name		Online	Type	Size	>Addr...	In/Out	User ID	Linked to
status word	X	0x0660 (1632)	INT	2.0	39.0	Input	0	nStatus1, nStatus2
modes of op disp		0x03 (3)	BYTE	1.0	41.0	Input	0	
WcState	X	0	BOOL	0.1	1522.1	Input	0	nStatus4 . Axis 1_Drive_I...
State		0x0008 (8)	UINT	2.0	1548.0	Input	0	
AdsAddr		C0 A8 13 2D 02 01 ...	AMSADDRESS	8.0	1550.0	Input	0	
Chn0		0x00 (0)	USINT	1.0	1558.0	Input	0	
control word	X	0x0000 (0)	WORD	2.0	39.0	Output	0	nCtrl1, nCtrl2
modes of op		0x00 (0)	BYTE	1.0	41.0	Output	0	
target position	X	0x00000000 (0)	DINT	4.0	42.0	Output	0	nOutData1 . Axis 1_Drive...

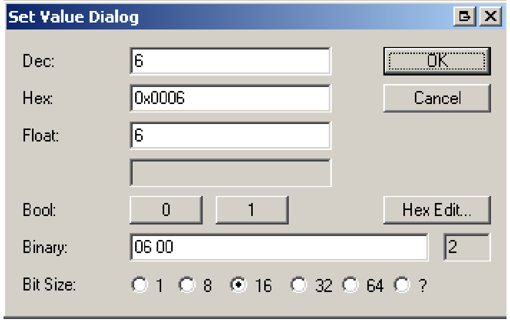
Local (192.168.19.45.1.1) Free Run

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



Name		Online	Type	Size	>Addr...
status word	X	0x0240 (576)	INT	2.0	39.0
modes of op disp		0x00 (0)	USINT	1.0	41.0
WcState	X	0	BOOL	0.1	1522.1
State		0x0008 (8)	UINT	2.0	1548.0
AdsAddr		C0 A8 13 2D 02 01 ...	AMSADDRESS	8.0	1550.0
Chn0		0x00 (0)	USINT	1.0	1558.0
control word			UINT	2.0	39.0
modes of op			USINT	1.0	41.0
target position			DINT	4.0	42.0

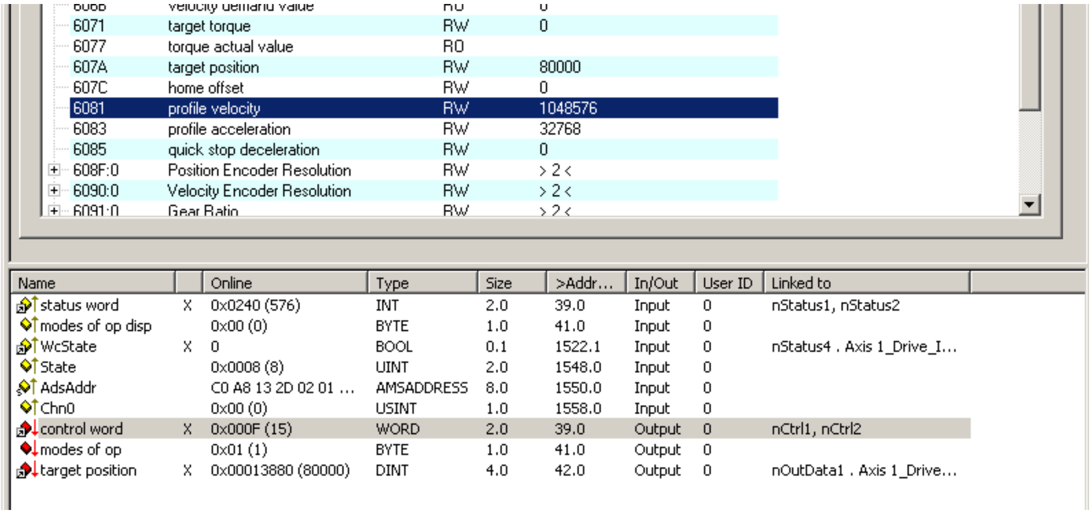
4. To enter in **Ready to switch** on state, write in the Binary field 06 00 and click OK.



Set Value Dialog

Dec:	6	OK
Hex:	0x0006	Cancel
Float:	6	
Bool:	0 | 1	Hex Edit...
Binary:	06 00	2
Bit Size:	○ 1 ○ 8 ● 16 ○ 32 ○ 64 ○ ?	

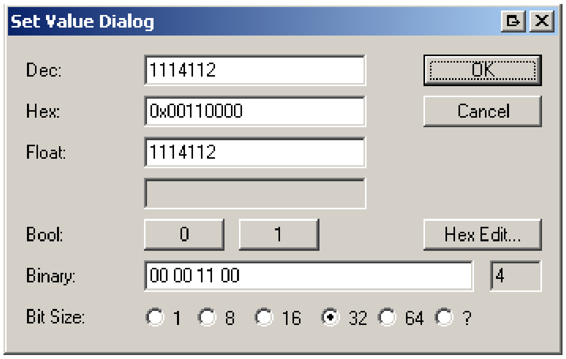
5. Repeat Step 3 and send **Switch** On (07 00) into *Control word*. After this command, the drive will apply voltage to the motor though applied torque will be zero.
6. Do the same for *Target position* and write 80000 into Dec field. The drive will finally execute 80000 internal position units (encoder counts) after the start motion command is given.
7. In the object list choose the object with index *0x6081 Profile velocity* and double click on it.



606B	velocity demand value	RO	0
6071	target torque	RW	0
6077	torque actual value	RO	
607A	target position	RW	80000
607C	home offset	RW	0
6081	profile velocity	RW	1048576
6083	profile acceleration	RW	32768
6085	quick stop deceleration	RW	0
608F:0	Position Encoder Resolution	RW	> 2 <
6090:0	Velocity Encoder Resolution	RW	> 2 <
6091:0	Gear Ratio	RW	> 2 <

Name		Online	Type	Size	>Addr...	In/Out	User ID	Linked to
status word	X	0x0240 (576)	INT	2.0	39.0	Input	0	nStatus1, nStatus2
modes of op disp		0x00 (0)	BYTE	1.0	41.0	Input	0	
WcState	X	0	BOOL	0.1	1522.1	Input	0	nStatus4 . Axis 1_Drive_I...
State		0x0008 (8)	UINT	2.0	1548.0	Input	0	
AdsAddr		C0 A8 13 2D 02 01 ...	AMSADDRESS	8.0	1550.0	Input	0	
Chn0		0x00 (0)	USINT	1.0	1558.0	Input	0	
control word	X	0x000F (15)	WORD	2.0	39.0	Output	0	nCtrl1, nCtrl2
modes of op		0x01 (1)	BYTE	1.0	41.0	Output	0	
target position	X	0x00013880 (80000)	DINT	4.0	42.0	Output	0	nOutData1 . Axis 1_Drive...

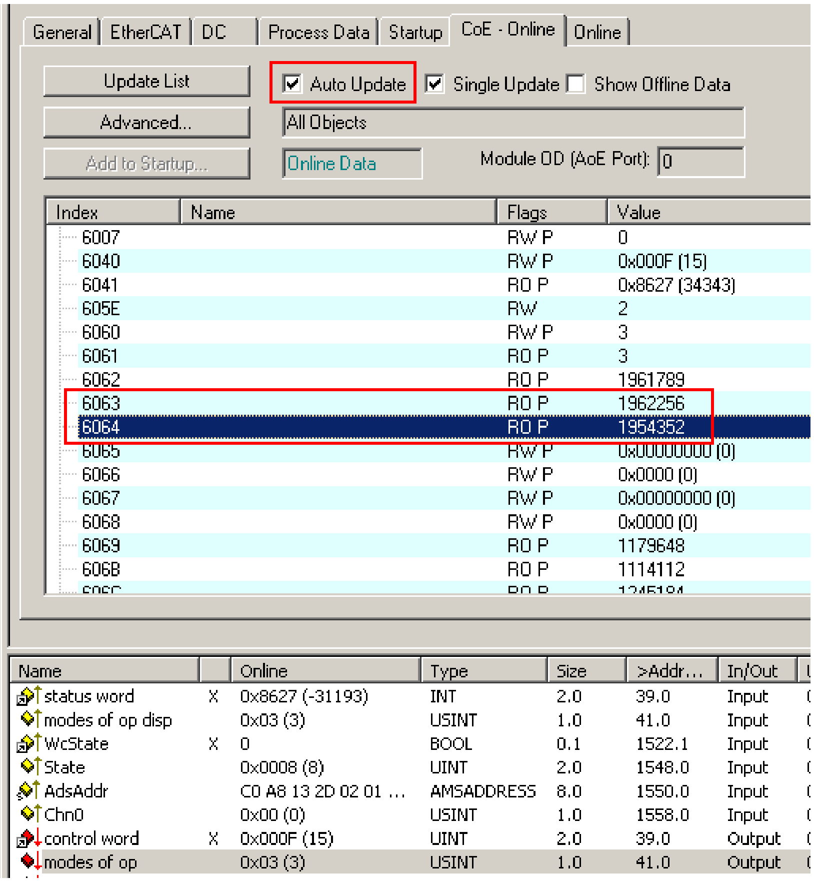
8. Write 00 00 11 00 in the binary field and click *OK*. The profile shall be executed with 11.0 IU (encoder counts)/second.



Set Value Dialog

Dec:	1114112	OK
Hex:	0x00110000	Cancel
Float:	1114112	
Bool:	0 | 1	Hex Edit...
Binary:	00 00 11 00	4
Bit Size:	○ 1 ○ 8 ○ 16 ● 32 ○ 64 ○ ?	

9. As in Step 7, Online write the value 01 into the object 6060h *Modes of operation* to enable the position profile in Modes of operation.
10. Repeat Step 3 and send **Operation Enable** (0F 00) into *Control word*. After this command, the drive will apply voltage to the motor and will keep its current position.
11. To start the motion, write in *Control word* variable 001F as in Step 3.
12. To follow the actual position of the motor, scroll to the object index *0x6064 Position actual value* and click the check box *Auto Update*. The motor actual value shall update automatically.
13. After reaching 80000 counts, the motor shall stop and hold its position until it receives new motion commands.



☑ Auto Update ☑ Single Update ☐ Show Offline Data

All Objects

Online Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
6007		RW P	0
6040		RW P	0x000F (15)
6041		RO P	0x8627 (34343)
605E		RW	2
6060		RW P	3
6061		RO P	3
6062		RO P	1961789
6063		RO P	1962256
6064		RO P	1954352
6065		RW P	0x00000000 (0)
6066		RW P	0x0000 (0)
6067		RW P	0x00000000 (0)
6068		RW P	0x0000 (0)
6069		RO P	1179648
606B		RO P	1114112
606C		RO P	1245184

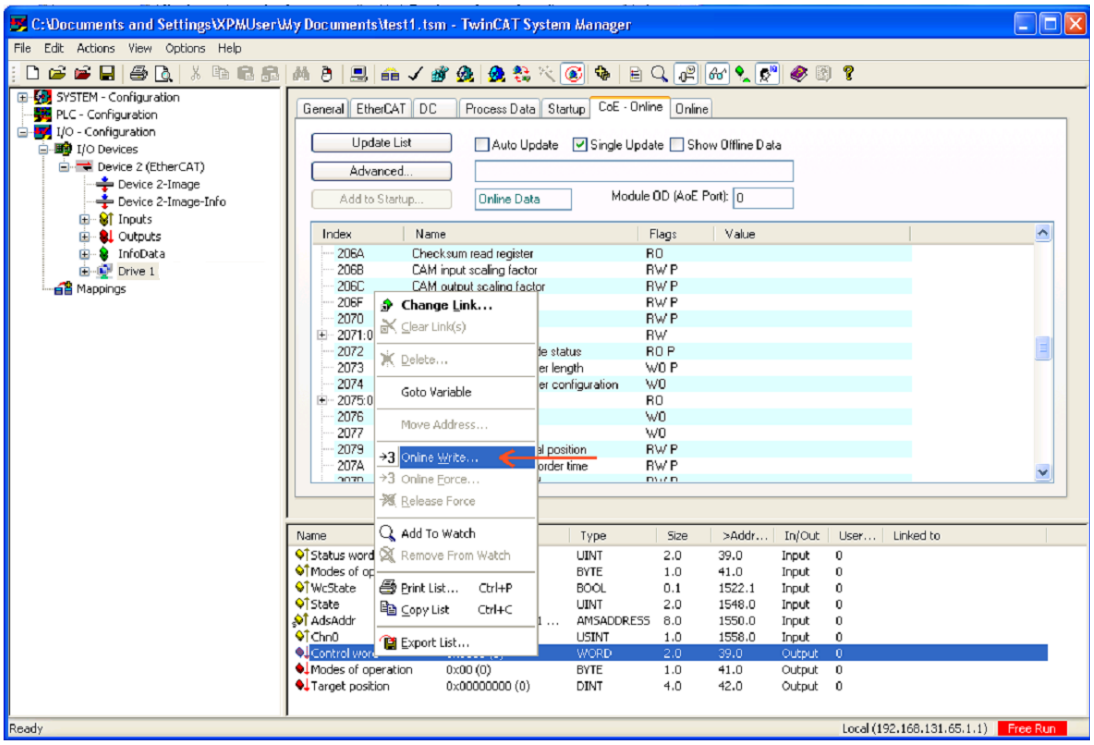
Name		Online	Type	Size	>Addr...	In/Out	I
status word	X	0x8627 (-31193)	INT	2.0	39.0	Input	(
modes of op disp		0x03 (3)	USINT	1.0	41.0	Input	(
WcState	X	0	BOOL	0.1	1522.1	Input	(
State		0x0008 (8)	UINT	2.0	1548.0	Input	(
AdsAddr		C0 A8 13 2D 02 01 ...	AMSADDRESS	8.0	1550.0	Input	(
Chn0		0x00 (0)	USINT	1.0	1558.0	Input	(
control word	X	0x000F (15)	UINT	2.0	39.0	Output	(
modes of op		0x03 (3)	USINT	1.0	41.0	Output	(

1.3.2 Starting a cyclic synchronous position mode

Remark: The cyclic synchronous position mode is the one used by the TwinCAT software in chapter 1.2.5 Running the NC-PTP interface . This example shows the manual steps that are behind this operation mode.

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

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



□ Auto Update ☑ Single Update □ Show Offline Data

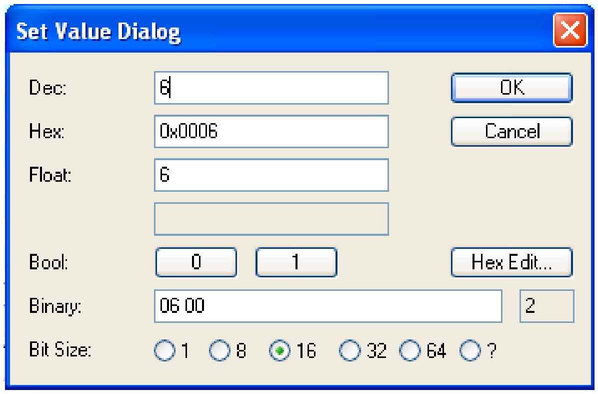
Online Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
206A	Checksum read register	RO	
206B	CAM input scaling factor	RW P	
206C	CAM output scaling factor	RW P	
206F		RW P	
2070		RW P	
2071:0		RW	
2072	...e status	RO P	
2073	...er length	WO P	
2074	...er configuration	WO	
2075:0		RO	
2076		WO	
2077		WO	
2079	...l position	RW P	
207A	...order time	RW P	

Name		Type	Size	>Addr...	In/Out	User...	Linked to
Status word		UINT	2.0	39.0	Input	0	
Modes of op...		BYTE	1.0	41.0	Input	0	
WcState		BOOL	0.1	1522.1	Input	0	
State		UINT	2.0	1548.0	Input	0	
AdsAddr	1 ...	AMSADDRESS	8.0	1550.0	Input	0	
Chn0		USINT	1.0	1558.0	Input	0	
Control wor...		WORD	2.0	39.0	Output	0	
Modes of operation	0x00 (0)	BYTE	1.0	41.0	Output	0	
Target position	0x00000000 (0)	DINT	4.0	42.0	Output	0	

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



Dec: 6

Hex: 0x0006

Float: 6

Bool: 0 1

Binary: 06 00 2

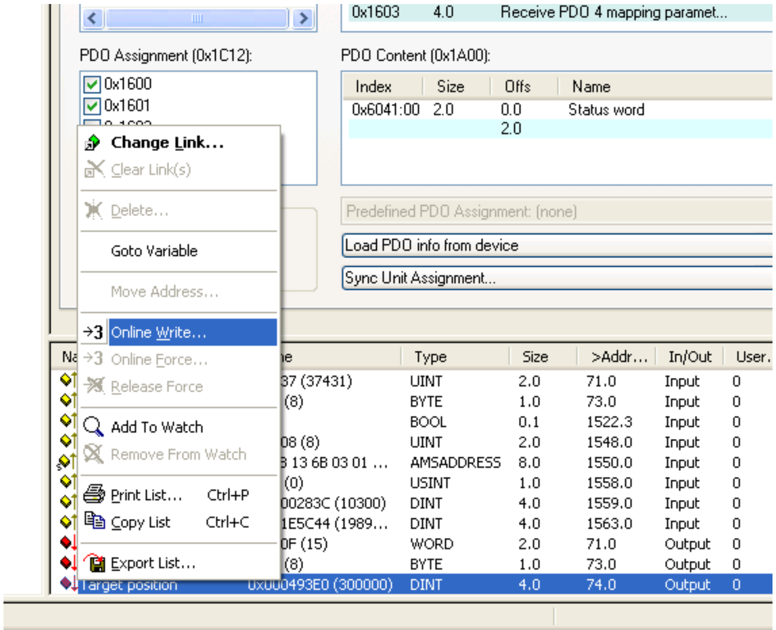
Bit Size: ○ 1 ○ 8 ● 16 ○ 32 ○ 64 ○ ?

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

Now modify Modes of operation object 6060h value 08 via SDO write. This value sets the drive in Cyclic synchronous position mode.

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

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



0x1603	4.0	Receive PDO 4 mapping paramet...

PDO Assignment (0x1C12):

☑ 0x1600
☑ 0x1601

PDO Content (0x1A00):

Index	Size	Offs	Name
0x6041:00	2.0	0.0	Status word
		2.0	

Na	...e	Type	Size	>Addr...	In/Out	User...
	37 (37431)	UINT	2.0	71.0	Input	0
	(8)	BYTE	1.0	73.0	Input	0
		BOOL	0.1	1522.3	Input	0
	08 (8)	UINT	2.0	1548.0	Input	0
	3 13 6B 03 01 ...	AMSADDRESS	8.0	1550.0	Input	0
	(0)	USINT	1.0	1558.0	Input	0
	00283C (10300)	DINT	4.0	1559.0	Input	0
	1E5C44 (1989...	DINT	4.0	1563.0	Input	0
	0F (15)	WORD	2.0	71.0	Output	0
	(8)	BYTE	1.0	73.0	Output	0
Target position	0x000493E0 (300000)	DINT	4.0	74.0	Output	0

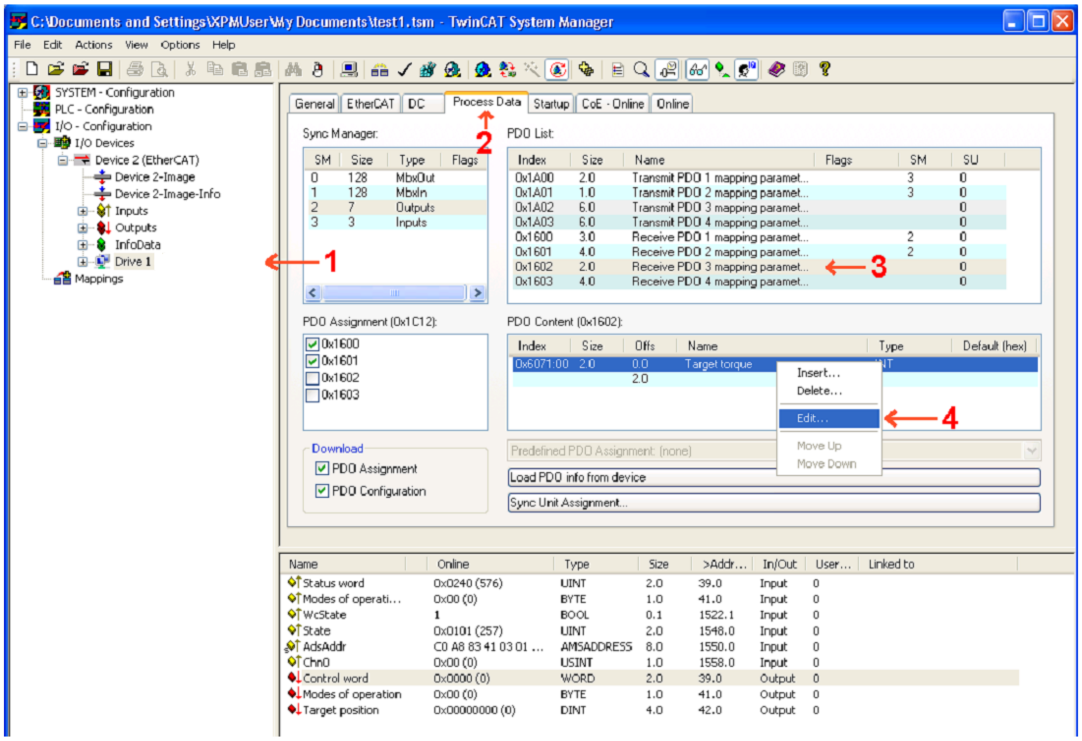
1.3.3 Mapping objects to TxPDOs and RxPDOs in TwinCAT System Manager

Set the drive in Config mode (Shift + F4). The drive can be with free run mode activated or not.

1. Select Drive 1 in the left.
2. Select Process Data tab to the upper right of the screen
3. Select object 0x1602 (RPDO3) by clicking on it.

Below, in the PDO content window, the default mapped object is shown.

4. To map another object instead of the existing one, right click on it and choose Edit.. from the menu.


C:\Documents and Settings\XPMUser\My Documents\test1.tsm - TwinCAT System Manager
File Edit Actions View Options Help
SYSTEM - Configuration
PLC - Configuration
I/O - Configuration
I/O Devices
Device 2 (EtherCAT)
Device 2-Image
Device 2-Image-Info
Inputs
Outputs
InfoData
Drive 1
Mappings
General EtherCAT DC Process Data Startup CoE - Online Online
1
2
3
4
Insert...
Delete...
Edit...
Move Up
Move Down
Predefined PDO Assignment: (none)
Load PDO info from device
Sync Unit Assignment...

Sync Manager:

SM	Size	Type	Flags
0	128	MbxOut	
1	128	MbxIn	
2	7	Outputs	
3	3	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A00	2.0	Transmit PDO 1 mapping paramet...		3	0
0x1A01	1.0	Transmit PDO 2 mapping paramet...		3	0
0x1A02	6.0	Transmit PDO 3 mapping paramet...			0
0x1A03	6.0	Transmit PDO 4 mapping paramet...			0
0x1600	3.0	Receive PDO 1 mapping paramet...		2	0
0x1601	4.0	Receive PDO 2 mapping paramet...		2	0
0x1602	2.0	Receive PDO 3 mapping paramet...			0
0x1603	4.0	Receive PDO 4 mapping paramet...			0

PDO Assignment (0x1C12):

☑ 0x1600
☑ 0x1601
□ 0x1602
□ 0x1603

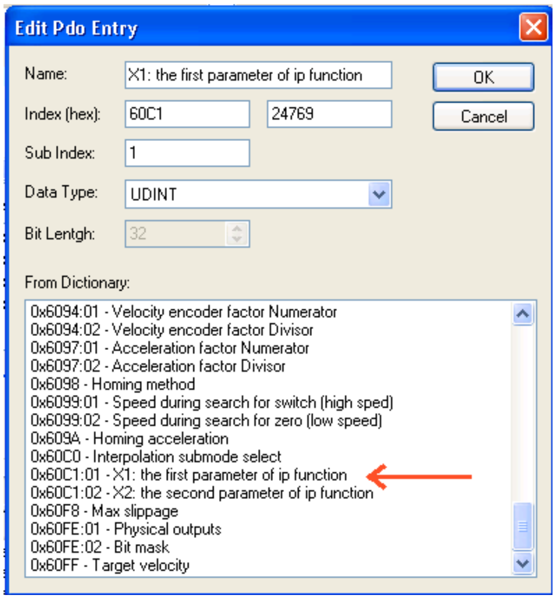
PDO Content (0x1602):

Index	Size	Offs	Name	Type	Default (hex)
0x6071:00	2.0	0.0	Target torque	INT	
		2.0			

Download
☑ PDO Assignment
☑ PDO Configuration

Name	Online	Type	Size	>Addr...	In/Out	User...	Linked to
Status word	0x0240 (576)	UINT	2.0	39.0	Input	0	
Modes of operati...	0x00 (0)	BYTE	1.0	41.0	Input	0	
WcState	1	BOOL	0.1	1522.1	Input	0	
State	0x0101 (257)	UINT	2.0	1548.0	Input	0	
AdsAddr	C0 A8 83 41 03 01 ...	AMSADDRESS	8.0	1550.0	Input	0	
Chn0	0x00 (0)	USINT	1.0	1558.0	Input	0	
Control word	0x0000 (0)	WORD	2.0	39.0	Output	0	
Modes of operation	0x00 (0)	BYTE	1.0	41.0	Output	0	
Target position	0x00000000 (0)	DINT	4.0	42.0	Output	0	

A list of PDO mappable objects appears. Choose object 60C1: sub-index 1 and click OK.



Name: X1: the first parameter of ip function
Index (hex): 60C1 24769
Sub Index: 1
Data Type: UDINT
Bit Lentgh: 32

From Dictionary:

0x6094:01 - Velocity encoder factor Numerator
0x6094:02 - Velocity encoder factor Divisor
0x6097:01 - Acceleration factor Numerator
0x6097:02 - Acceleration factor Divisor
0x6098 - Homing method
0x6099:01 - Speed during search for switch (high sped)
0x6099:02 - Speed during search for zero (low speed)
0x609A - Homing acceleration
0x60C0 - Interpolation submode select
0x60C1:01 - X1: the first parameter of ip function
0x60C1:02 - X2: the second parameter of ip function
0x60F8 - Max slippage
0x60FE:01 - Physical outputs
0x60FE:02 - Bit mask
0x60FF - Target velocity

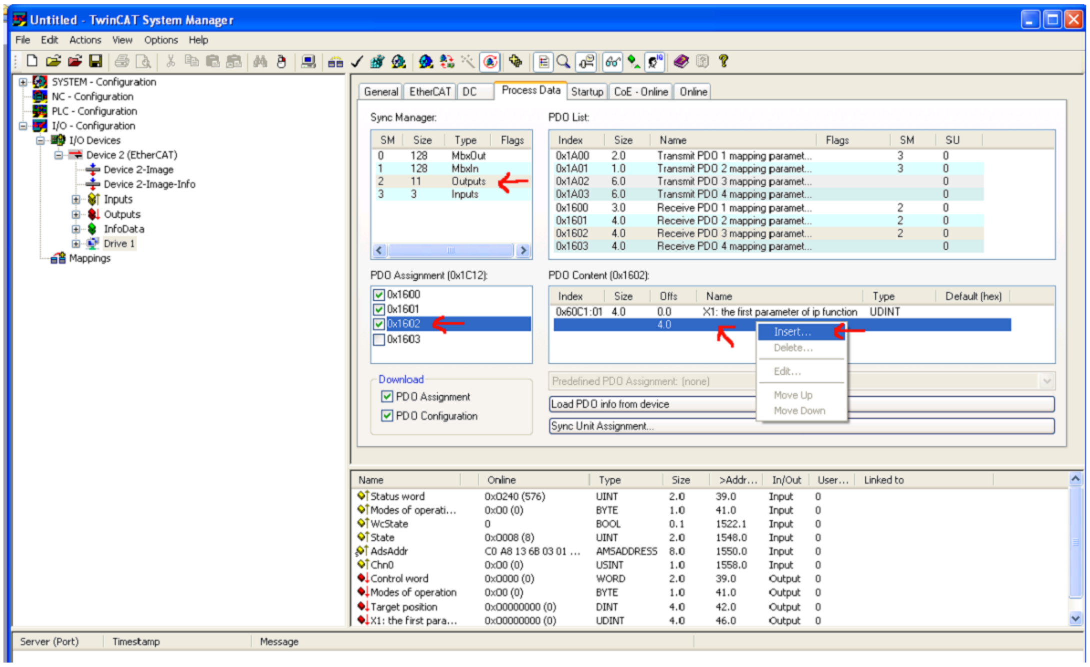
In the Sync Manager, click the Outputs to access and edit object 1C12h contents in the PDO Assignment window below. Check the box for object 1602h. This will include RxPDO3 in the sync manager.

To map a new object into RxPDO3, right click on the empty line below X1 and choose Insert…

A new window will appear like in the figure below. Choose object 60C1h sub-index 2.

Similar to the previous steps, add the object 2072h Interpolated position mode status to TxPDO2 in sub-index 1A01h using the insert object procedure.

To activate the new PDO setup, click Reload I/O devices button or press F4.



Sync Manager:

SM	Size	Type	Flags
0	128	MbxOut	
1	128	MbxIn	
2	11	Outputs	
3	3	Inputs	

PDO List:

Index	Size	Name	Flags	SM	SU
0x1A00	2.0	Transmit PDO 1 mapping paramet...		3	0
0x1A01	1.0	Transmit PDO 2 mapping paramet...		3	0
0x1A02	6.0	Transmit PDO 3 mapping paramet...			0
0x1A03	6.0	Transmit PDO 4 mapping paramet...			0
0x1600	3.0	Receive PDO 1 mapping paramet...		2	0
0x1601	4.0	Receive PDO 2 mapping paramet...		2	0
0x1602	4.0	Receive PDO 3 mapping paramet...		2	0
0x1603	4.0	Receive PDO 4 mapping paramet...			0

PDO Assignment (0x1C12):

☑ 0x1600
☑ 0x1601
☑ 0x1602
☐ 0x1603

PDO Content (0x1602):

Index	Size	Offs	Name	Type	Default (hex)
0x60C1:01	4.0	0.0	X1: the first parameter of ip function	UDINT	
		4.0			

Download
☑ PDO Assignment
☑ PDO Configuration

Predefined PDO Assignment: (none)

Name	Online	Type	Size	>Addr...	In/Out	User...	Linked to
Status word	0x0240 (576)	UINT	2.0	39.0	Input	0	
Modes of operati...	0x00 (0)	BYTE	1.0	41.0	Input	0	
WcState	0	BOOL	0.1	1522.1	Input	0	
State	0x0008 (8)	UINT	2.0	1548.0	Input	0	
AdsAddr	C0 A8 13 6B 03 01 ...	AMSADDRESS	8.0	1550.0	Input	0	
Chn0	0x00 (0)	USINT	1.0	1558.0	Input	0	
Control word	0x0000 (0)	WORD	2.0	39.0	Output	0	
Modes of operation	0x00 (0)	BYTE	1.0	41.0	Output	0	
Target position	0x00000000 (0)	DINT	4.0	42.0	Output	0	
X1: the first para...	0x00000000 (0)	UDINT	4.0	46.0	Output	0	

1.3.4 Running an Interpolated Position Time motion in TwinCAT

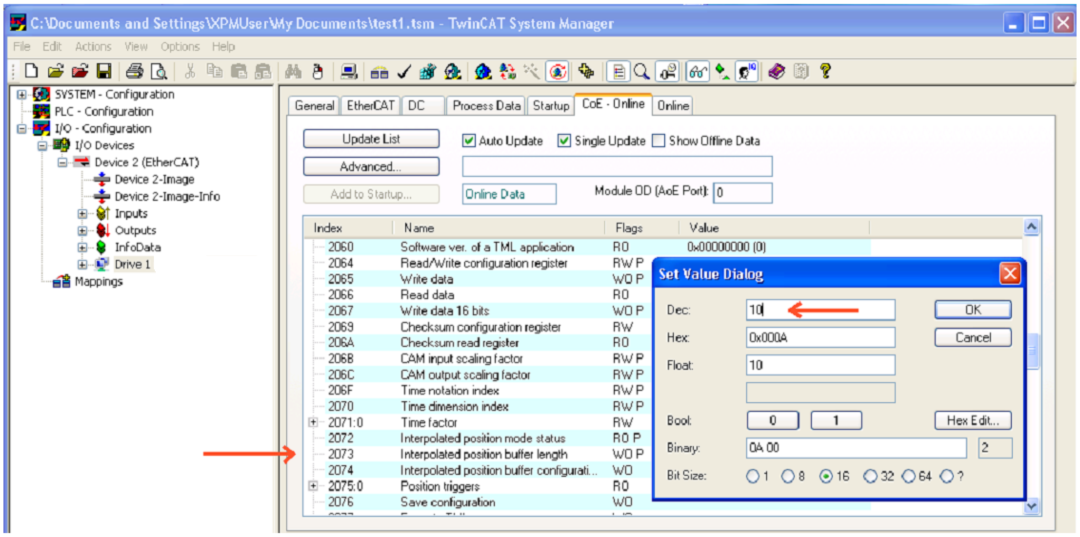
First, map variables X1 and X2 to RxPDO3. To do this, read the Mapping objects to TxPDOs and RxPDOs in TwinCAT System Manager chapter.

Write in Control Word the values 06, 07 and 0x000F to bring the drive to Operation enabled state.

To modify the RxPDO values or writing into an object using SDOs, first read the

Starting a cyclic synchronous position mode.

First, set in object 2073h a lager buffer length. Set it to 10. The default is 7 points, and the maximum is 15.



☑ Auto Update ☑ Single Update □ Show Offline Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
2060	Software ver. of a TML application	RO	0x00000000 (0)
2064	Read/Write configuration register	RW P	
2065	Write data	WO P	
2066	Read data	RO	
2067	Write data 16 bits	WO P	
2069	Checksum configuration register	RW	
206A	Checksum read register	RO	
206B	CAM input scaling factor	RW P	
206C	CAM output scaling factor	RW P	
206F	Time notation index	RW P	
2070	Time dimension index	RW P	
2071:0	Time factor	RW	
2072	Interpolated position mode status	RO P	
2073	Interpolated position buffer length	WO P	
2074	Interpolated position buffer configurati...	WO	
2075:0	Position triggers	RO	
2076	Save configuration	WO	

Dec: 10

Hex: 0x000A

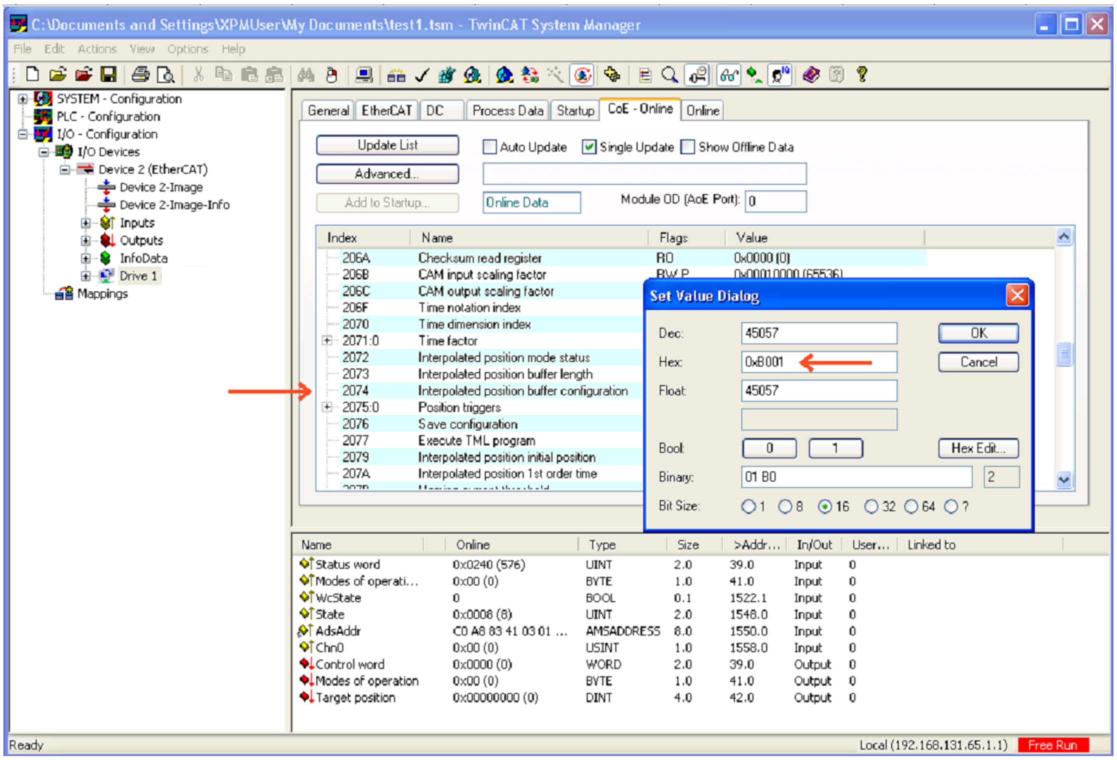
Float: 10

Bool: 0 1

Binary: 0A 00 2

Bit Size: ○ 1 ○ 8 ● 16 ○ 32 ○ 64 ○ ?

After setting another buffer size, or before sending new interpolation points, the buffer must be cleared. Send in object 2074h 0xB001. This will clear the buffer and reinitialize the integrity counter to 1.



☐ Auto Update ☑ Single Update ☐ Show Offline Data

Online Data

Module OD (AoE Port): 0

Index	Name	Flags	Value
206A	Checksum read register	RO	0x0000 (0)
206B	CAM input scaling factor	RW P	0x00010000 (65536)
206C	CAM output scaling factor		
206F	Time notation index		
2070	Time dimension index		
2071:0	Time factor		
2072	Interpolated position mode status		
2073	Interpolated position buffer length		
2074	Interpolated position buffer configuration		
2075:0	Position triggers		
2076	Save configuration		
2077	Execute TML program		
2079	Interpolated position initial position		
207A	Interpolated position 1st order time		

Set Value Dialog

Dec: 45057

Hex: 0xB001

Float: 45057

Bool: 0 1

Binary: 01 B0 2

Bit Size: ○ 1 ○ 8 ● 16 ○ 32 ○ 64 ○ ?

OK Cancel Hex Edit...

Name	Online	Type	Size	>Addr...	In/Out	User...	Linked to
Status word	0x0240 (576)	UINT	2.0	39.0	Input	0	
Modes of operati...	0x00 (0)	BYTE	1.0	41.0	Input	0	
WcState	0	BOOL	0.1	1522.1	Input	0	
State	0x0008 (8)	UINT	2.0	1548.0	Input	0	
AdsAddr	C0 A8 83 41 03 01 ...	AMSADDRESS	8.0	1550.0	Input	0	
Chn0	0x00 (0)	USINT	1.0	1558.0	Input	0	
Control word	0x0000 (0)	WORD	2.0	39.0	Output	0	
Modes of operation	0x00 (0)	BYTE	1.0	41.0	Output	0	
Target position	0x00000000 (0)	DINT	4.0	42.0	Output	0	

Local (192.168.131.65.1.1) Free Run

Note: the integrity counter in the ElectroCraft EtherCAT® drives is always on.

If the drive receives an interpolated point with the same integrity counter, it will just be ignored with no warning. If it receives a lower or a +2 higher integrity counter, it will send an emcy message with error code FF01h and the object 2072h (Buffer status) mapped on bytes 4 and 5. The only acceptable message is to send an integrity counter incremented by +1 than the previous number found in interpolated status, object 2072h.

An interpolated point is composed from Object 60C1h both subindexes. So if the data in these both subindexes is changed every sync cycle, the drive can receive an interpolated point every sync cycle.

In 60C1 index 1, the target position is sent, and in index2, the interpolated time is sent (which is optional if object 207Ah is used) and the integrity counter set in the last 7 bits of the 32 bit variable.

The interpolated time can be sent also in object 207Ah (interpolated time 1st order position). By sending a time in this object, the time data inside object 60C1 subindex 2 will be ignored.

So it is possible to set a time of 2 ms by sending 2 in 207Ah and in object 60C1 just send the target position and the integrity counter.

First send 0xFA0 to object 207Ah to set a fixed 4000ms time interval between interpolation points.

Send IP points:

1. In X1 send 4000 target position, in X2 send 0x02000000. The integrity counter is written in byte 3 of X2 on the last 7 bits. Meaning any number written in byte3 must be shifted to the left by 1 bit. So 1 is when byte3 = 2; 2 is when byte3=4; 3 is when byte3=6 …. 30 is when byte3=0x3C (60d) and so on.

   So in point1, the drive will do 4000 encoder tics in 4 seconds and the integrity counter is 1.
2. In X1 send 12000 and in X2 send 0x04000000 (integrity counter =2)

3. In X1 send 32000 and in X2 send 0x06000000 (integrity counter =3)
4. In X1 send 64000 and in X2 send 0x08000000 (integrity counter =4)
5. In X1 send 20000 and in X2 send 0x0A000000 (integrity counter =5)
6. In X1 send 4000 and in X2 send 0x0C000000 (integrity counter =6)

Send in Modes of operation variable mapped in a RPDO the value 07 (interpolated mode)

Note: before sending any IP points, the absolute or relative bit must be already set in Control Word. Also object 2079h (Interpolated position initial position) must be set with the correct position before loading the points and starting a trapezoidal position profile).

Send 0x001F in controlword to start an absolute motion with the loaded PT points.

The motor will start moving, travel until position 64000 encoder counts and return to position 4000.

In the recently mapped Interpolated position mode status variable, the status of the interpolated mode can be monitored. So in this variable, you can see if the buffer is low, empty, the current integrity counter value and if there was an integrity counter error.

The integrity counter is 7 bits long, so it can have a value up to 127. So when the integrity counter reaches 127 (0x7F), the next logical value is 0x00.

In case the interpolated time is equal to the EtherCAT® PDO cycle time, there is no need to monitor the buffer. Just send points with a correct integrity counter every cycle, taking care of the position value.

2 CAN application protocol over EtherCAT®

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

2.1 EtherCAT® Architecture

EtherCAT® fieldbus accepts different network topologies, the most commonly used being presented in ***Figure 2.1***.

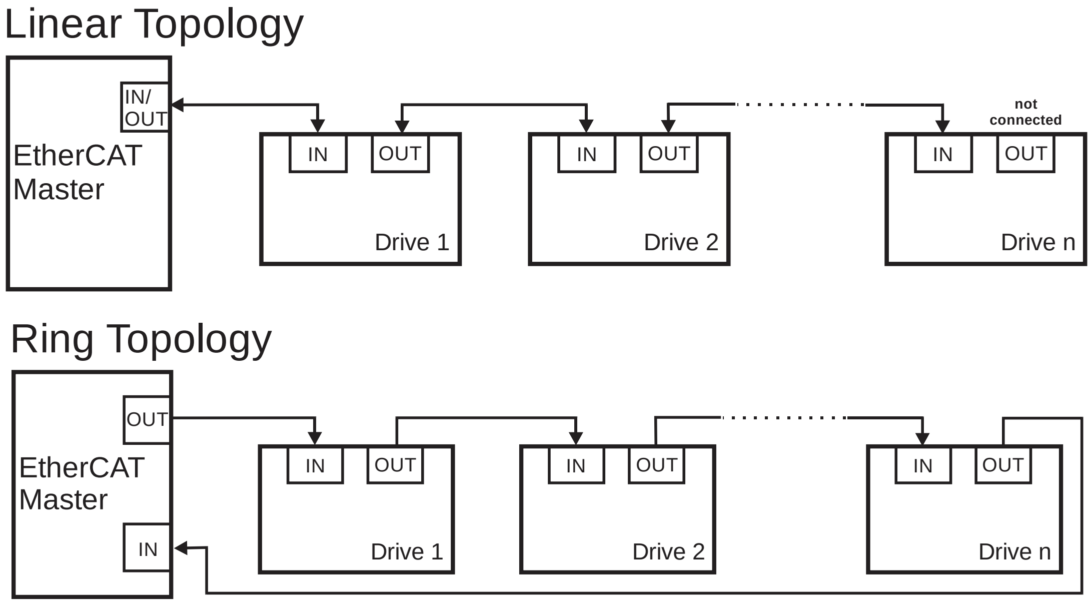


Figure 2.1 EtherCAT® Architecture

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

2.2 Accessing EtherCAT® devices

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

2.2.1 CoE elements

In table 2.0.1 are described the Mailbox Header and CoE Header.

Table 2.0.1 *CoE elements*

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

2.2.2 Object dictionary

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

2.2.3 Object access using index and sub-index

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

2.2.4 Service Data Objects (SDO)

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

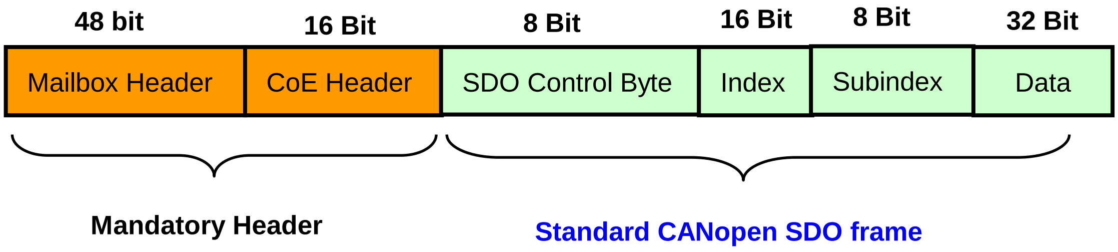


Figure 2.2 EtherCAT® message SDO structure

Standard CANopen SDO frames can be used:

– Initiate SDO Download

– Download SDO Segment

– Initiate SDO Upload

– Upload SDO Segment

– Abort SDO Transfer

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

SDO transfers are confirmed services. In case of an error, an Abort SDO message is transmitted with one of the codes listed in ***Table 2.1*****.**

Table 2.1 *SDO Abort Codes*

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

2.2.5 Process Data Objects (PDO)

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

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

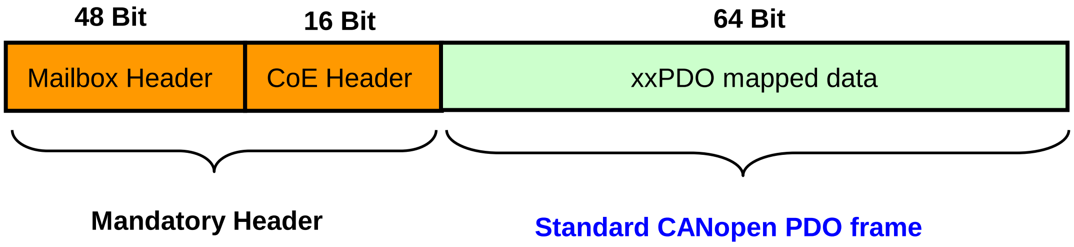


Figure 2.3 EtherCAT® message PDO structure

2.3 Objects that define SDOs and PDOs

2.3.1 Object 1600h: 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 drive must be in Pre-Op state and 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.

Also, to be able to use the PDO data, the active PDOs must be mapped in the Sync Manager objects $1C12_h$ and $1C13_h$. Read more in the provided example

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.

The entire length of a TPDO or RPDO must not exceed 64 bits. This means that the sum of the mapped objects lengths of a PDO must not exceed 64 bits.

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: deactivated 1 – 8: activated
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	02_h
Description	2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	60600008_h – modes of operation

2.3.2 Object 1602h: 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.

Object description:

Index	1602h
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: deactivated 1 – 64: activated
Default value	1

Sub-index	012_h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	$607A00020_h$ – Target Position

2.3.3 Object 1602h: Receive PDO3 Mapping Parameters

This object contains the mapping parameters of the receive PDO3. 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.

Object description:

Index	1602_h
Name	RPDO3 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: deactivated 1 – 64: activated
Default value	1

Sub-index	01_h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	607100210_h – Target torque

2.3.4 Object 1603h: Receive PDO4 Mapping Parameters

This object contains the mapping parameters of the receive PDO4. 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.

Object description:

Index	1603_h
Name	RPDO4 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: deactivated 1 – 64: activated
Default value	1

Sub-index	01_h
Description	1st mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	$60FF0020_h$ – target speed

2.3.5 Object 1A00h: 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)

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: deactivated 1 – 64: activated
Default value	1

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

2.3.6 Object 1A01h: 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)

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: deactivated 1 – 64: activated
Default value	1

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

2.3.7 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.

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: deactivated 1 – 64: activated
Default value	0

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

2.3.8 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.

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: deactivated 1 – 64: activated
Default value	0

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

2.3.9 Object 1C00h: Sync Manager Communication type

Object description:

Index	$1C00_h$
Name	Sync Manager Com. type
Object code	ARRAY
Data type	UNSIGNED8

Entry description:

Sub-index	00_h
Description	Number of sub-indexes
Access	RO
PDO mapping	No
Default value	5

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

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

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

	slave)
Default value	3

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

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

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

Object description:

Index	$1C12_h$
Name	Sync Manager Channel 2
Object code	ARRAY
Data type	UNSIGNED8

Entry description:

Sub-index	00_h
Description	Number of sub-indexes
Access	RO
PDO mapping	No
Value Range	UNSIGNED8 1-4
Default value	2

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

Sub-index	02_h
Description	PDO Mapping object index of assigned RxPDO : 2nd mapped object
Access	RW
PDO mapping	No
Value range	UNSIGNED16 0x1600 (RxPDO1) 0x1601 (RxPDO2) 0x1602 (RxPDO3) 0x1603 (RxPDO4)
Default value	0x1601 (RxPDO2)

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

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

Object description:

Index	$1C13_h$
Name	Sync Manager Channel 3
Object code	ARRAY
Data type	UNSIGNED8

Entry description:

Sub-index	00_h
Description	Number of sub-indexes
Access	RO
PDO mapping	No
Value Range	UNSIGNED8 0-4

Default value	2

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

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

2.3.12 Object 207Dh: Dummy

This object may be used to fill a RxPDO or TxPDO up to a length matching the CoE master requirements.

Object description:

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

Entry description:

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

2.4 PDOs mapping example

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

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

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

1. **Set the drive to Pre-Operational**.

2. **Disable the PDO**. 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.

3. **Map the new objects**.

a. Write in object 1602_h sub-index 1 the description of the Control Word:

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

Send the following message: SDO access to object 1602_h sub-index 1, 32-bit value 60400010_h.

b. Write in object 1602_h sub-index 2 the description of the Modes of Operation:

Index	Sub-index	Length	
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.

4. **Enable the PDO**. Set the object 1602_h sub-index 0 with the value 2.

Send the following message: SDO access to object 1602_h sub-index 0, 8-bit value 2.

5. **Add the new TPDO to the Sync Manager**:

   - Write zero in object 1C12$_h$ sub-index 0, this will disable the Sync. Manager.

   Send the following message: SDO access to object 1C12$_h$ sub-index 0, 8-bit value 00$_h$.

   -Write in object 1C12$_h$ sub-index 3 the RPDO3 mapping parameter object number:

   Send the following message: SDO access to object 1C12$_h$ sub-index 3, 16-bit value 1602$_h$.

   -Write 03$_h$ in object 1C12$_h$ sub-index 0, this will enable the Sync. Manager.

   Send the following message: SDO access to object 1C12$_h$ sub-index 0, 8-bit value 03$_h$.

Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new PDO configuration. Press F4 to reload the IO devices and enter in Operation state.

6. **Set the drive to Safe-Operational.**

7. **Set the drive to Operational.**

3 EtherCAT® State Machine (ESM)

3.1 Overview

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

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

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

No process data communication is possible in this state.

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

The ‘**Operational**’ state can be entered if the settings of the output buffer have been done and actual outputs have been delivered to the slave (provides outputs of the slave will be used). The application of the slave shall deliver actual input data and the application of the master will provide output data.

The ESM defines four states, which shall be supported:

- Init,
- Pre-Operational,
- Safe-Operational, and
- Operational.

3.2 Device control

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

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

ESM is specified in Figure 3.1.

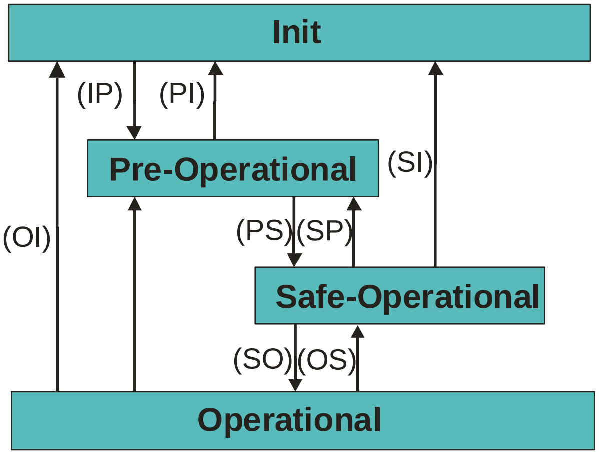


Figure 3.1 **EtherCAT® State Machine Diagram**

The local management services are related to the transitions in the ESM, as specified in Table 3.0.1. If there is more than one service related to the transition, the slave's application will process all of the related services.

Table 3.0.1 *State transitions and local management services*

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

***Table 3.0.2 AL** Control Description*

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

3.3 EtherCAT® State Machine and CANopen State Machine

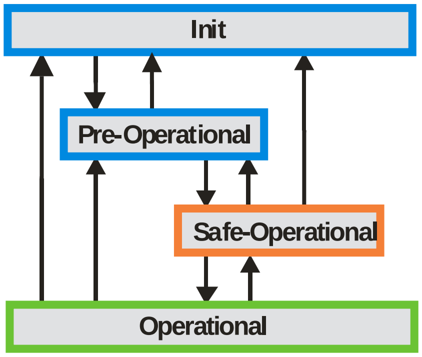


***Figure 3.2* EtherCAT® State Machine**

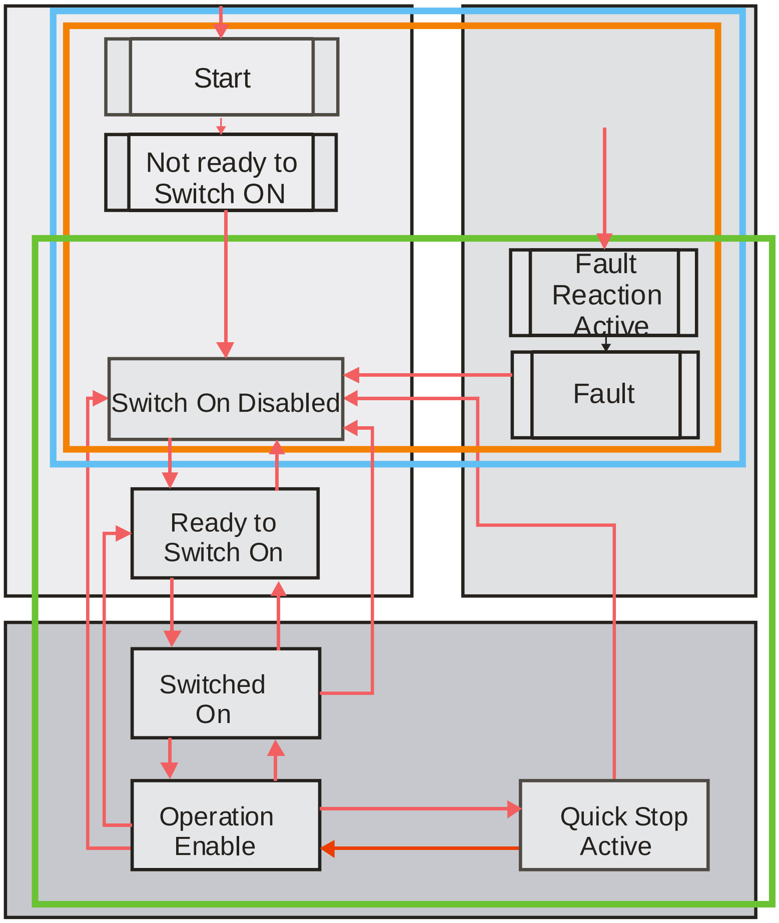


Figure 3.3 **Drive State-machine based on CANopen (DS402)**

3.4 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 ElectroCraft drives are listed in **Table 3.1**. Details regarding the conditions that may generate emergency messages are presented at object Motion Error Register index 2000_h.

Table 3.1 *Emergency Error Codes*

Error code (hex)	Description
00xx	Error Reset or No Error
10xx	Generic Error
2310	Continuous over-current
2340	Short-circuit

30xx	Voltage
3210	DC-link over-voltage
3220	DC-link under-voltage
33xx	Output Voltage
4280	Over temperature motor
4310	Over temperature drive
5441	Drive disabled due to enable input
5442	Negative limit switch active
5443	Positive limit switch active
6100	Invalid setup data
7500	Serial communication error
8100	EtherCAT® communication error
8210	PDO not processed due to length error
8220	PDO length exceeded
8331	I2t protection triggered
8580	Position wraparound / or hall sensor error
8611	Control error / Following error
9000	Command error
A0xx	ESM Transition Error
F0xx	Additional Functions
FF01	Generic interpolated position mode error (PVT / PT error)
FF02	Change set acknowledge bit wrong value
FF03	Specified homing method not available
FF04	A wrong mode is set in object 6060h, 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

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

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

3.5 EtherCAT® Synchronization

3.5.1 Overview

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

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

3.5.2 Synchronization signals

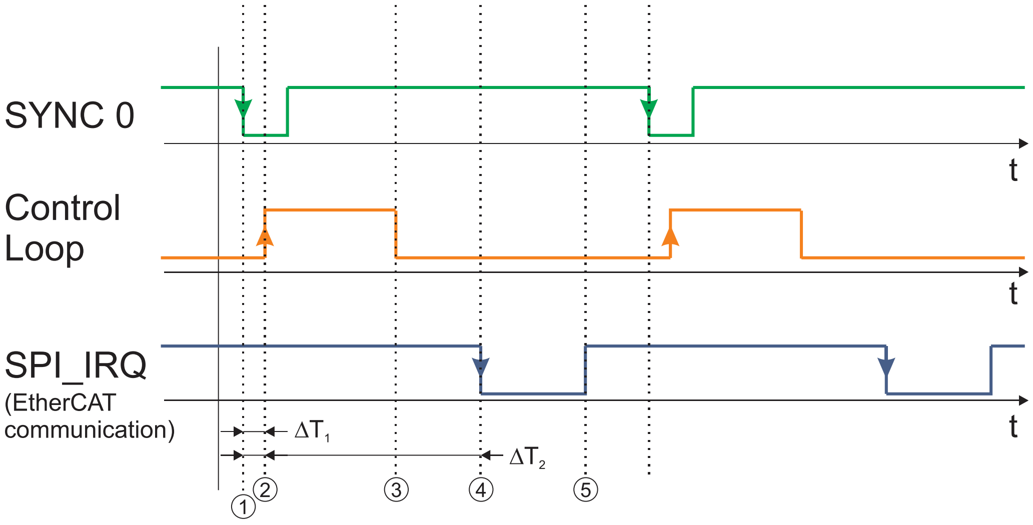


Figure 3.4 Synchronization signal and control loop timing

Time moments description:

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

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

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

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

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

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

- communication cycle shall be a multiple of the SYNC0 cycle
- SYNC0 shall be a multiple of control loop
- time moment 3 shall be before time moment 4
- time moment 5 shall be before time moment 1 of the next cycle

Remarks:

- *the minimum jitter between SYNC0 and the control loop is obtained when both have the same period*
- *it is possible to have different values for control loop, SYNC0 and communication cycle as long as conditions a) and b) are respected. Example: control loop at 1ms; SYNC0 at 2 ms and communication cycle at 4ms*

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

Important:

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

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

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

In order to preserve the synchronization when changing ΔT_1 and ΔT_2, conditions c) and d) shall be checked with an oscilloscope to make sure that a safe margin exists considering the worst case jitter of the control loop and the EtherCAT communication processing time.

3.5.3 Object 2089h: Synchronization test config

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

Object description:

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

Entry description:

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

Table 3.2 *Bit Assignment in Synchronization test config*

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

Remarks:

- Before activating this feature, disconnect any other device connected to the 3 outputs;
- Ready and Error outputs are also connected to the green and red LEDs. These will flicker when this feature is activated. This case shall not be treated as an error condition!
- SYNC0 signal on OUT0 is generated by the drive starting from SYNC0 signal generated by the EtherCAT ASIC. OUT0 goes down almost simultaneously with the SYNC0 generated by the ASIC, but it rises much faster. So SYNC0 from OUT0 will be shorter than the real signal

4 Drive control and status

4.1 Overview

The state machine from **Device Profile Drives and Motion Control** describes the drive status and the possible control sequences of the drive. The drive states can be changed by the **Control Word** and/or according to internal events. The drive current state is reflected in the **Status Word**. The state machine presented in **Figure 4.** describes the state machine of the drive with respect to the control of the power stage as a result of user commands and internal drive faults.

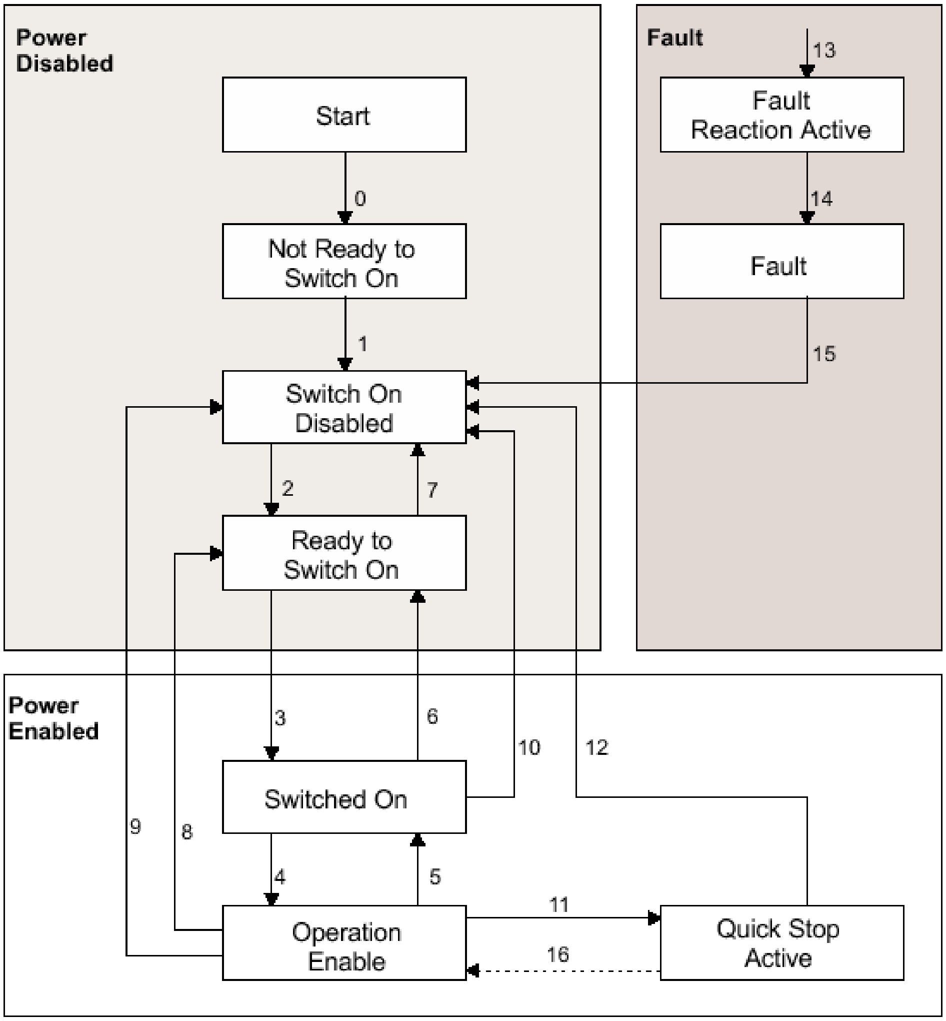


Figure 4.1 *Drive's status machine. States and transitions*

Table 4.1 Finite State Automaton 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, MPL 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, motion functions cannot be carried out yet.
Switched On (Operation Disabled)	The motor supply voltage must be switched on. The power stage is switched on (enabled). The applied torque will be zero. The motion functions cannot be carried out yet.
Operation Enable	No fault present, power stage is switched on, motion functions are enabled. This corresponds to the normal operation of the drive.
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 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

If in the *quick stop active* state the quick stop option code is set to 5, 6, 7 or 8, the drive shall not leave this state, but it may transit to the *operation enabled* state with the *Operation enable* command.

Table 4.2 **Drive State Transitions**

Transition	**Event**	**Action**
0	Automatic transition after power-on or reset application	Drive device self-test and/or self initialization shall be performed.
1	Automatic transition	Communication will be activated.
2	Bit 1 *Disable Voltage* and Bit 2 *Quick Stop,* are set in Control Word (*Shutdown* command). Motor voltage may be present.	None
3	Bit 0 is also set in Control Word (*Switch On* command)	High-level power is switched on (enabled), provided that the enable input is on enable status. Keep the motor at zero torque.
4	Bit 3 is also set Control Word (*Enable Operation* command)	Motion function is enabled, depending on the mode that is set. Also all the internal set-points are cleared.
5	Bit 3 is cancelled in Control Word (*Disable Operation* command)	Motion function is inhibited. Drive is stopped, using the acceleration rate set for position or speed profiles. Depending on the mode of operation set before the Disable Operation command, the motor may be held at the present position, kept at zero speed or zero torque
6	Bit 0 is cancelled in Control Word (*Shutdown* command)	Power stage is disabled. Drive has no torque. Motor is free to rotate
7	Bit 1 or 2 is cancelled in Control Word (*Quick Stop* or *Disable Voltage* command)	None
8	Bit 0 is cancelled in Control Word (*Shutdown* command)	Power stage is disabled. Drive has no torque. Motor is free to rotate
9	Bit 1 is cancelled in Control Word (*Disable Voltage* command)	Power stage is disabled. Drive has no torque. Motor is free to rotate
10	Bit 1 or 2 is cancelled in Control Word (*Quick Stop* or *Disable Voltage* command)	Power stage is disabled. Drive has no torque. Motor is free to rotate
11	Bit 2 is cancelled in Control Word (*Quick Stop* command)	Drive is stopped with the quick-stop deceleration rate. The power stage remains enabled. Depending on the mode of operation set before the Quick Stop command, the motor may be held at the present position, kept at zero speed or zero torque.
12	Automatic transition when the	Output stage is disabled. Drive has no torque.

	quick stop function is completed and quick stop option code is 1, 2, 3 or 4, or disable voltage command received from control device (depends on the quick stop option code)	
13	A fault has occurred	Execute specific fault treatment routine
14	The fault treatment is complete (Automatic transition)	The drive function is disabled and the high-level power is switched off
15	Bit 7 is set in Control Word (*Reset Fault* command)	Some of the bits from 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* the bit *Fault Reset* of the *controlword* has to be cleared by the host.
16	Bit 2 is set in Control Word (*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.

- If a state transition is requested, the related actions shall be processed completely before transitioning to the new state. Example: In *operation enabled* state, when the *disable operation* command is received, the drive device shall stay in the *operation enabled* state until the disable operation function (see object 605Ch) is completed.
- Drive devices able to control the contactor for the mains may switch the high-level power. If the high-level power is switched-off, the motor shall be free to rotate if not braked.
- Drive function is disabled implies no energy shall be supplied to the motor. Target or set-point values (e.g. torque, velocity, position) shall be not processed.
- Drive function is enabled implies that energy may be supplied to the motor. Target or set-point values will be processed.
- If a fault is detected in the drive device, there shall be a transition to the *fault reaction active* state. In this state, the drive will execute a special fault reaction. After the execution of this fault reaction, the drive device shall switch automatically to the *fault* state. This state shall only be left by the fault reset command, but only if the fault is not active any more.
- In case of fatal error, the drive device is no longer able to control the motor, so that an immediate switch-off of the drive device is necessary.
- The behavior of drive disabling, quick stop, halt, and fault reaction functions is configurable by means of configuration objects.

4.2 Drive control and status objects

4.2.1 Object 6040h: Control Word

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 Control Word.

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

The Control Word has the following bit assignment:

Table 4.3 *Bit Assignment in Control Word*

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

Table 4.4 Command coding

Command	Bits of the *controlword*					Transitions
	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
Switch on + enable operation	0	1	1	1	1	3 + 4
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	15

Bits 9, 6, 5, and 4 of the controlword are operation mode specific. The halt function (bit 8) behavior is operation mode specific. If the bit is 1, the commanded motion shall be interrupted, the drive shall behave as defined in the halt option code. After releasing the halt function, the commanded motion shall be continued if possible.

The bit 10 is reserved for further use; it shall be set to 0. The bits 11, 12, 13, 14, and 15 are manufacturer-specific.

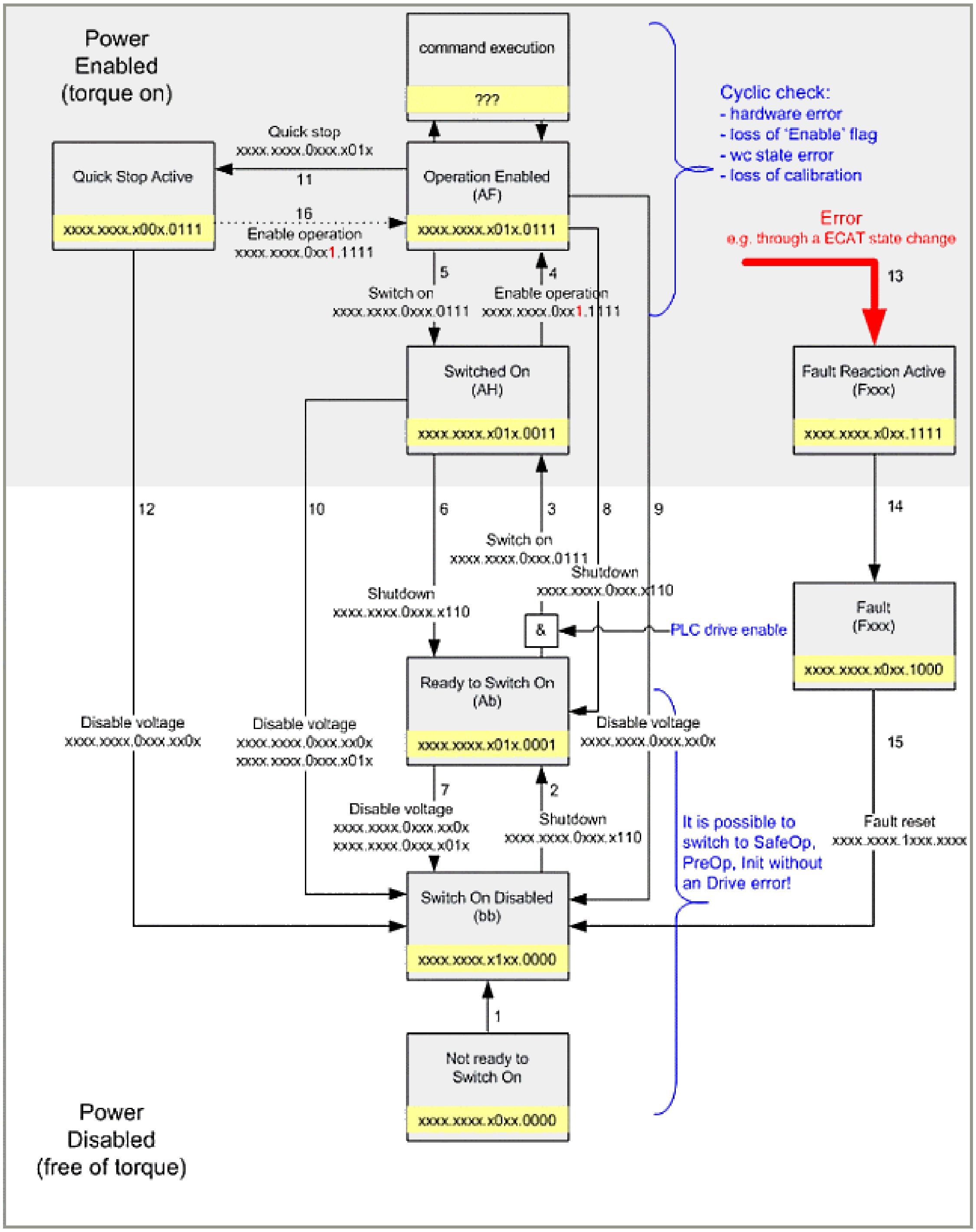


***Figure 4.2** Drive's status machine. States and transitions command coding*

4.2.2 Object 6041h: Status Word

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 Status Word has the following bit assignment:

Table 4.5 *Bit Assignment in Status Word*

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
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 MPL function or homing is executed. The execution of the last called MPL function or homing is completed.
	1	A MPL function or homing is executed. Until the function or homing execution ends or is aborted, no other MPL function / homing may be called
7	0	No Warning
	1	Warning. A MPL function / homing was called, while another MPL 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
	1	Motor supply voltage is present
3		Fault. If set, a fault condition is or was present in the drive.
2		Operation Enabled
1	0	Axis off. Power stage is disabled. Motor control is not performed
0	1	Axis on. Power stage is enabled. Motor control is performed

Table 4.6 State coding

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

If bit 4 (voltage enabled) of the statusword is 1, this shall indicate that high voltage is applied to the drive.
If bit 10 (target reached) of the statusword is 1, this shall indicate that the drive has reached the set-point. Bit 10 shall also be set to 1, if the operation mode has been changed. The change of a target value by software shall alter this bit. If quick stop option code is 5, 6, 7 or 8, bit 10 shall be set to 1, when the quick stop operation is finished and the drive is halted. If halt occurred and the drive has halted then bit 10 shall be set to 1, too.

4.2.3 Object 1002h: Manufacturer Status Register

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

Object description:

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

Entry description:

Access	RO
PDO mapping	Optional
Value range	UNSIGNED32
Default value	No

Table 4.7 *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, Status Word

4.2.4 Object 6060h: Modes of Operation

The object selects the mode of operation of the drive.

Object description:

Index	6060_h
Name	Modes of Operation
Object code	VAR
Data type	SINT8

Entry description:

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

Value description:

Value	Description
-128…-6	Reserved
-5	Manufacturer specific – External Reference Torque Mode
-4	Manufacturer specific – External Reference Velocity Mode
-3	Manufacturer specific – External Reference Position Mode
-2	Manufacturer specific – Electronic Camming Position Mode
-1	Manufacturer specific – Electronic Gearing Position Mode
0	No mode change/no mode assigned
1	Profile Position Mode
2	Reserved
3	Profile Velocity Mode
4	Reserved
5	Reserved
6	Homing Mode
7	Interpolated Position Mode
8	Cyclic sync Position Mode
9	Cyclic sync Velocity Mode
10	Cyclic sync Torque Mode
11…127	Reserved

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

4.2.5 Object 6061h: Modes of Operation Display

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

Object description:

Index	6061_h
Name	Modes of Operation Display
Object code	VAR
Data type	SINT

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	-128 … 127
Default value	No

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

4.3 Error monitoring

4.3.1 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	1001_h
Name	Error register
Object code	VAR
Data type	UNSIGNED8

Entry description:

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

Table 4.8 *Bit description of the object*

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.

4.3.2 Object 2000h: Motion Error Register

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

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

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 4.9 *Bit Assignment in Motion Error Register*

Bit	Description
15	Drive disabled due to enable input. Set when enable input is on disable state. Reset when enable 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: 0xFF03 - Specified homing method not available 0xFF04 - A wrong mode is set in object 6060h, modes_of_operation 0xFF05 - Specified digital I/O line not available A function is called during the execution of another function (+ set bit 7 of object 6041h, statusword) Update of operation mode received during a transition This bit acts just as a warning.
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	Motor position wraps around. Set when protection is triggered. Reset by a Reset Fault command
4	Communication error. Set when protection is triggered. Reset by a Reset Fault command
3	Control error (position/speed error too big). Set when protection is triggered. Reset by a Reset Fault command
2	Invalid setup data. Set when the EEPROM stored setup data is not valid or not present.
1	Short-circuit. Set when protection is triggered. Reset by a Reset Fault command
0	EtherCAT® communication error. Reset by a Reset Fault command or by Clear Error in the EtherCAT® State Machine.

4.3.3 Object 2001h: Motion Error Register Mask

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	$FFFF_h$

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

4.3.4 Object 2002h: Detailed Error Register

The Detailed Error Register displays detailed information about the errors signaled with command Error bit from Motion Error Register. This register also displays the status of software limit switches. 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 4.10 *Bit Assignment in Motion Error Register*

Bit	Description
15	Reserved
14	Reserved
13	Reserved
12	Reserved
11	Reserved
10	Reserved
9	Reserved
8	S-curve parameters caused and 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 MPL stack. Last function call was ignored.

4.3.5 Object 605Ah: Quick stop option code

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

4.3.6 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 slow down ramp is a deceleration value set by the Profile acceleration object, index 6083_h.

Object description:

Index	$605B_h$
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 slow down ramp and disable the drive function
2…32767	Reserved

4.3.7 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 slow down ramp is a deceleration value set by the Profile acceleration object, index 6083_h.

Object description:

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

Entry description:

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

Data description:

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

4.3.8 Object 605Dh: Halt option code

This object determines what action is taken if when the halt command 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	$605D_h$
Name	Halt option code
Object code	VAR
Data type	INTEGER16

Entry description:

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

Data description:

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

4.3.9 Object 605Eh: Fault reaction option code

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

- Under-voltage
- Over-voltage
- Drive over-temperature
- Motor over-temperature
- I^2t –when the internal register ASR bit1 is 0 in setup.
- Communication error (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

4.3.10 Object 6007h: Abort connection option code

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

Object description:

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

Entry description:

Access	RW
PDO mapping	Possible
Value range	-32768…32767
Default value	0

Table 5.1 *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

4.4 Digital I/O control and status objects

4.4.1 Object 60FDh: 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) and Switched ON. If the bit is RESET, then the corresponding drive input status is logical '0' (low) and Switched OFF.

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	**Description**
Manufacturer specific	31	IN15 status
	30	IN14 status
	29	IN13 status
	28	IN12 status
	27	IN11 status
	26	IN10 status
	25	IN9 status
	24	IN8 status
	23	IN7 status
	22	IN6 status
	21	IN5 status
	20	IN4 status
	19	IN3 status
	18	IN2 status
	17	IN1 status
	16	IN0 status
Device profile defined	15..4	Reserved
	3	Interlock (Drive enable)
	2	Home switch status
	1	Positive limit switch status
	0	Negative limit switch status

4.4.2 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 it is allowed by the mask in the second sub-index which defines the outputs that can be controlled.

If any of the bits is **SET**, then the corresponding drive output will be switched to logical ‘1’ (high) and Switched ON. If the bit is **RESET**, then the corresponding drive output will be switched to logical ‘0’ (low) and Switched OFF.

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 specified, the drive will issue an emergency message.

Object description:

Index	$60FE_h$
Name	Digital outputs
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

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

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

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

Table 4.11 *Bits mask description:*

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

4.4.2.1 Example: setting digital outputs

Set OUT0 to 1(ON) and OUT1 to 0 (OFF).

1. **Set sub-index 1 with the desired outputs states.**
   - **set** bit 16 (1) to set OUT0 to High state
   - **reset** bit17 (0) to set OUT1 to Low state.

   Set in **60FEh sub-index 1** to 0x00010000.
2. **Set in sub-index 2 the outputs that need to be changed.**
   - **set** bits 16 and 17 to 1, to apply sub-index 1 settings only for OUT0 and OUT1

   Set in **60FEh sub-index 2** to 0x00030000.

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

The sub-index setting order can also be reversed:

set sub-index 2 with the outputs that will be controlled.

set sub-index 1 with the needed output values

4.4.3 Object 2045h: Digital outputs status

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

Object description:

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

Entry description:

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

Data description:

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

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

4.4.4 Object 2102h: Brake Status

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

Object description:

Index	2102_h
Name	Brake Status
Object code	VAR
Data type	USINT

Entry description:

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

4.4.5 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	2046_h
Name	Analogue input: Reference
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65472
Default value	No

4.4.6 Object 2047h: Analogue input: Feedback

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

Object description:

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

Entry description:

Access	RO
PDO mapping	Possible
Units	-
Value range	0 … 65272
Default value	No

4.4.7 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	2055_h
Name	Analogue input: DC-link voltage
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RO
PDO mapping	Possible
Units	Internal Units (IU)
Value range	0 … 65472
Default value	No

The computation formula for the *DC-link voltage [IU]* to [V] is:

$$DC-link\ \_Voltage[V]=\frac{VdcMaxMeasurable}{65520}\times DC-link\ \_Voltage[IU]$$

where *VdcMaxMeasurable* is the Maximum measurable DC voltage found in Setup/Drive Setup/Drive info button.

4.4.8 Object 2058h: Drive Temperature

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

Note: Some drives may not have a temperature sensor. In this case, the object will not show reliable data.

Object description:

Index	2058_h
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

4.4.9 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	$208B_h$
Name	Sin AD signal from Sin/Cos encoder
Object code	VAR
Data type	INTEGER16

Entry description:

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

4.4.10 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	$208C_h$
Name	Cos AD signal from Sin/Cos encoder
Object code	VAR
Data type	INTEGER16

Entry description:

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

[1] Object 208Bh is available only on F515C and above firmware

[2] Object 208Ch is available only on F515C and above firmware

4.4.11 Object 208Eh: Auxiliary Settings Register

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

Object description:

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

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	-

Table 5.2 **Bit Assignment in Auxiliary Settings Register**

Bit	**Value**	**Meaning**
4-15	0	Reserved.
3	0	When 6040 bit 14 = 1, at the next *update*[1], the Target Speed Starting Value is the Actual Speed
	1	When 6040 bit 14 = 1, at the next *update*, the Target Speed Starting Value is zero.
0-2	0	Reserved.

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

4.4.12 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	2108_h
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

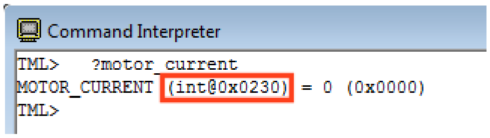
Sub-index	1
Description	16 bit variable address
Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	0x0230 (adr. or motor current)

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

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

How it works:

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



```
TML>   ?motor_current
MOTOR_CURRENT (int@0x0230) = 0 (0x0000)
TML>
```

The variable address can be found between the parenthesis.

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

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

4.5 Protections Setting Objects

4.5.1 Object 607Dh: Software position limit

The object sets the maximal and minimal software position limits. If the actual position is lower than the negative position limit or higher than the positive one, a software position limit emergency message will be launched. Also the motion will do a quick stop using Object 6085h: Quick stop deceleration data.

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

Entry description:

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

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

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

4.5.2 Object 2050h: 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	Possible
Units	CU
Value range	0 … 32767
Default value	No

4.5.3 Object 2051h: 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 equal to the drive slow loop sampling period which is set by default to 1ms.

Object description:

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

Entry description:

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

4.5.4 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: I2t protection integrator limit for more details.

Object description:

Index	2052_h
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

4.5.5 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	$207F_h$
Name	Current actual value
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 $207E_h$.

4.5.6 Object 2053h: I2t protection integrator limit

Objects 2053_h and 2054_h 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 2052_h, motor nominal current. Select a point on the I^2t motor thermal protection curve, which is characterized by the points I_I2t (current, [A]) and t_I2t: (time, [s]) (see **Figure 4.3**)

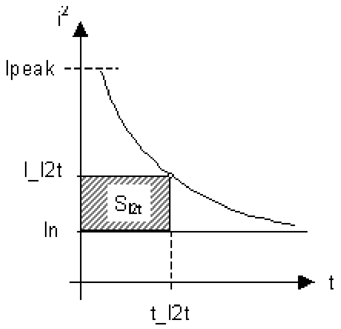


Figure 4.3 *I2t motor thermal protection curve*

The points I_I2t and t_I2t on the motor thermal protection curve together with the nominal motor current **In** define the surface S_{I2t}. If the motor instantaneous current is greater than the nominal current In and the I2t protection is activated, the difference between the square of the instantaneous current and the square of the nominal current is integrated and compared with the SI2t value (see **Figure 4.4**). When the integral equals the SI2t surface, the I2t protection is triggered.

Object description:

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

Entry description:

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

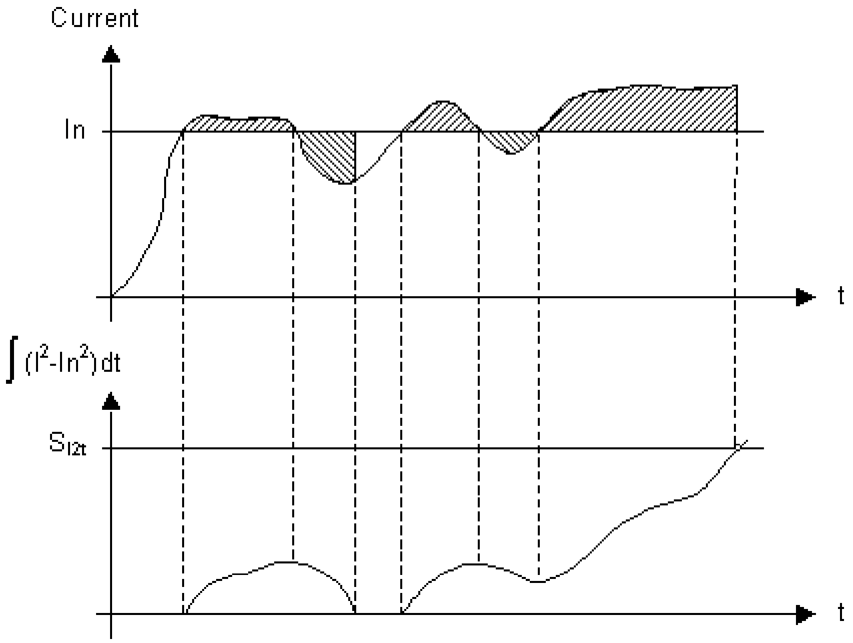


Figure 4.4 *I2t protection implementation*

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

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

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

4.5.7 Object 2054h: I2t protection scaling factor

Object description:

Index	2054_h
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

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

where Ts_S is the sampling time of the speed control loop [s], and t_I2t is the I2t protection time corresponding to the point on the graphic in **Figure 4.3.**

4.6 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, it means using a stepper in open loop configuration and an incremental encoder to detect lost steps. When the protection triggers, the drive enters Fault state signaling a Control error.

4.6.1 Object 2083h: Encoder Resolution

Sets the number of encoder counts the motor does for one full rotation. For example, if the quadrature encoder has 500 lines (2000 counts/rev), 2000 must be written into the object.

Object description:

Index	2083_h
Name	Encoder resolution
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Yes
Value range	INTEGER32
Default value	2000

4.6.2 Object 2084h: Stepper Resolution

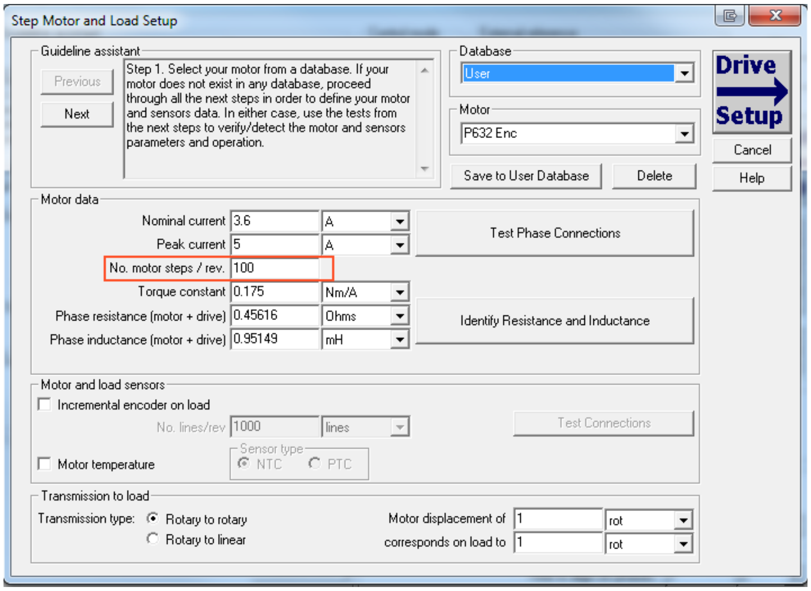
Sets the number of microsteps the step motor does for one full rotation. For example, if the motor has 100 steps / revolution (see Figure 2.1) and is controlled with 256 microsteps / step (see Figure 2.2), you must write 100x256=25600 into this object.

Object description:

Index	2084_h
Name	Stepper resolution
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Yes
Value range	INTEGER32
Default value	51200



Guideline assistant

Step 1. Select your motor from a database. If your motor does not exist in any database, proceed through all the next steps in order to define your motor and sensors data. In either case, use the tests from the next steps to verify/detect the motor and sensors parameters and operation.

Database: User

Motor: P632 Enc

Motor data

- Nominal current: 3.6 A
- Peak current: 5 A
- No. motor steps / rev.: 100
- Torque constant: 0.175 Nm/A
- Phase resistance (motor + drive): 0.45616 Ohms
- Phase inductance (motor + drive): 0.95149 mH

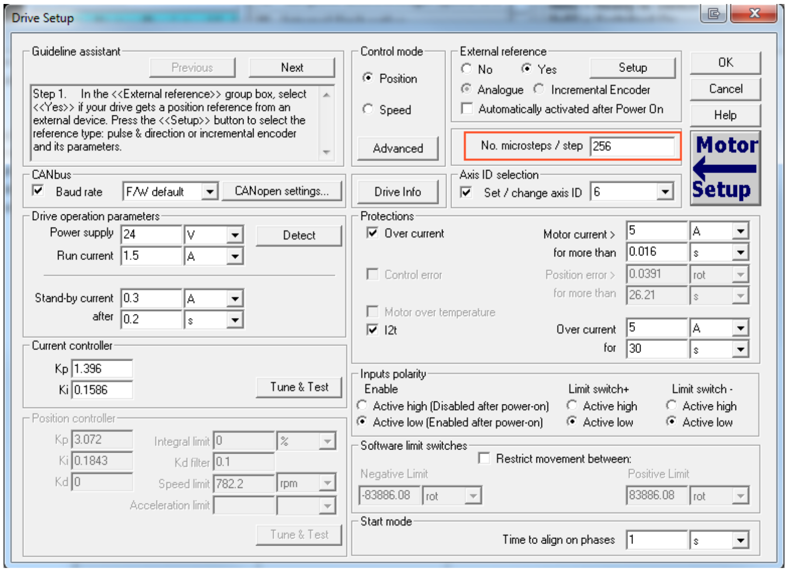
Motor and load sensors

- ☐ Incremental encoder on load
- No. lines/rev: 1000 lines
- ☐ Motor temperature
- Sensor type: ● NTC ○ PTC

Transmission to load

- Transmission type: ● Rotary to rotary ○ Rotary to linear
- Motor displacement of: 1 rot
- corresponds on load to: 1 rot

***Figure 2.1** Motor steps / revolution*



Guideline assistant

Step 1. In the <<External reference>> group box, select <<Yes>> if your drive gets a position reference from an external device. Press the <<Setup>> button to select the reference type: pulse & direction or incremental encoder and its parameters.

Control mode: ● Position ○ Speed

External reference: ○ No ● Yes

● Analogue ○ Incremental Encoder

☐ Automatically activated after Power On

No. microsteps / step: 256

CANbus: ☑ Baud rate: F/W default

Axis ID selection: ☑ Set / change axis ID: 6

Drive operation parameters

- Power supply: 24 V
- Run current: 1.5 A
- Stand-by current: 0.3 A
- after: 0.2 s

Current controller

- Kp: 1.396
- Ki: 0.1586

Position controller

- Kp: 3.072
- Ki: 0.1843
- Kd: 0
- Integral limit: 0 %
- Kd filter: 0.1
- Speed limit: 782.2 rpm
- Acceleration limit:

Protections

- ☑ Over current: Motor current > 5 A, for more than 0.016 s
- ☐ Control error: Position error > 0.0391 rot, for more than 26.21 s
- ☐ Motor over temperature
- ☑ I2t: Over current 5 A, for 30 s

Inputs polarity

- Enable: ○ Active high (Disabled after power-on) ● Active low (Enabled after power-on)
- Limit switch+: ○ Active high ● Active low
- Limit switch -: ○ Active high ● Active low

Software limit switches

- ☐ Restrict movement between:
- Negative Limit: -83886.08 rot
- Positive Limit: 83886.08 rot

Start mode

- Time to align on phases: 1 s

***Figure 2.2** Motor microsteps / step*

4.6.3 Enabling step loss detection protection

Before enabling the step loss detection protection, the *Encoder resolution* in object 2083h and the *Stepper resolution* in object 2084h must be set.

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

The following error window is expressed in microsteps. The Following error time is expressed in multiples of position/speed control loops (0.8ms by default for stepper configurations).

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

Example: Following error window is set to 1000 and *Following error time* is set to 20. The step motor has 100 steps/rev and is controlled with 256 microsteps/step. The step loss protection will be triggered if the difference between the commanded position and the measured position is bigger than 1000 microsteps (i.e. 1000/(100*256) rev = 14,06 degrees) for a time interval bigger or equal than 20 control loops of 0.8ms each i.e. 16ms.

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

4.6.4 Step loss protection setup

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

1. Move your motor with the highest velocity and load planned to be used in your application
2. During the movement at maximal speed, read object 60F4h - *Following error actual value* as often as possible to determine its highest value.

   Remark: *Following error actual value* can be read at every control loop using MotionPRO Developer or PRO Config by logging the MPL variable POSERR.
3. Add a margin of about 25% to the highest error value determined at previous step and set the new obtained value into object 6065h - *Following error window*.
4. Activate the step loss detection by writing a non-zero value in object 6066h - *Following error time out.* Recommended values are between 1 and 10.

4.6.5 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 Control Word.
- Send Switch On command into Control Word. At this moment, voltage is applied to the motor and it will execute the phase alignment procedure again. The position error will be reset automatically.
- Start a homing procedure to find again the motor zero position.

4.6.6 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 6064h Position actual value and 6062h Position demand.

Only object 6063h *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 6064h *Position actual value* and 6062h *Position demand value* will provide correct values.

4.7 Drive info objects

4.7.1 Object 1000h: 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 is described in the CiA standard . It is 0x0000 if no standardized device profile is used
- The 16MSB is the value given by ElectroCraft for it's products.

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 PRO Series family

4.7.2 Object 6502h: Supported drive modes

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

Object description:

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

Entry description:

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

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

Data description:

MSB **LSB**

0 0	X …x	0 0	1	1	1	1	1	0	1	1	1	1
Manufacturer specific		reserved	cst	csv	csp	ip	hm	reserved	tq	pv	vl	pp
31 21	20 …16	15 10	9	8	7	6	5	4	3	2	1	0

Data description – manufacturer specific:

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

4.7.3 Object 1008h: 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	RO
PDO mapping	No
Value range	No
Default value	PRO-A08V48x[1] PRO-A10V80x[1] PRO-A20V80x[1]

4.7.4 Object 1009h: Manufacturer Hardware Version

The object contains the manufacturer hardware version 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	RO
PDO mapping	No
Value range	No
Default value	Product dependent

4.7.5 Object 100Ah: 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	RO
PDO mapping	No
Value range	No
Default value	Product dependent

4.7.6 Object 2060h: Software version of a MPL application

By inspecting this object the user can find out the software version of the MPL application (drive setup plus motion setup and eventually cam tables) that is stored in the EEPROM memory of the drive. The object stores the software version coded in a string of 4 elements, grouped in a 32-bit variable. Each byte represents an ASCII character. The value can be written in Drive Setup/Drive Info/Application ID.

Object description:

Index	2060_h
Name	Software version of MPL 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 MPL application is read as:

0x56 – ASCII code of letter **V**

0x31 – ASCII code of number **1**

0x2E – ASCII code of character **.**

0x32 – ASCII code of number **2**

So the version is **V1.2**.

4.7.7 Object 1018h: Identity Object

This object provides general information about the device.

Sub-index 01_h shows the unique Vendor ID allocated to ElectroCraft ($1A3_h$).

Sub-index 02_h contains the ElectroCraft 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 ElectroCraft product ID is P027.214.E121, subindex 02_h will be read as the number 27214121 in decimal.

Sub-index 03_h shows the Revision number.

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

Object description:

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

Entry description:

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

Sub-index	01_h
Description	Vendor ID
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	000000FFh

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

Sub-index	03_h
Description	Revision number
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	-

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

4.8 Miscellaneous Objects

4.8.1 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	2025_h
Name	Stepper current in open-loop operation
Object code	VAR
Data type	INTEGER16

Entry description:

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

4.8.2 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	2026_h
Name	Stand-by current for stepper in open-loop operation
Object code	VAR
Data type	INTEGER16

Entry description:

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

4.8.3 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. The value is set in ms.

Object description:

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

Entry description:

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

4.8.4 Object 2100h: Number of Steps per revolution

This object shows the number of steps per revolution, in case a stepper motor is defined in setup.

It is the same number that is defined in ***Motor Setup*** in the filed ***No. motor steps / rev***.

Object description:

Index	2100_h
Name	Number of Steps per revolution
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Value range	0 … 32767
Default value	-

4.8.5 Object 2101h: Number of microSteps per Step

This object shows the number of microSteps per motor Step, in case a stepper open loop configuration is defined in setup.

It is the same number that is defined in ***Drive Setup*** in the filed ***No. microsteps / step.***

Object description:

Index	2101_h
Name	Number of microSteps per Step
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Value range	0 … 32767
Default value	-

4.8.6 Object 2103h: Number of encoder counts per revolution

This object represents the number of encoder increments the motor will register after one full revolution.

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

Object description:

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

Entry description:

Access	RO
PDO mapping	Possible
Value range	0 … 2^{31}-1
Default value	-

4.8.7 Object 2075h: Position triggers

This object is used in order to define a set of 4 position values whose proximity will be signaled through bits 17-20 of object 1002_h, *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

4.8.8 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.

The save operation can be monitored through **Object 208Ah: Save setup status**.

Object description:

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

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	-

4.8.9 Object 208Ah: Save setup status

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

Object description:

Index	$208A_h$
Name	Save setup status
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	YES
Value range	INTEGER16
Default value	1

4.8.10 Object 2080h: Reset drive

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

Object description:

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

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	-

4.8.11 Object 2082h: Sync on fast loop

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

By default, the fast loop period for all configurations is set to 0.1 ms, the slow loop period is 0.8 ms for stepper configurations and 1ms for the rest.

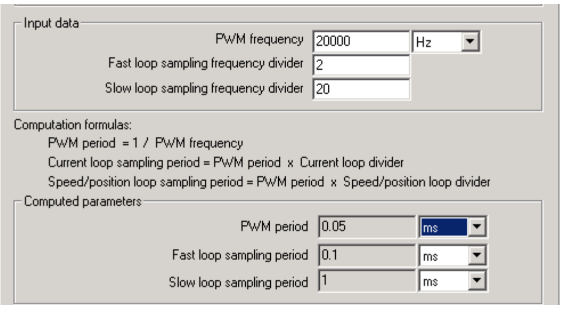
Object description:

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

Entry description:

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

The fast or slow loop sample period can be set and be seen in PRO Config->Drive Setup-> Advanced button.



Input data

PWM frequency 20000 Hz

Fast loop sampling frequency divider 2

Slow loop sampling frequency divider 20

Computation formulas:

PWM period = 1 / PWM frequency

Current loop sampling period = PWM period x Current loop divider

Speed/position loop sampling period = PWM period x Speed/position loop divider

Computed parameters

PWM period 0.05 ms

Fast loop sampling period 0.1 ms

Slow loop sampling period 1 ms

Example:

Assuming the Fast loop sampling period is 0.1ms and the Slow loop sampling period is 1 ms, the following cases should be considered:

- 2082_h = 1, the SYNC 0 time should be set to 0.1 ms.
- 2082_h = 0, the SYNC 0 time should be set to 1 ms.

If the SYNC 0 time from the Distributed Clock is not set the same as the drive synchronization period, the drive will not synchronize. See also paragraph **1.2.4**.

4.8.12 Object 2085h: 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 1002h *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 4.12 **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

[1] OUT5 setting is available only on some PRO-A08V48 drives.

5 Factor group

The PRO-CAT 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 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.

5.1 Factor group objects

5.1.1 Object 607Eh: Polarity

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

Object description:

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

Entry description:

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

The *Polarity* object has the following bit assignment:

Table 5.3 **Bit Assignment in Polarity object:**

Bit	**Bit name**	**Value**	**Meaning**
7	Position polarity	0	Multiply by 1 the values of objects 607Ah, 6062h and 6064h
		1	Multiply by -1 the values of objects 607Ah, 6062h and 6064h
6	Velocity polarity	0	Multiply by 1 the values of objects 60FFh, 606Bh and 606Ch
		1	Multiply by -1 the values of objects 60FFh, 606Bh and 606Ch
5-0	reserved	0	Reserved

5.1.2 Object 6089h: Position notation index

The *position notation index* is used to scale the following objects*:*

- Position actual value
- Position demand value
- Target position
- Position window
- Following error window
- Following error actual value

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

5.1.3 Object 608Ah: Position dimension index

The *position dimension index* is used to scale the following objects*:*

- Position actual value
- Position demand value
- Target position
- Position window
- Following error window
- Following error actual value

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

5.1.4 Object 608Bh: Velocity notation index

The *velocity notation index* is used to scale the following objects*:*

- Velocity actual value
- Velocity demand value
- Target velocity
- Profile velocity

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

5.1.5 Object 608Ch: Velocity dimension index

The *velocity dimension index* is used to scale the following objects*:*

- Velocity actual value
- Velocity demand value
- Target velocity
- Profile velocity

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

5.1.6 Object 608Dh: Acceleration notation index

The *acceleration notation index* is used to scale the following objects*:*

- Profile acceleration
- Quick stop deceleration

Object description:

Index	608D$_h$
Name	Acceleration notation index
Object code	VAR
Data type	INTEGER8

Entry description:

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

5.1.7 Object 608Eh: Acceleration dimension index

The *acceleration dimension index* is used to scale the following objects:

- Profile acceleration
- Quick stop deceleration

Object description:

Index	608E$_h$
Name	Acceleration dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

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

5.1.8 Object 206Fh: Time notation index

The *time dimension index* is used to scale the following objects:

- Following error time out
- Position window time
- Jerk time
- Max slippage time out
- Over-current time out

Object description:

Index	$206F_h$
Name	Time notation index
Object code	VAR
Data type	INTEGER8

Entry description:

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

5.1.9 Object 2070h: Time dimension index

The *time dimension index* is used to scale the following objects:

- Following error time out
- Position window time
- Jerk time
- Max slippage time out
- Over-current time out

Object description:

Index	2070_h
Name	Time dimension index
Object code	VAR
Data type	UNSIGNED8

Entry description:

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

5.1.10 Object 6093h: Position factor

The *position factor* converts the desired position (in position units) into the internal format (in increments):

$$Position[IU] = Position[UserUnits] \times \frac{PositionFactor.Numerator}{PositionFactor.Divisor}$$

Object description:

Index	6093_h
Name	Position factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Entry description:

Sub-index	01_h
Description	Numerator
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

Sub-index	02_h
Description	Divisor
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

5.1.11 Object 6094h: Velocity encoder factor

The *velocity encoder factor* converts the desired velocity (in velocity units) into the internal format (in increments).

$$Velocity[IU] = Velocity[UserUnits] \times \frac{VelocityEncoderFactor.Numerator}{VelocityEncoderFactor.Divisor}$$

Object description:

Index	6094_h
Name	Velocity encoder factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Entry description:

Sub-index	01_h
Description	Numerator
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

Sub-index	02_h
Description	Divisor
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

5.1.12 Object 6097h: Acceleration factor

The *acceleration factor* converts the velocity (in acceleration units/sec^2) into the internal format (in increments/sampling^2).

$$Acceleration[IU] = Acceleration[UserUnits] \times \frac{AccelerationFactor.Numerator}{AccelerationFactor.Divisor}$$

Object description:

Index	6097_h
Name	Acceleration factor
Object code	ARRAY
Number of elements	2
Data type	UNSIGNED32

Entry description:

Sub-index	01_h
Description	Numerator
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

Sub-index	02_h
Description	Divisor
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	1

5.1.13 Object 2071h: Time factor

The *time factor* converts the desired time values (in time units) into the internal format (in speed / position loop samplings).

$$Time[IU] = Time[UserUnits] \times \frac{TimeFactor.Numerator}{TimeFactor.Divisor}$$

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 Homing Mode

6.1 Overview

Homing is the method by which a drive seeks the home position. There are various methods to achieve this position using the four available sources for the homing signal: limit switches (negative and positive), home switch (IN0) and index pulse.

Remark: on an PRO Series drive or integrated motor, the home switch is always the digital input IN0.

A homing move is started by setting bit 4 of the *Control Word* object (index 0x6040). The results of a homing operation can be accessed in the *Status Word* (index 0x6041).

A homing mode is chosen by writing a value to homing method which will clearly establish:

1. the homing signal (positive limit switch, negative limit switch, home switch)
2. the direction of actuation and where appropriate
3. the position of the index pulse.

The user can specify the home method, the home offset, the speed and the acceleration.

The **home offset** (object 607C$_h$) is the difference between the zero position for the application and the machine home position. During homing, the home position is found. Once the homing is completed, the zero position is offset from the home position by adding the home_offset to the home position. This is illustrated in the following diagram.

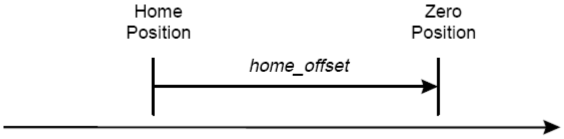


Figure 6.1 *Home Offset*

In other words, after the home position has been found, the drive will set the actual position (object 6064$_h$) with the value found in object 607C$_h$.

There are two **homing speeds:** a fast speed (which is used to find the home switch), and a slow speed (which is used to find the index pulse).

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 Standard).

As in the figure below for each homing method we will present a diagram that clearly describes the sequence of homing operation.

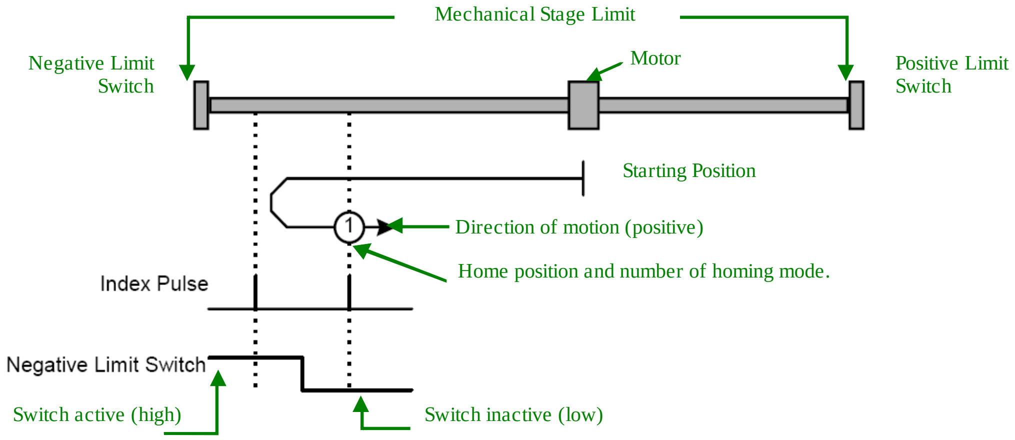


Figure 6.2 *Homing method diagram*

6.2 Homing methods

6.2.1 Method 1: Homing on the Negative Limit Switch.

If the negative limit switch is inactive (low) the initial direction of movement is leftward (negative sense). After negative limit switch is reached the motor will reverse the motion, moving in the positive sense with slow speed. The home position is at the first index pulse to the right of the position where the negative limit switch becomes inactive.

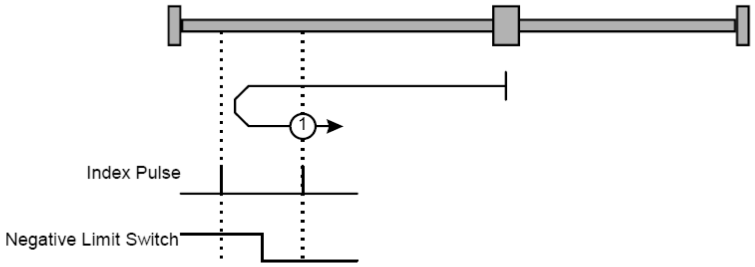


Figure 6.3 *Homing on the Negative Limit Switch*

6.2.2 Method 2: Homing on the Positive Limit Switch.

If the positive limit switch is inactive (low) the initial direction of movement is rightward (negative sense). After positive limit switch is reached the motor will reverse the motion, moving in the negative sense with slow speed. The home position is at the first index pulse to the left of the position where the positive limit switch becomes inactive.

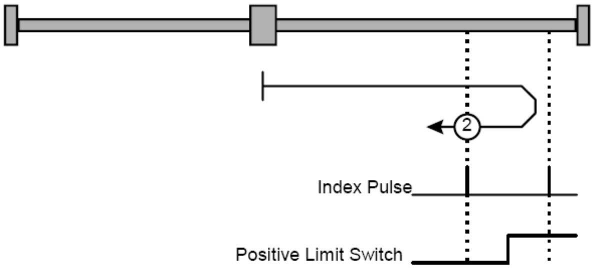


Figure 6.4 *Homing on the Positive Limit Switch*

6.2.3 Methods 3 and 4: Homing on the Positive Home Switch and Index Pulse.

The home position is at the index pulse either after home switch high-low transition (method 3) or after home switch low-high transition (method 4).

The initial direction of movement is dependent on the state of the home switch (if low - move positive, if high - move negative).

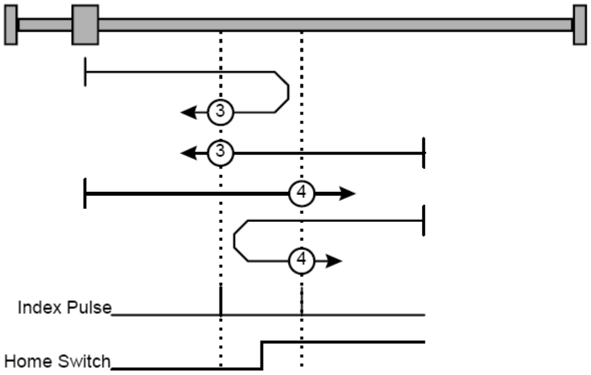


Figure 6.5 *Homing on the Negative Home Switch and Index Pulse*

For **method 3**, if home input is high the initial direction of movement will be negative, or positive if home input is low, and reverse (with slow speed) after home input low-high transition. The motor will stop at first index pulse after home switch high-low transition.

For **method 4**, if home input is low the initial direction of movement will be positive, or negative if home input is high, and reverse (with slow speed) after home input high-low transition. The motor will stop at first index pulse after home switch low-high transition.

In all cases after home switch transition the speed of the movement is slow.

6.2.4 Methods 5 and 6: Homing on the Negative Home Switch and Index Pulse.

The home position is at the index pulse either after home switch high-low transition (method 5) or after home switch low-high transition (method 6).

The initial direction of movement is dependent on the state of the home switch (if high - move positive, if low - move negative).

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.

In all cases after home switch transition the speed of the movement is slow.

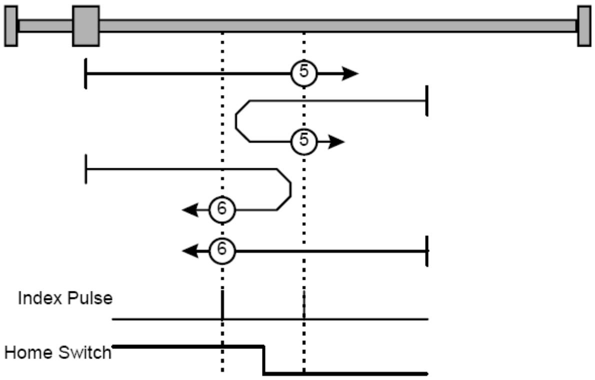


Figure 6.6 *Homing on the Negative Home Switch and Index Pulse*

6.2.5 Methods 7 to14: Homing on the Negative Home Switch and Index Pulse.

These methods use a home switch that is active over only a portion of the travel distance, in effect the switch has a ‘momentary’ action as the axle’s position sweeps past the switch.

Using methods 7 to 10 the initial direction of movement is to the right (positive), and using methods 11 to 14 the initial direction of movement is to the left (negative), except the case when the home switch is active at the start of the motion (initial direction of motion is dependent on the edge being sought – the rising edge or the falling edge).

The home position is at the index pulse on either side of the rising or falling edges of the home switch, as shown in the following two diagrams.

If the initial direction of movement leads away from the home switch, the drive will reverse on encountering the relevant limit switch (negative limit switch for methods 7 to 10, or positive limit switch for methods 11 to 14).

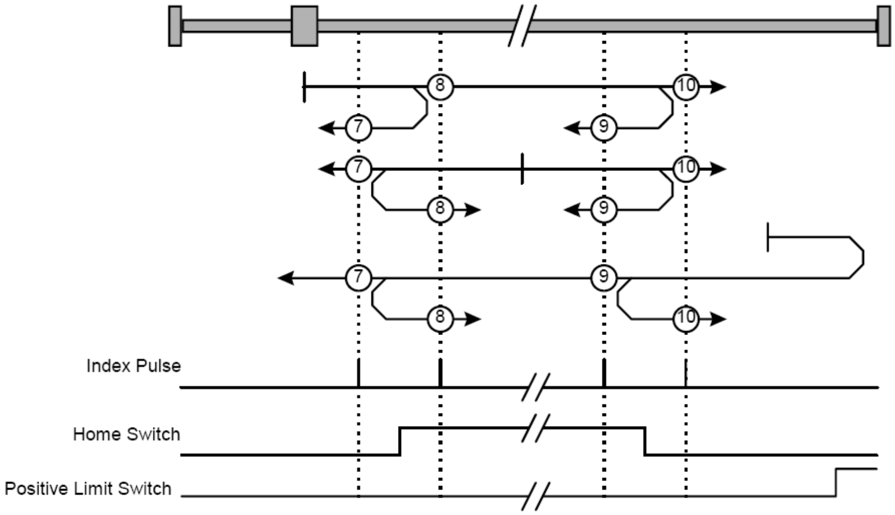


Figure 6.7 *Homing on the Home Switch and Index Pulse – Positive Initial Move*

Using **method 7** the initial move will be positive if home input is low and reverse after home input low-high transition, or move negative if home input is high. Reverse also if the positive limit switch is reached. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

Using **method 8** the initial move will be positive if home input is low, or negative if home input is high and reverse after home input high-low transition. Reverse also if the positive limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition). In all cases after low-high home switch transition the motor speed will be slow.

Using **method 9** the initial move will be positive and reverse(slow speed) after home input high-low transition. Reverse also if the positive limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition).

Using **method 10** the initial move will be positive. Reverse if the positive limit switch is reached, then reverse once again after home input low-high transition. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

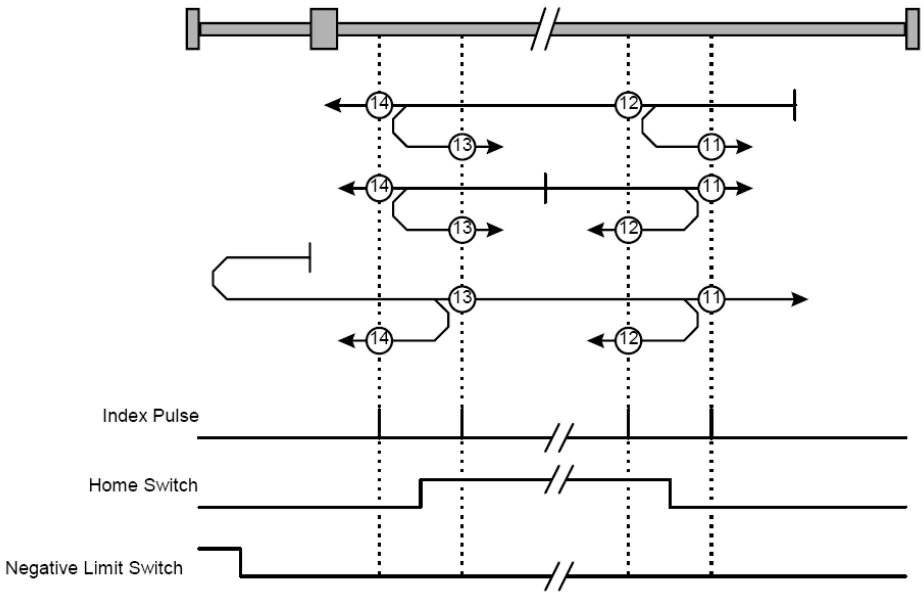


Figure 6.8 *Homing on the Home Switch and Index Pulse – Positive Initial Move*

Using **method 11** the initial move will be negative if home input is low and reverse after home input low-high transition. Reverse also if the negative limit switch is reached. If home input is high move positive. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

Using **method 12** the initial move will be negative if home input is low. If home input is high move positive and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition). In all cases after low-high home switch transition the motor speed will be slow.

Using **method 13** the initial move will be negative and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop at first index pulse after home switch active region starts (low-high transition). In all cases after high-low home switch transition the motor speed will be slow.

Using **method 14** the initial move will be negative. Reverse if the negative limit switch is reached, then reverse once again after home input low-high transition. Stop at first index pulse after home switch active region ends (high-low transition). In all cases after high-low home switch transition the motor speed will be slow.

Methods 15 and 16: Reserved

6.2.6 Methods 17 to 30: Homing without an Index Pulse

These methods are similar to methods 1 to 14 except that the home position is not dependent on the index pulse but only on the relevant home or limit switch transitions.

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.

Using **method 18** if the positive limit switch is inactive (low) the initial direction of movement is rightward (negative sense). After positive limit switch reached the motor will reverse the motion, moving in the negative sense with slow speed. The home position is at the left of the position where the positive limit switch becomes inactive.

For example methods 19 and 20 are similar to methods 3 and 4 as shown in the following diagram.

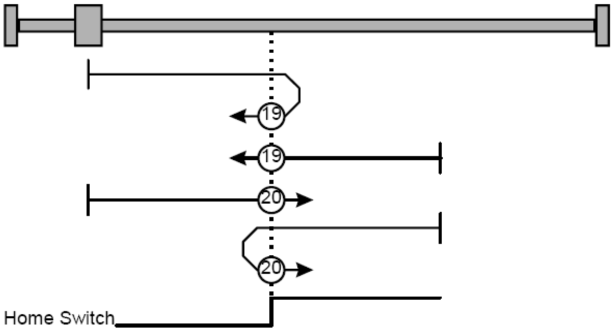


Figure 6.9 *Homing on the Positive Home Switch*

Using **method 19**, if home input is high, the initial direction of movement will be negative, or positive if home input is low, and reverse (with slow speed) after home input low-high transition. The motor will stop right after home switch high-low transition.

Using **method 20**, if home input is low, the initial direction of movement will be positive, or negative if home input is high, and reverse (with slow speed) after home input high-low transition. The motor will stop after right home switch low-high transition.

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.

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).

Using **method 27** the initial move will be negative if home input is low and reverse after home input low-high transition. Reverse also if the negative limit switch is reached. If home input is high move positive. Stop right after home switch active region ends (high-low transition).

Using **method 28** the initial move will be negative if home input is low. If home input is high move positive and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop right after home switch active region starts (low-high transition).

Using **method 29** the initial move will be negative and reverse after home input high-low transition. Reverse also if the negative limit switch is reached. Stop right after home switch active region starts (low-high transition).

Using **method 30** the initial move will be negative. Reverse if the negative limit switch is reached, then reverse once again after home input low-high transition. Stop right after home switch active region ends (high-low transition).

Methods 31 and 32: Reserved

6.2.7 Methods 33 and 34: Homing on the Index Pulse

Using **methods 33** or **34** the direction of homing is negative or positive respectively. During those procedure the motor will move only at slow speed. The home position is at the index pulse found in the selected direction.

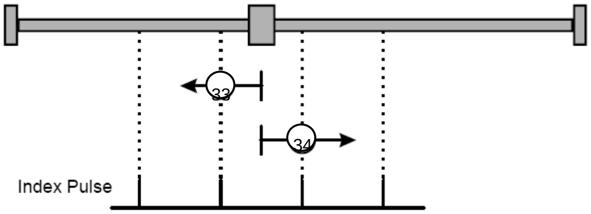


Figure 6.10 *Homing on the Index Pulse*

6.2.8 Method 35: Homing on the Current Position

In **method 35** the current position is set with the value found in *Home offset* (index 0x607C).

Remark: see also *Object 2081h: Set/Change the actual motor position* which can be used to obtain the same outcome as in Method 35.

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

6.2.9.1 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 1.1.5. *Current Threshold < Current Limit*

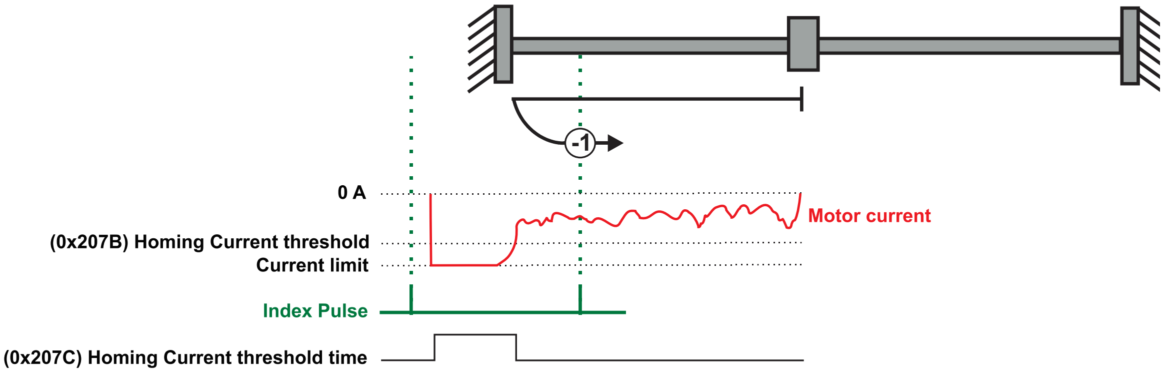

-1
0 A
(0x207B) Homing Current threshold
Current limit
Motor current
Index Pulse
(0x207C) Homing Current threshold time

Figure 6.11 *Homing on the Negative Mechanical Limit and Index Pulse detecting the motor current increase*

6.2.9.2 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 work with the Control Error protection parameters set in Drive Setup.

Move negative until a control error is detected, then reverse and stop at the first index pulse. The home position is at the first index pulse to the right of the negative mechanical limit. At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

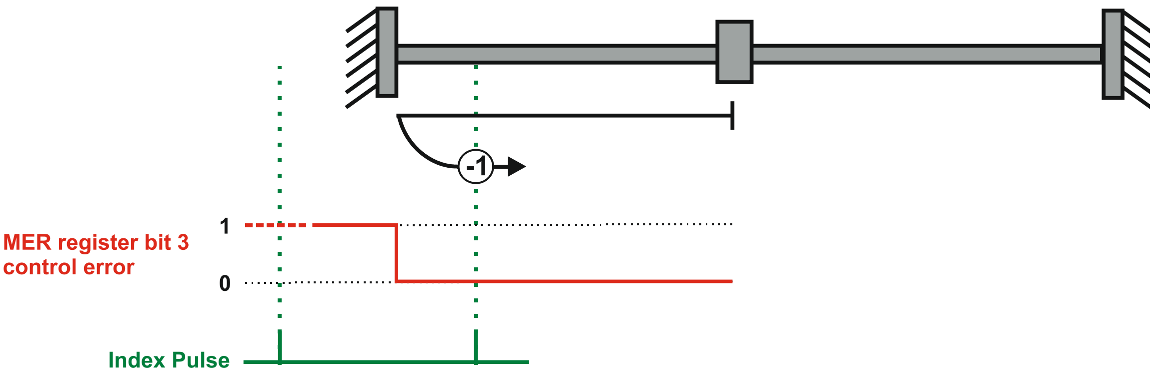


Figure 6.12 *Homing on the Negative Mechanical Limit and Index Pulse detecting a control error*

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

6.2.10.1 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 1.1.5. *Current Threshold < Current Limit*

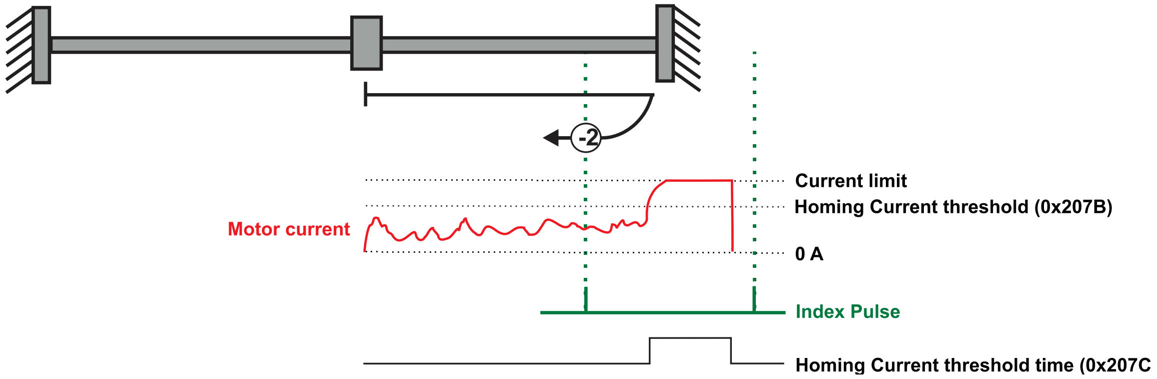

-2
Motor current
Current limit
Homing Current threshold (0x207B)
0 A
Index Pulse
Homing Current threshold time (0x207C

Figure 6.13 *Homing on the Positive Mechanical Limit and Index Pulse detecting the motor current increase*

6.2.10.2 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 work with the Control Error protection parameters set in Drive Setup.

Move positive until a control error is detected, then reverse and stop at the first index pulse. The home position is at the first index pulse to the left of the positive mechanical limit. At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

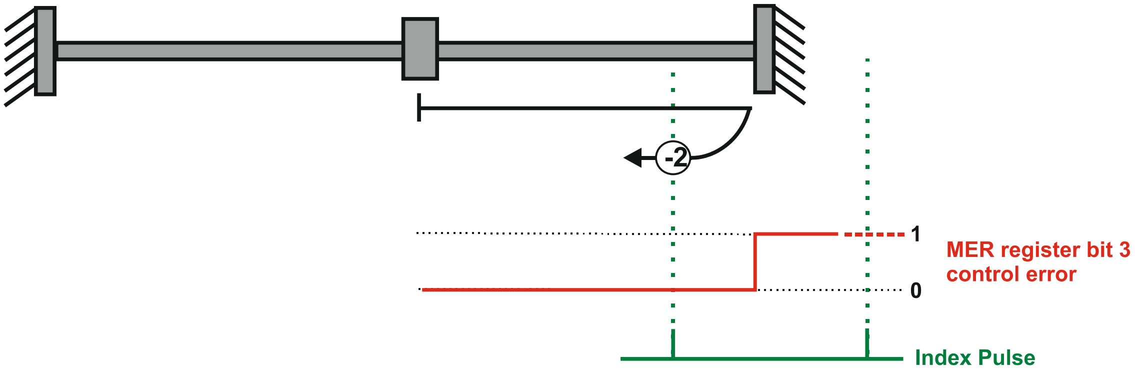


Figure 6.14 *Homing on the Positive Mechanical Limit and Index Pulse detecting a control error*

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

6.2.11.1 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 1.1.5. *Current Threshold < Current Limit*

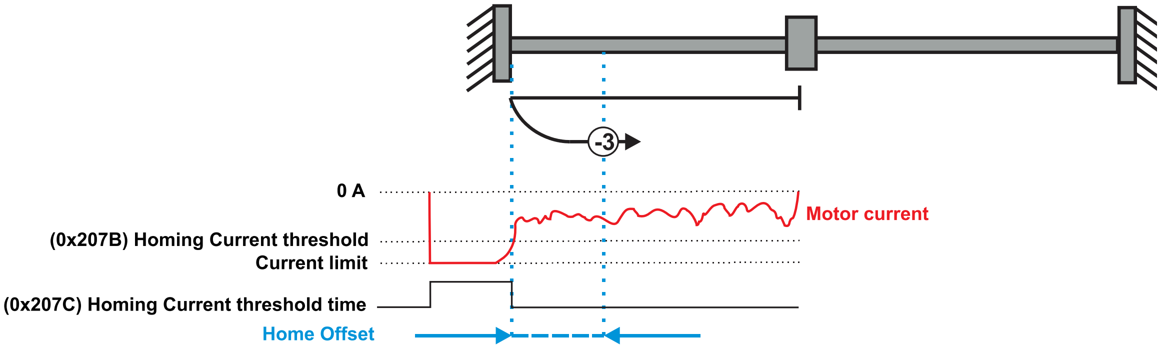

-3
0 A
(0x207B) Homing Current threshold
Current limit
(0x207C) Homing Current threshold time
Motor current
Home Offset

Figure 6.15 *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

6.2.11.2 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 work with the Control Error protection parameters set in Drive Setup. When the motor reaches the mechanical limit and detects a Control Error, it will consider the mechanical limit position as the home position.

Move negative until a control error is detected, then reverse and stop at the position set in “Home position”. The motor will reverse direction and stop after it has travelled the value set in *Home offset* (index 0x607C). At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

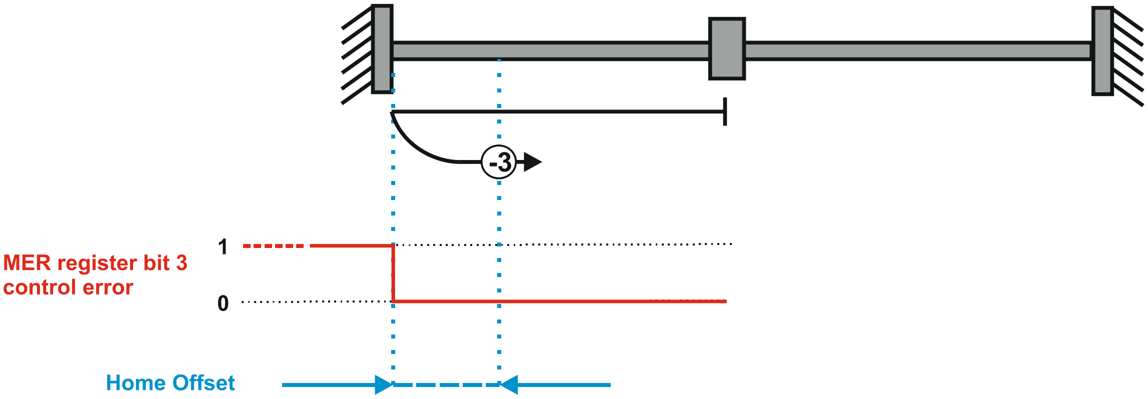


Figure 6.16 *Homing on the Positive Mechanical Limit without an Index Pulse detecting a control error*

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

6.2.12.1 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 1.1.5. *Current Threshold < Current Limit*

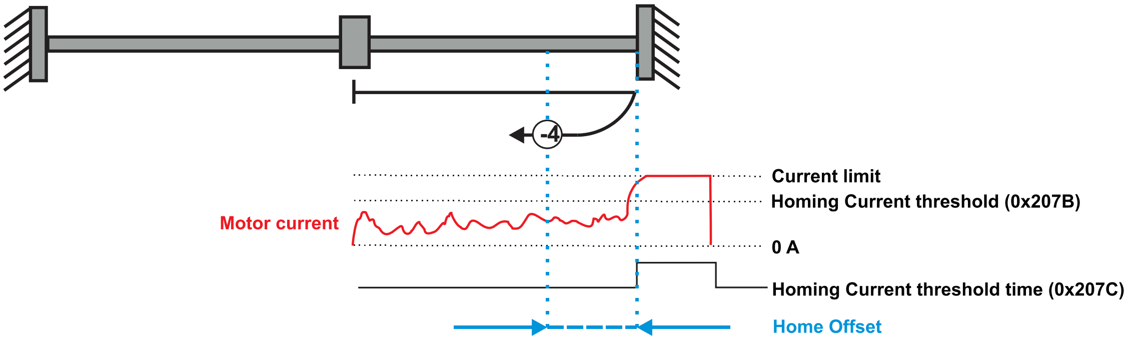

-4
Current limit
Homing Current threshold (0x207B)
Motor current
0 A
Homing Current threshold time (0x207C)
Home Offset

Figure 6.17 *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

6.2.12.2 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 work with the Control Error protection parameters set in Drive Setup. When the motor reaches the mechanical limit and detects a Control Error, it will consider the mechanical limit position as the home position.

Move positive until a control error is detected, then reverse and stop at the position set in “Home position”. The motor will reverse direction and stop after it has travelled the value set in *Home offset* (index 0x607C). At the end of the procedure, the reported motor position will be the one set in *Home offset* (index 0x607C).

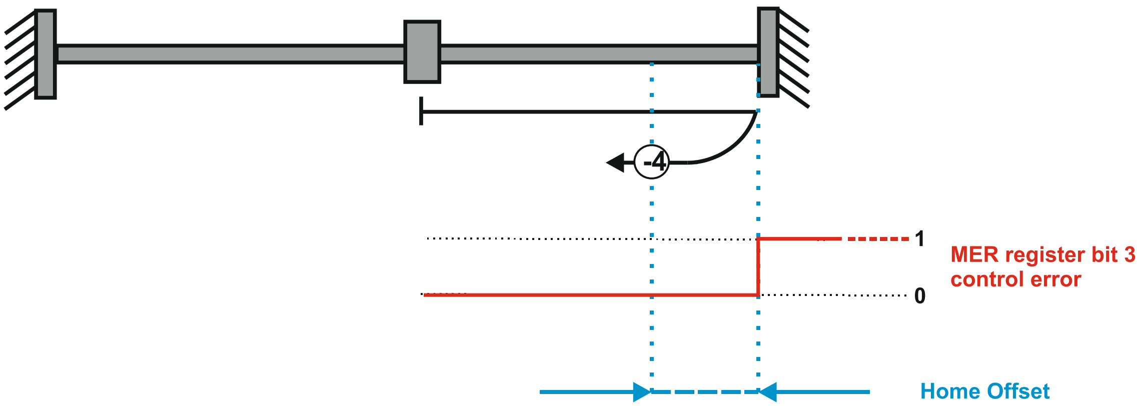


Figure 6.18 *Homing on the Positive Mechanical Limit without an Index Pulse detecting the motor current increase*

6.3 Homing Mode Objects

This chapter describes the method by which the drive seeks the home position. There are 35 built-in homing methods, as described in **paragraph 6.1**. Using the MotionPRO Developer 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 6098h *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.

6.3.1 Controlword in homing mode

MSB **LSB**

See 4.2.1	Halt	See 4.2.1	Reserved	Homing operation start	See 4.2.1
15 9	8	7	6 5	4	3 0

Table 6.1 *Control Word bits description for Homing Mode*

Name	**Bit**	**Value**	**Description**
Homing operation start	4	0	Do not start operation
		0 -> 1	Start homing procedure
		1	Homing mode active
		1 -> 0	Do nothing (does not stop current procedure)
Halt	8	0	Execute the instruction of bit 4
		1	Stop homing procedure drive with *homing acceleration*

6.3.2 Statusword in homing mode

MSB LSB

See 6041h	Homing error	Homing attained	See 6041h	Target reached	See 6041h
15 14	13	12	11	10	9 0

***Table 6.2** Status Word 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 6.3 Definition of bit 10,bit 12 and bit 13

Bit 13	Bit 12	Bit 10	Definition
0	0	0	Homing procedure is in progress
0	0	1	Homing procedure is interrupted or not started
0	1	0	Homing is attained, but target is not reached
0	1	1	Homing procedure is completed successfully
1	0	0	Homing error occurred, velocity is not 0
1	0	1	Homing error occurred, velocity is 0
1	1	X	reserved

6.3.3 Object 607Ch: Home offset

The *home offset* will be set as the new drive position (reported in object 6064_h) after a homing procedure is finished. An exception applies only to the homing motions -3 and -4. See their description for more details.

Object description:

Index	$607C_h$
Name	Home offset
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Units	Position Units
Value range	INTEGER32
Default value	0

6.3.4 Object 6098h: Homing method

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

Object description:

Index	6098_h
Name	Homing method
Object code	VAR
Data type	INTEGER8

Entry description:

Access	RW
PDO mapping	Possible
Value range	INTEGER8
Default value	0

Data description:

Value	**Description**
-128 … -5	Reserved
-4...-1	Methods -1 to -4
0	No homing operation required
1 … 35	Methods 1 to 35
37 … 127	reserved

There are 35 built-in homing methods. Using the MotionPRO Developer software, one can alter each of these homing methods to create a custom one.

6.3.5 Object 6099h: Homing speeds

This object defines the speeds used during homing. It is given in user-defined 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.

By default, the speed values are given in IU and they are of a 16.16 bit structure. To elaborate, see Paragraph 7.2.2 example.

Object description:

Index	6099_h
Name	Homing speeds
Object code	ARRAY
Data type	UNSIGNED32

Entry description:

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

Sub-index	1
Description	Speed during search for switch
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

Sub-index	2
Description	Speed during search for zero
Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	0

6.3.6 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 user-defined acceleration units.

Object description:

Index	$609A_h$
Name	Homing acceleration
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Units	AU
Value range	UNSIGNED32
Default value	-

6.3.7 Object 207Bh: 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 or by object 207Fh. See Paragraph 1.1.5. *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	No

6.3.8 Object 207Ch: 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.

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	No

6.4 Homing example

Execute homing method number 18.

3. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

4. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06_h.

5. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07_h.

6. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

7. **Homing speed during search for zero.** Set the speed during search for zero to 150 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6099_h sub-index 2 expressed in encoder counts per sample is 50000_h.

   Send the following message: SDO access to object 6099_h sub-index 2, 32-bit value 00050000_h.

8. **Homing speed during search for switch.** Set the speed during search for switch to 600 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6099_h sub-index 1 expressed in encoder counts per sample is 140000_h.

   Send the following message: SDO access to object 6099_h sub-index 1, 32-bit value 00140000_h.

9. **Homing acceleration.** The homing acceleration establishes the acceleration to be used with the standard homing moves. Set this value at 5 rot/s^2. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object $609A_h$ expressed in encoder counts per square sample is $28F_h$.

   Send the following message: SDO access to object $609A_h$, 32-bit value $0000028F_h$.

10. **Home offset.** Set the home offset to 1 rotation. By using a 500 lines incremental encoder the corresponding value of object $607C_h$ expressed in encoder counts is $7D0_h$.

    Send the following message: SDO access to object $607C_h$, 32-bit value $000007D0_h$.

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

    Send the following message: SDO access to object 6098_h, 8-bit value 12_h.

12. **Modes of operation.** Select homing mode.

    Set in **Modes of Operation** mapped in RPDO1 the value 06_h.

13. **Start the homing.**

    Set in **Control Word** mapped in RPDO1 the value 001F$_h$.

14. **Press for 5s the LSP button.**

15. **Wait for homing to end.**

    When Status Word (object 6040$_h$) bit13=0, bit12=1 and bit10=1, means homing procedure is completed successfully.

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

Read by SDO protocol the value of object 6064$_h$.

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

7 Position Profile Mode

7.1 Overview

In Position Profile Mode the drive controls the position.

The Position Profile Mode supports 2 motion modes:

- **Trapezoidal profile**. The built-in reference generator computes the position profile with a trapezoidal shape of the speed, due to a limited acceleration. The EtherCAT® master specifies the absolute or relative **Target Position** (index 607A$_h$), the **Profile Velocity** (index 6081$_h$) and the **Profile Acceleration** (6083$_h$)

  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 *Control Word* activates the additive relative mode.

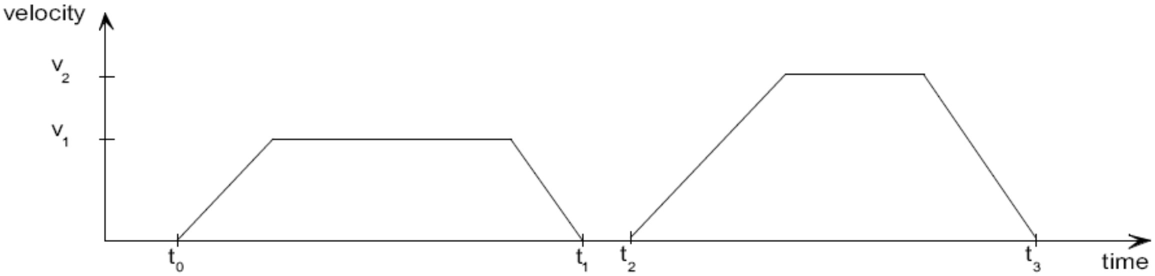
- **S-curve profile.** The built-in reference generator computes a position profile with an S-curve shape of the speed. This shape is due to the jerk limitation, leading to a trapezoidal or triangular profile for the acceleration and an S-curve profile for the speed. The EtherCAT® master specifies the absolute or relative **Target Position** (index 607A$_h$), the **Profile Velocity** (index 6081$_h$), the **Profile Acceleration** (6083$_h$) and the jerk rate. The jerk rate is set indirectly via the **Jerk time** (index 2023$_h$), 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 Control Word:

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

After reaching the *target position* the drive unit signals this status to an EtherCAT® master and then receives a new set-point. After reaching a *target position* the velocity normally is reduced to zero before starting a move to the next set-point.

After the *target position* is sent to the drive, the EtherCAT® master has to set the *new set-point* bit in *controlword*. The drive responds with bit *set-point acknowledge* set in *statusword*. After that, the master has to reset bit *new set-point* to 0. Following this action, the drive will signalize that it can accept a new set-point by resetting *set-point acknowledge* bit in *statusword* after the reference generator has reached the designated demand position.



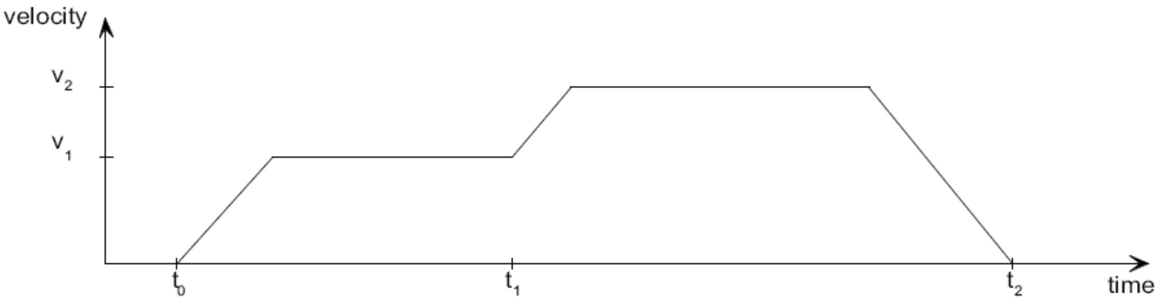
7.1.2 Continuous motion profile (*change set immediately* = 1)

The drive unit immediately processes the next *target position*, even if the actual movement is not completed. The drive readapts the actual move to the new target position.

In this case, the handshake presented for *change set immediately* = 0 is not necessary. By setting the *new set-point* bit, the master will trigger the immediate update of the target position. In this case, if the *target position* is set as relative, also bit 11 is taken into consideration (with or without additive movement).

Remark:

In case object 6086h (Motion Profile Type) is set to 3 (jerk-limited ramp = S-curve profile), then *change set immediately* bit must be 0, else a command error is issued.



7.1.3 Controlword in profile position mode

MSB **LSB**

See 6040h	Operation Mode	See 6040h	Halt	See 6040h	Abs/rel	Change set immediately	New set-point	See 6040h
15 12	11	10 9	8	7	6	5	4	3 0

Table 7. *Control Word 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	Do not assume *target position*
	1	Assume *target position* (update the new motion parameters)
Change set immediately	0	Finish the actual positioning and then start the next positioning
	1	Interrupt the actual positioning and start the next positioning. Valid only for linear ramp profile.
Abs / rel	0	*Target position* is an absolute value
	1	*Target position* is a relative value
Halt	0	Execute positioning
	1	Stop drive with *profile acceleration*

7.1.4 Statusword in profile position mode

MSB **LSB**

See 6041h	Following error	Set-point acknowledge	See 6041h	Target reached	See 6041h
15 14	13	12	11	10	9 0

Table 7.1 *Status Word bits description for Position Profile Mode*

Name	**Value**	**Description**

Target reached	0	Halt = 0: *Target position* not reached
		Halt = 1: Drive decelerates

	1	Halt = 0: *Target position* reached Halt = 1: Velocity of drive is 0
Set-point acknowledge	0	Trajectory generator will accept a new set-point
	1	Trajectory generator will not accept a new set-point.
Following error	0	No following error
	1	Following error

7.2 Position Profile Mode Objects

7.2.1 Object 607Ah: Target position

The *target position* is the position that the drive should move to in position profile mode using the current settings of motion control parameters such as velocity, acceleration, and *motion profile type* etc. It is given in position units.

The position units can be user defined position units (see **Chapter 5 Factor group**).

If Control Word bit 6 = 0 (e.g. absolute positioning), represents the position to reach.

If Control Word bit 6 = 1 (e.g. relative positioning), represents the position displacement to do. When Control Word bit 14 = 0, the new position to reach is computed as: motor actual position (6063h) + displacement. When Control Word bit 14 = 1, the new position to reach is computed as: actual demand position (6062_h) + displacement.

Object description:

Index	$607A_h$
Name	Target position
Object code	VAR
Data type	INTEGER32

Entry description:

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

7.2.2 Object 6081h: Profile velocity

In a position profile, it represents the maximum speed to reach at the end of the acceleration ramp. The *profile velocity* is given in user-defined speed units (see **Chapter 5 Factor group**).

By default the velocity value is given in internal units. They are encoder increments/Sample loop. The default Sample loop is 1ms. The velocity variable is 32 bits long and it receives 16.16 data. The MSB takes the integer part and the LSB takes the factionary part.

Example: for a target speed of 50.00 IU, 0x00320000 must be set in 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	-

7.2.3 Object 6083h: Profile acceleration

In position or speed profiles, represents the acceleration and deceleration rates used to change the speed between 2 levels. The same rate is used when *Quick Stop* or *Disable Operation* commands are received. The *profile acceleration* is given in user-defined acceleration units (see **Chapter 5 Factor group**). 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 7.2.2 example.

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	-

7.2.4 Object 6085h: Quick stop deceleration

The *quick stop deceleration* is the deceleration used to stop the motor if the *Quick stop* function is activated. The value is given in the same physical unit as *profile acceleration* object (6083h). 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 7.2.2 example.

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	-

7.2.5 Object 2023h: Jerk time

In this object you can set the time to use for S-curve profile (jerk-limited ramp set in Object 6086h: Motion profile type)

Object description:

Index	2023$_h$
Name	Jerk time
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

7.2.6 Object 6086h: Motion profile type

Object description:

Index	6086$_h$
Name	Motion profile type
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Possible
Value range	INTEGER16
Default value	0

Data description:

Profile code	**Profile type**
-32768 … -1	Manufacturer specific (reserved)
0	Linear ramp (trapezoidal profile)
1,2	Reserved
3	Jerk-limited ramp (S-curve)
4 … 32767	Reserved

7.2.7 Object 6062h: Position demand value

This object represents the output of the trajectory generation. The *position demand value is* shown in position units.

Object description:

Index	6062_h
Name	Position demand value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	-2^{31} … 2^{31}-1
Default value	-

7.2.8 Object 6063h: Position actual internal value

This object represents the actual value of the position measurement device in internal units. It can be used as an alternative to *position actual value* (6064h).

Object description:

Index	6063_h
Name	Position actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Units	increments
Value range	-2^{31} … 2^{31}-1
Default value	-

7.2.9 Object 6064h: Position actual value

This object represents the actual value of the position measurement device. The *position actual value* is shown in user-defined position units.

Remark: when using a stepper open loop with encoder on motor configuration (for step loss detection), a position value will not be reported.

Object description:

Index	6064_h
Name	Position actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Yes
Value range	-2^{31} … 2^{31}-1
Default value	-

7.2.10 Object 6065h: Following error window

This object defines a range of tolerated position values symmetrically to the *position demand value*, expressed in position units. If the *position actual value* is above the *following error window* for a period larger than the one defined in *following error time out*, a following error occurs. A following error may occur when a drive is blocked, unreachable profile velocity occurs, or at wrong closed-loop coefficients. If the value of the *following error window* is set larger or equal to 7FFFh (32767), the following control is switched off.

The maximum value allowed for the *following error window* parameter, expressed in increments, is 32767. When this object is written with a higher corresponding value, the *following error window* parameter will be set to its maximum value (32767).

Object description:

Index	6065_h
Name	Following error window
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED32
Default value	-

7.2.11 Object 6066h: Following error time out

This object indicates the configured time for a following error condition, after that the bit

13 of the statusword shall be set to 1. The value is given in ms. Also see 6065h, *following error window*.

Object description:

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

Entry description:

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

7.2.12 Object 6067h: Position window

The *position window* defines a symmetrical range of accepted positions relative to the *target position*. If the *position actual value* is within the *position window* for a time period defined inside the *position window time* object, this *target position* is regarded as reached. The *position window* is given in user-defined position units.

The maximum value allowed for the *position window* parameter, is 32767. When this object is written with a higher corresponding value, the *position window* parameter will be set to its maximum value (32767).

If 0xFFFFFFFF is written, the object will be read as 0xFFFFFFFF and the position window control will be switched off and object 6068h will be set automatically to 0xFFFF also. When the position window control is turned off, Target reached, Bit10 of Status Word, will be set when the position reference reaches the target position.

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	-

7.2.13 Object 6068h: Position window time

This object indicates the configured time, during which the actual position within the position window is measured. The value shall be given in ms. Also see description of object 6067h, *position window*.

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	-

7.2.14 Object 60F4h: Following error actual value

This object represents the actual value of the following error, given in user-defined position units.

Object description:

Index	$60F4_h$
Name	Following error actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

7.2.15 Object 60FCh: Position demand internal value

This output of the trajectory generator in profile position mode is an internal value using increments as unit. It can be used as an alternative to *position demand value* (6062h).

Object description:

Index	$60FC_h$
Name	Position demand internal value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Units	Increments
Value range	-2^{31} … $2^{31}-1$
Default value	-

7.2.16 Object 2022h: Control effort

This object can be used to visualize the control effort of the drive (the reference for the current controller). It is available in internal units.

Object description:

Index	2022_h
Name	Control effort
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER16
Default value	-

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

This object sets the motor position to the value specified.

Object description:

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

Entry description:

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

7.2.18 Object 2088h[1]: Actual internal position from sensor on motor

This object shows the position value read from the encoder on the motor in increments, in case a dual loop control method is used.

The factor group objects have no effect on it.

Object description:

Index	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	$-2^{31} \dots 2^{31}-1$
Default value	-

7.2.19 Object 208Dh[2]: Auxiliary encoder position

This object represents the actual value of the auxiliary position measurement device in internal units. 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	$-2^{31} \dots 2^{31}-1$
Default value	-

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

[2] Object 208Dh is available only drives which have a secondary feedback input

7.3 Position Profile Example

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

1. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

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

   Set in **Control Word** mapped in RPDO1 the value 06_h.

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

   Set in **Control Word** mapped in RPDO1 the value 07_h.

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

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

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

   Set in **Modes of Operation** mapped in RPDO1 the value 01_h.

6. **Target position.** Set the target position to 4 rotations. By using a 500 lines incremental encoder the corresponding value of object $607A_h$ expressed in encoder counts is $1F40_h$.

   Set in **Target position** mapped in RPDO2 the value $00001F40_h$.

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 $10AAAC_h$.

   Send the following message: SDO access to object 6081_h, 32-bit value $0010AAAC_h$.

8. **Start the profile.**

   Set in **Control Word** mapped in RPDO1 the value $001F_h$.

9. **Wait movement to finish.**

   Wait for Bit10 to become 1 in Status Word.

10. **Reset the set point.**

    Set in **Control Word** mapped in RPDO1 the value $000F_h$.

11. **Target position.** Set the target position to 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 $00007D00_h$.

12. **Start the profile.**

    Set in **Control Word** mapped in RPDO1 the value 001F$_h$.

13. **Wait movement to finish.**

    Wait for Bit10 to become 1 in Status Word.

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

    Read by SDO protocol the value of object 6064$_h$.

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

    Read by SDO protocol the value of object 6062$_h$.

At the end of movement the motor position actual value should be equal with position demand value (plus or minus few encoder counts depending on your position tuning) and the motor should rotate 16 times.

Absolute Jerk-limited(S-curve) Profile Example

Example: Execute an absolute Jerk-limited ramp profile.

1. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

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

   Set in **Control Word** mapped in RPDO1 the value 06_h.

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

   Set in **Control Word** mapped in RPDO1 the value 07_h.

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

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

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

   Set in **Modes of Operation** mapped in RPDO1 the value 01_h.

6. **Motion profile type.** Select Jerk-limited ramp.

   Send the following message: SDO access to object 6086_h, 16-bit value 0003_h.

7. **Target position.** Set the target position to 5 rotations. By using a 500 lines incremental encoder the corresponding value of object $607A_h$ expressed in encoder counts is 2710_h.

   Set in **Target position** mapped in RPDO2 the value 00002710_h.

8. **Target speed.** Set the target speed to 150 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object 6081_h expressed in encoder counts per sample is 00050000_h.

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

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

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

10. **Start the profile.**

    Set in **Control Word** mapped in RPDO1 the value $001F_h$.

11. **Wait movement to finish.**

    Wait for Bit10 to become 1 in Status Word.

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

    Read by SDO protocol the value of object 6064_h.

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

Read by SDO protocol the value of object 6062_h.

At the end of movement the motor position actual value should be equal with position demand value (plus or minus few encoder counts depending on your position tuning).

8 Interpolated Position Mode

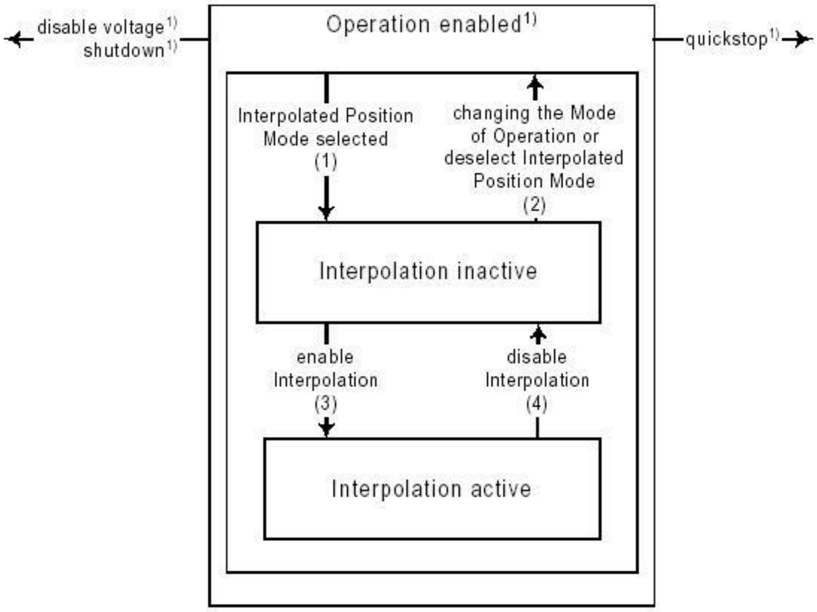
8.1 Overview

The interpolated Position Mode is used to control multiple coordinated axis or a single on with the need for time-interpolation of set-point data. The Interpolated Position Mode can use the time synchronization mechanism for a time coordination of the related drive units.

The Interpolated Position Mode allows a host controller to transmit a stream of interpolation data to a drive unit. The interpolation data is better sent in bursts because the drive supports an input buffer. The buffer size is the number of *interpolation data records* that may be sent to the drive to fill the input buffer.

The interpolation algorithm can be defined in the *interpolation sub mode select*. Linear (PT – Position Time) interpolation is the default interpolation method.

8.1.1 Internal States



disable voltage1)
shutdown1)
Operation enabled1)
quickstop1)
Interpolated Position Mode selected (1)
changing the Mode of Operation or deselect Interpolated Position Mode (2)
Interpolation inactive
enable Interpolation (3)
disable Interpolation (4)
Interpolation active

Figure 8.1 *Internal States for the Interpolated Position Mode*

$^{1)}$ See state machine

***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

8.1.2 Controlword in interpolated position mode

MSB **LSB**

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

Table 8.1 *Control Word bits description for Interpolated Position Mode*

Name	Value	Description
Enable ip mode	0	Interpolated position mode inactive
	1	Interpolated position mode active
Abs / rel	0	Set position is an absolute value[1]
	1	Set position is a relative value[1]
Halt	0	Execute the instruction of bit 4
	1	Stop drive with (*profile acceleration*)
Stop option	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.

8.1.3 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

[1] This bit must be set before loading the interpolation points in the buffer

Table 8.2 *Status Word 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

8.2 Interpolated Position Objects

8.2.1 Object 60C0h: Interpolation sub mode select

In the Interpolated Position Mode the drive supports two interpolation modes: PT (Position – Time) linear interpolation and PVT (Position – Velocity – Time) cubic interpolation. 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	60C0$_h$
Name	Interpolation sub mode select
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Possible
Value range	-2^{15} … $2^{15}-1$
Default value	0

Data description:

Profile code	Profile type
-32768 … -2	Manufacturer specific (reserved)
-1	PVT (Position – Velocity – Time) cubic interpolation
0	PT (Position – Time) Linear Interpolation
+1…+32767	Reserved

8.2.2 Object 60C1h: Interpolation data record

The **Interpolation Data Record** contains the data words that are necessary to perform the interpolation algorithm. The number of data words in the record is defined by the *interpolation data configuration.*

Object description:

Index	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:

a) For PVT (Position – Velocity – Time) cubic interpolation:

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

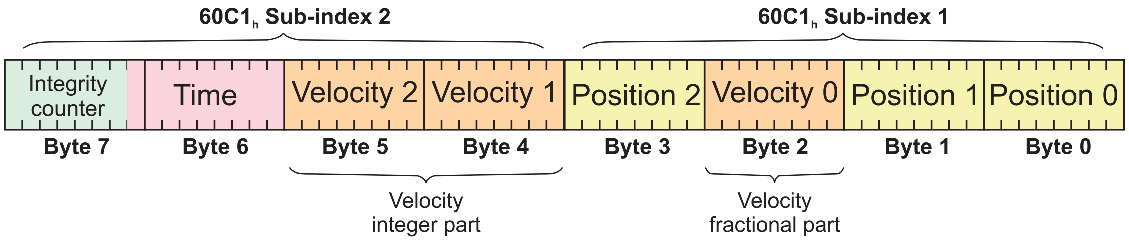


Figure 8.2 *PVT interpolation point 64-bit data structure*

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

b) For PT (Position –Time) linear interpolation:

There are 3 parameters in this mode:

Position – a 32-bit long integer value representing the target position (relative or absolute). Unit - position increments.

Time – a 16-bit unsigned integer value representing the time of a PT segment. Unit - position / speed loop samplings.

Counter – a 7-bit unsigned integer value representing an integrity counter. It can be used in order to have a feedback of the last point sent to the drive and detect errors in transmission.

In the example below Position[7…0] represents bits 0..7 of the position value.

Byte 0	Position [7...0]	
Byte 1	Position [15...8]	
Byte 2	Position [23...16]	
Byte 3	Position [31...24]	
Byte 4	Time [7...0][1]	
Byte 5	Time [15...8][1]	
Byte 6	Reserved	
Byte 7	Counter[6…0]	Reserved

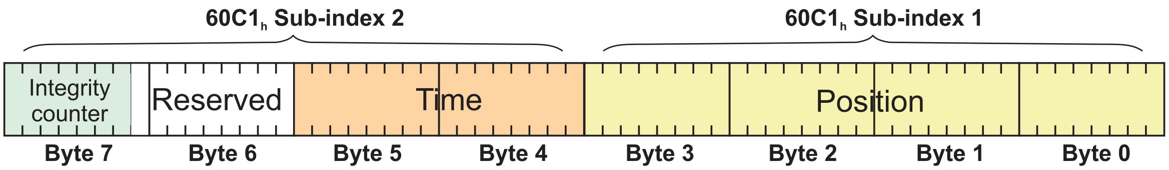


Figure 8.3 *PT interpolation point 64-bit data structure*

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

8.2.3 Object 60C2h: Interpolation time period

The **Interpolation time period** indicates the configured interpolation cycle time. Its value must be set with the one of the EtherCAT master communication cycle time in order for the CSP, CSV and CST modes to work properly. The interpolation time period (sub-index 01_h) value is given in $10^{(\text{interpolation time index})}$ s(second). The interpolation time index (sub-index 02_h) is dimensionless.

Example: to set a communication cycle time of 4ms, $60C2_h$ sub-index 01_h = 4 and $60C2_h$ sub-index 02_h = -3. The result is 4ms = $4*10^{-3}$.

Object description:

Index	$60C2_h$
Name	Interpolation time period
Object code	ARRAY
Number of elements	2
Data Type	Interpolation time period record

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

Entry description:

Sub-index	00_h
Description	Number of sub-indexes
Access	RO
PDO mapping	No
Default value	2

Sub-index	01_h
Description	Interpolation time period value
Access	RW
PDO mapping	Possible
Value range	Unsigned8
Default value	1

Sub-index	02_h
Description	Interpolation time index
Access	RW
PDO mapping	Possible
Value range	INTEGER8, (-128 to +63)
Default value	-3

8.2.4 Object 2072h: Interpolated position mode status

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

Object description:

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

Entry description:

Access	RO
PDO mapping	Possible
Value range	UNSIGNED16
Default value	-

Table 8.3 *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 2074_h.
13	0	Buffer is not full
	1	Buffer is full – the number of points in the buffer is equal with the buffer dimension.
12	0	No integrity counter error
	1	Integrity counter error. If integrity counter error checking is enabled and the integrity counter sent by the master does not match the integrity counter of the drive.
11	0	Valid only for PVT (cubic interpolation): Drive has maintained interpolated position mode after a buffer empty condition (the velocity of the last point was 0).
	1	Valid only for PVT (cubic interpolation): Drive has performed a quick stop after a buffer empty condition because the velocity of the last point was different from 0
10 7		Reserved
6 0		Current integrity counter value

Remark: *when a status bit changes from this object, an emergency message with the code 0xFF01 will be generated. This emergency message will have mapped object 2072h data onto bytes 3 and 4.*

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

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

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

8.2.5 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	2073_h
Name	Interpolated position buffer length
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	1-15
Default value	7

8.2.6 Object 2074h: Interpolated position buffer configuration

Through this object you can control more in detail the behavior of the buffer.

Note: The integrity counter is always enabled.

Object description:

Index	2074_h
Name	Interpolated position buffer configuration
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	-

Table 8.4 *Interpolated position buffer configuration*

Bit	Value	Description
15	0	Nothing
	1	Clear buffer and reinitialize buffer internal variables
14		Reserved.
13	0	No change in the integral integrity counter
	1	Change internal integrity counter with the value specified in bits 0 to 6
12	0	If absolute positioning is set (bit 6 of *controlword* is 0), the initial position is read from object 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

8.2.7 Object 2079h: Interpolated position initial position

Through this object you can set an initial position for absolute positioning in order to be used to compute the distance to move up to the first point. It is given in position units.

Object description:

Index	2079$_h$
Name	Interpolated position initial position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Yes
Value range	INTEGER32
Default value	0

8.2.8 Object 207Ah: Interpolated position 1st order time

Through this object you can set the time in a PT (Position – Time) Linear Interpolation mode. By setting a value in this object, there is no need to send the time together with the position and integrity counter in object **60C1h**. 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

8.2.9 Loading the interpolated points

The integrity counter is always 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 as is the case with cyclic RxPDO data, it will ignore the point without giving an error. 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 2072h: Interpolated position mode status 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. So 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 Control Word.

A correct interpolated PT/PVT motion would be like this:

- Enter mode 07 in Modes of Operation
- set the IP buffer size
- Clear the buffer and reinitialize the integrity counter
- Set in controlword the bit for absolute or relative motion
- If motion is absolute, set in 2079h the actual position of the drive (read from object 6063h)
- If the motion is PT, set in object 207Ah a fixed time interval if not supplied in 60C1 subindex2
- 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
- Load more points until buffer full bit is active
- Return to monitoring the buffer status and so on

8.3 PT absolute movement example

Execute a PT movement.

1. **Start remote node**.

Enter **Pre-Operational** state.

2. **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.

3. **Map the new objects**:
   a. 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 1602_h sub-index 1, 32-bit value $60C10120_h$.

   b. Write in object 1602_h sub-index 2 the description of the interpolated data record sub-index 2:

   Send the following message: SDO access to object 1602_h sub-index 2, 32-bit value $60C10220_h$.

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

Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 2.

5. **Add the new TPDO to the Sync Manager**:
   a. Write zero in object $1C12_h$ sub-index 0, this will disable the Sync. Manager.

   Send the following message: SDO access to object $1C12_h$ sub-index 0, 8-bit value 00_h.

   b. Write in object $1C12_h$ sub-index 3 the RPDO3 mapping parameter object number:

   Send the following message: SDO access to object $1C12_h$ sub-index 3, 16-bit value 1602_h.

   c. Write 03_h in object $1C12_h$ sub-index 0, this will enable the Sync. Manager.

   Send the following message: SDO access to object $1C12_h$ sub-index 0, 8-bit value 03_h.

Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new DO configuration. Press F4 to reload the IO devices and enter in Operation state.

6. Enter **Safe-Operational** state.
7. Enter **Operational** state.
8. **Ready to switch on.** Set in **Control Word** mapped in RPDO1 the value 06_h.
9. **Switch on.** Set in **Control Word** mapped in RPDO1 the value 07_h.
10. **Enable Operation.** Set in **Control Word** mapped in RPDO1 the value $0F_h$. For relative motion, set $4F_h$.
11. Set in **Modes of operation** mapped in RPDO1 the value 7 to enable Interpolated mode.
12. **Interpolation sub mode select**. Select PT interpolation position mode.

Send the following message: SDO access to object 60C0$_h$, 16-bit value 0000$_h$.

13. **Interpolated position buffer length**.

    Send the following message: SDO access to object 2073$_h$, 16-bit value 000C$_h$. The maximum is 000F$_h$.

14. **Interpolated position buffer configuration**. By setting the value A001$_h$, the buffer is cleared and the integrity counter will be set to 1.

    Send the following message: SDO access to object 2074$_h$, 16-bit value A001$_h$.

15. **Interpolated position initial position**. Set the initial position to 0.5 rotations. By using a 500 lines incremental encoder the corresponding value of object 2079$_h$ expressed in encoder counts is (1000$_d$) 3E8$_h$. 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 2079$_h$, 32-bit value 3E8$_h$.

16. **Loading the PT points.** Assuming X1 and X2 are 60C1 sub index 01 and 02 which were recently mapped, send the following data:

17. **Send the 1st PT point**.

    Position= 20000 IU (0x00004E20)   1IU = 1 encoder pulse

    Time = 1000 IU (0x03E8)   1IU = 1 control loop = 1ms by default

    IC   = 1 (0x01)   IC=Integrity Counter

The drive motor will do 10 rotations (20000 counts) in 1000 milliseconds.

Set X1=00004E20$_h$; X2=020003E8 $_h$;

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

    Position= 30000 IU (0x00007530)

    Time = 2000 IU (0x07D0)

    IC   = 2 (0x02)

Set X1=00007530$_h$; X2=040007D0 $_h$;

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

    Position= 2000 IU (0x000007D0)

    Time = 1000 IU (0x03E8)

    IC   = 3 (0x03)

Set X1=000007D0$_h$; X2=060003E8 $_h$;

20. **Send the last PT point**.

Set X1=00000000 $_h$ (0 counts); X2=080001F4 (IC=4 (0x08), time =500 (0x01F4))

    Position= 0 IU (0x00000000)

    Time = 500 IU (0x01F4)

    IC   = 4 (0x04)

Set X1=00000000h; X2=080001F4h;

21. **Start an absolute motion**.

    Set in **Control Word** mapped in RPDO1 the value 1Fh.

After the sequences are executed, if the drive actual position before starting the motion was 0, now it should be -1000 counts because of Step 15.

8.4 PVT absolute movement example

Execute an absolute PVT movement.

1. **Start remote node**.

   Enter **Pre-Operational** state.

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

   Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 0.

3. **Map the new objects**:

   - Write in object 1602h sub-index 1 the description of the interpolated data record sub-index 1:

   Send the following message: SDO access to object 1602h sub-index 1, 32-bit value 60C10120h.

   - Write in object 1602h sub-index 2 the description of the interpolated data record sub-index 2:

   Send the following message: SDO access to object 1602h sub-index 2, 32-bit value 60C10220h.

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

   Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 2.

5. **Add the new TPDO to the Sync Manager**:

   - Write zero in object 1C12h sub-index 0, this will disable the Sync. Manager.

   Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 00h.

   - Write in object 1C12h sub-index 3 the RPDO3 mapping parameter object number:

   Send the following message: SDO access to object 1C12h sub-index 3, 16-bit value 1602h.

   - Write 03 h in object 1C12h sub-index 0, this will enable the Sync. Manager.

   Send the following message: SDO access to object 1C12h sub-index 0, 8-bit value 03h.

   Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new PDO configuration. Press F4 to reload the IO devices and enter in Operation state.

6. Enter **Safe-Operational** state.

7. Enter **Operational** state.

8. **Ready to switch on.** Set in **Control Word** mapped in RPDO1 the value 06_h.

9. **Switch on.** Set in **Control Word** mapped in RPDO1 the value 07_h.

10. **Enable Operation and set an absolute motion.** Set in **Control Word** mapped in RPDO1 the value $0F_h$.

11. Set in **Modes of operation** mapped in RPDO1 the value 7 to enable Interpolated mode.

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

    Send the following message: SDO access to object $60C0_h$, 16-bit value $FFFF_h$.

13. **Interpolated position buffer length**.

    Send the following message: SDO access to object 2073_h, 16-bit value $000F_h$. The maximum is $000F_h$.

14. **Interpolated position buffer configuration**. By setting the value $B001_h$, the buffer is cleared and the integrity counter will be set to 1.

    Send the following message: SDO access to object 2074_h, 16-bit value $B001_h$.

15. **Loading the PVT points.** Assuming X1 and X2 are 60C1 sub index 01 and 02 which were recently mapped, send the following data:

16. **Send the 1st PVT point**.

    Position = 88 IU (0x000058) 1IU = 1 encoder pulse

    Velocity = 3.33 IU (0x000354) 1IU = 1 encoder pulse/ 1 control loop

    Time = 55 IU (0x37) 1IU = 1 control loop = 1ms by default

    IC = 1 (0x01) IC=Integrity Counter

Set X1=00540058_h; X2=02370003_h;

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

    Position = 370 IU (0x000172)

    Velocity = 6.66 IU (0x0006A8)

    Time = 55 IU (0x37)

    IC = 2 (0x02)

Set X1=$00A80172_h$; X2=04370006_h;

18. Send the 3rd PVT point.

Position = 2982 IU (0x000BA6)

Velocity = 6.66 IU (0x0006A8)

Time = 390 IU (0x186)

IC = 3 (0x03)

Set X1=00A80BA6$_h$; X2=07860006$_h$;

19. Send the 4th PVT point.

Position = 5631 IU (0x0015FF)

Velocity = 6.66 IU (0x0006A8)

Time = 400 IU (0x190)

IC = 4 (0x04)

Set X1=00A815FF$_h$; X2=09900006$_h$;

20. Send the 5th PVT point.

Position = 5925 IU (0x001725)

Velocity = 3.00 IU (0x000300)

Time = 60 IU (0x3C)

IC = 5 (0x05)

Set X1=00001725$_h$; X2=0A3C0003$_h$;

21. Send the 6th PVT point.

Position = 6000 IU (0x001770)

Velocity = 0.00 IU (0x000000)

Time = 50 IU (0x32)

IC = 6 (0x06)

Set X1=00001770$_h$; X2=0C320000$_h$;

22. Send the 7th PVT point.

Position = 5127 IU (0x001407)

Velocity = -7.5 IU (0xFFF880)

Time = 240 IU (0xF0)

IC = 7 (0x07)

Set X1=00801407$_h$; X2=0EF0 FFF8$_h$;

23. Send the 8th PVT point.

Position = 3115 IU (0x000C2B)

Velocity = -13.33 IU (0xFFF2AB)

Time = 190 IU (0xBE)

IC = 8 (0x08)

Set X1=00AB0C2B $_h$; X2=10BEFFF2$_h$;

24. Send the 9th PVT point.

Position = -1686 IU (0xFFF96A)

Velocity = -13.33 IU (0xFFF2AB)

Time = 360 IU (0x168)

IC = 9 (0x09)

Set X1=FFABF96A $_h$; X2=1368FFF2$_h$;

25. Send the 10nth PVT point.

Position = -7145 IU (0xFFE417)

Velocity = -13.33 IU (0xFFF2AB)

Time = 410 IU (0x19A)

IC = 10 (0x0A)

Set X1=FFABE417 $_h$; X2=159AFFF2$_h$;

26. Send the 11th PVT point.

Position = -9135 IU (0xFFDC51)

Velocity = -7.4 IU (0xFFF899)

Time = 190 IU (0xBE)

IC = 11 (0x0B)

Set X1=FF990C2B $_h$; X2=16BEFFF8$_h$;

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

Position = -10000 IU (0xFFD8F0)

Velocity = -7.4 IU (0x000000)

Time = 240 IU (0xF0)

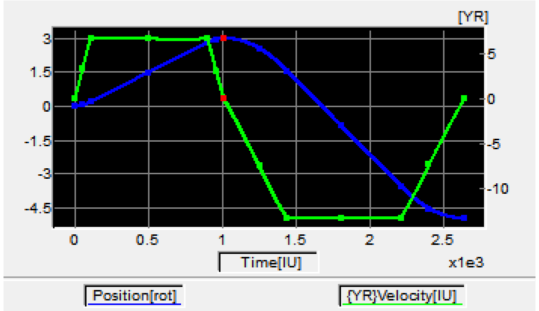
IC = 12 (0x0C)

Set X1=FF00D8F0 $_h$; X2=18F00000$_h$;

28. Start an absolute PVT motion.

Set in **Control Word** mapped in RPDO1 the value 1F$_h$.

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.

8.5 PVT relative movement example

Execute a relative PVT movement.

1. **Start remote node**.

   Enter **Pre-Operational** state.

2. **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.

3. **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 1602$_h$ sub-index 1, 32-bit value 60C10120$_h$.

   - Write in object 1602$_h$ sub-index 2 the description of the interpolated data record sub-index 2:

     Send the following message: SDO access to object 1602$_h$ sub-index 2, 32-bit value 60C10220$_h$.

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

   Send the following message: SDO access to object 1602h sub-index 0, 8-bit value 2.

5. **Add the new TPDO to the Sync Manager**:
   - Write zero in object 1C12$_h$ sub-index 0, this will disable the Sync. Manager.

   Send the following message: SDO access to object 1C12$_h$ sub-index 0, 8-bit value 00$_h$.

   - Write in object 1C12$_h$ sub-index 3 the RPDO3 mapping parameter object number:

   Send the following message: SDO access to object 1C12$_h$ sub-index 3, 16-bit value 1602$_h$.

   - Write 03$_h$ in object 1C12$_h$ sub-index 0, this will enable the Sync. Manager.

   Send the following message: SDO access to object 1C12$_h$ sub-index 0, 8-bit value 03$_h$.

   Note: if using TwinCAT System Manager, enter in Configuration Mode, select the drive, select Process Data tab, uncheck the PDO Assignment and PDO Configuration boxes. Click Load PDO info from device button to load the new PDO configuration. Press F4 to reload the IO devices and enter in Operation state.

6. Enter **Safe-Operational** state.
7. Enter **Operational** state.
8. **Ready to switch on.** Set in **Control Word** mapped in RPDO1 the value 06$_h$.
9. **Switch on.** Set in **Control Word** mapped in RPDO1 the value 07$_h$.
10. **Enable Operation and set a relative motion.** Set in **Control Word** mapped in RPDO1 the value 4F$_h$. For an absolute motion, set 0F$_h$ but the example points will not apply.
11. Set in **Modes of operation** mapped in RPDO1 the value 7 to enable Interpolated mode.
12. **Interpolation sub mode select**. Select PVT interpolation position mode.

    Send the following message: SDO access to object 60C0$_h$, 16-bit value FFFF$_h$.

13. **Interpolated position buffer length**.

    Send the following message: SDO access to object 2073$_h$, 16-bit value 000C$_h$. The maximum is 000F$_h$.

14. **Interpolated position buffer configuration**. By setting the value A001$_h$, the buffer is cleared and the integrity counter will be set to 1.

    Send the following message: SDO access to object 2074$_h$, 16-bit value A001$_h$.

15. **Loading the PVT points.** Assuming X1 and X2 are 60C1 sub index 01 and 02 which were recently mapped, send the following data:
16. **Send the 1st PVT point**.

    Position = 400 IU (0x000190) 1IU = 1 encoder pulse

    Velocity = 3.00 IU (0x000300) 1IU = 1 encoder pulse/ 1 control loop

    Time = 250 IU (0xFA) 1IU = 1 control loop = 1ms by default

    IC = 1 (0x01) IC=Integrity Counter

Set X1=00000190$_h$; X2=02FA0003$_h$;

17. Send the 2nd PVT point.

Position = 1240 IU (0x0004D8)

Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 2 (0x02)

Set X1=000004D8$_h$; X2=04FA0006$_h$;

18. Send the 3rd PVT point.

Position = 1674 IU (0x00068A)

Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 3 (0x03)

Set X1=0000068A$_h$; X2=06FA0006$_h$;

19. Send the 4th PVT point.

Position = 1666 IU (0x000682)

Velocity = 6.00 IU (0x000600)

Time = 250 IU (0xFA)

IC = 4 (0x04)

Set X1=00000682$_h$; X2=08FA0006$_h$;

20. Send the 5th PVT point.

Position = 1240 IU (0x0004D8)

Velocity = 3.00 IU (0x000300)

Time = 250 IU (0xFA)

IC = 5 (0x05)

Set X1=000004D8$_h$; X2=0AFA0003$_h$;

21. Send the last PVT point.

Position = 410 IU (0x00019A)

Velocity = 0.00 IU (0x000000)

Time = 250 IU (0xFA)

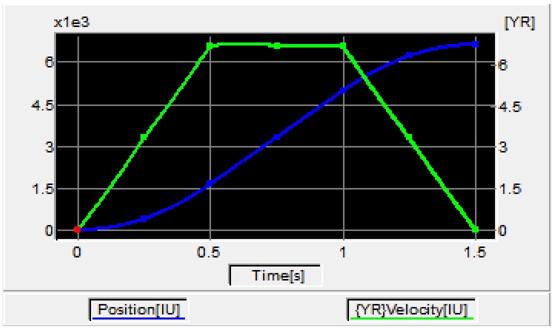
IC = 6 (0x06)

Set X1=0000019A $_h$; X2=0CFA0000$_h$;

22. Start a relative PVT motion.

Set in **Control Word** mapped in RPDO1 the value $5F_h$.

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.

9 Velocity Profile Mode

9.1 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 60FF$_h$) specifies the jog speed (speed sign specifies the direction) and the **Profile Acceleration** object (index 6083$_h$) the acceleration/deceleration rate. While the mode is active, any change of the Target Velocity object by the EtherCAT® master will update the drive's demand velocity enabling you to change on the fly the slew speed and/or the acceleration/deceleration rate. The motion will continue until the **Halt** bit from the Control Word 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.

Remark: If the velocity is already set when entering velocity mode, the motion will not start until a value (even if it is the same) will be set again in Target Velocity 60FFh.

9.1.1 Controlword in profile velocity mode

MSB **LSB**

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

Table 9.1 *Control Word bits for Velocity Profile Mode*

Name	Value	Description
Halt	0	Execute the motion
	1	Stop drive with *profile acceleration*

9.1.2 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 9.2 *Statusword bits for Velocity Profile*

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 606F$_h$, 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.*

9.2 Velocity Mode Objects

9.2.1 Object 6069h: Velocity sensor actual value

This object describes the velocity value read from the load encoder in increments. 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. To elaborate, see Paragraph 7.2.2 example.

Object description:

Index	6069$_h$
Name	Velocity sensor actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

9.2.2 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. 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 7.2.2 example.

Object description:

Index	$606B_h$
Name	Velocity demand value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

9.2.3 Object 606Ch: Velocity actual value

The *velocity actual value* is given in velocity units and is read from the velocity sensor. 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 7.2.2 example.

Object description:

Index	$606C_h$
Name	Velocity actual value
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Yes
Value range	INTEGER32
Default value	-

9.2.4 Object 606Fh: Velocity threshold

The *velocity threshold* is given in 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.

Object description:

Index	606F$_h$
Name	Velocity threshold
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	-

9.2.5 Object 60FFh: Target velocity

The *target velocity* is the input for the trajectory generator and the value is given in user-defined velocity units. 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 7.2.2 example.

Object description:

Index	60FF$_h$
Name	Target velocity
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	possible
Value range	INTEGER32
Default value	-

9.2.6 Object 60F8h: Max slippage

The *max slippage* monitors whether the maximal slippage of an asynchronous motor has actually been reached. The value is given in user-defined velocity units. When the *max slippage* has been reached, the corresponding bit 13 *max slippage error* in the *statusword* is set to 1. 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 7.2.2 example.

Object description:

Index	60F8$_h$
Name	Max slippage
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	possible
Value range	INTEGER32
Default value	-

9.2.7 Object 2005h: Max slippage time out

Time interval for

Object 60F8h: Max slippage. The value is given in ms.

Object description:

Index	2005_h
Name	Max slippage time out
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Possible
Value range	UNSIGNED16
Default value	-

9.2.8 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. To elaborate, see **Paragraph 7.2.2** example.

Object description:

Index	2087_h
Name	Actual internal velocity sensor on motor
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	INTEGER32
Default value	-

9.3 Velocity profile example[2]

Execute a speed control with 600 rpm target speed.

1. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

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

   Set in **Control Word** mapped in RPDO1 the value 06_h.

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

   Set in **Control Word** mapped in RPDO1 the value 07_h.

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

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

[1] Object 2087h applies only to drives which have a secondary feedback

[2] In order to run this example, the speed loop must be activated in PRO Config.

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

   Set in **Modes of Operation** mapped in RPDO1 the value 03_h.

6. **Target velocity.** Set the target velocity to 600 rpm. By using a 500 lines incremental encoder and 1ms sample rate for position/speed control the corresponding value of object $60FF_h$ expressed in encoder counts per sample is 140000_h (20.0 IU).

   Send the following message: SDO access to object $60FF_h$ 32-bit value 00140000_h.

7. **Check the motor actual speed.** It should rotate with 600 rpm.
   Read by SDO protocol the value of object $606C_h$.

10 Electronic Gearing Position (EGEAR) Mode

10.1 Overview

In Electronic Gearing Position Mode the drive follows the position of an electronic gearing master with a programmable gear ratio.

The electronic gearing slave can get the position information from the electronic camming master in three ways:

1. Via EtherCAT® master, which writes the master position in object **Master position** (index 201E$_h$).
2. Via an external digital reference[1] of type pulse & direction or quadrature encoder. Both options have dedicated inputs. The pulse & direction signals are usually provided by an indexer and must be connected to the pulse & direction inputs of the drive. The quadrature encoder signals are usually provided by an encoder on the master and must be connected to the 2nd encoder inputs.
3. From one of the analogue inputs of the drive.

The reference type, i.e. the selection between the online reference received via communication channel and the digital reference read from dedicated inputs is done with object **External Reference Type** (index 201D$_h$). The source of the digital reference (pulse & direction or second encoder inputs) is set during drive commissioning.

The drive set as slave in electronic gearing mode performs a position control. At each slow loop sampling period, the slave computes the master position increment and multiplies it with its programmed gear ratio. The result is the slave position reference increment, which added to the previous slave position reference gives the new slave position reference.

Remark: *The slave executes a relative move, which starts from its actual position*

The gear ratio is specified via **EGEAR multiplication factor** object (index 2013$_h$). EGEAR ratio numerator (sub-index 1) is a signed integer, while EGEAR ratio denominator (sub-index 2) is an unsigned integer. The EGEAR ratio numerator sign indicates the direction of movement: positive – same as the master, negative – reversed to the master. The result of the division between EGEAR ratio numerator and EGEAR ratio denominator is used to compute the slave reference increment.

The **Master Resolution** object (index 2012$_h$) provides the master resolution, which is needed to compute correctly the master position and speed (i.e. the position increment). If master position is not cyclic (i.e. the resolution is equal with the whole 32-bit range of position), set master resolution to 0x80000001.

You can smooth the slave coupling with the master, by limiting the maximum acceleration of the slave drive. This is particularly useful when the slave has to couple with a master running at high speed, in order to minimize the shocks in the slave. The feature is activated by setting ControlWord.5=1 and the maximum acceleration value in **Profile Acceleration** object (index 6083$_h$).

[1] Not all drives have a secondary encoder input.

10.1.1 Controlword in electronic gearing position mode (slave axis)

MSB **LSB**

See 6040h	Halt	See 6040h	Reserved	Activate Acceleration Limitation	Enable Electronic Gearing Mode	See 6040h
15 9	8	7	6	5	4	3 0

Table 10.1 *Control Word bits for Electronic Gearing Position Mode*

Name	**Bit**	**Value**	**Description**
Enable Electronic Gearing Mode	4	0	Do not start operation
		0 -> 1	Start electronic gearing procedure
		1	Electronic gearing mode active
		1 -> 0	Do nothing (does not stop current procedure)
Activate Acceleration Limitation	5	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	8	0	Execute the instruction of bit 4
		1	Stop electronic gearing motion with the *profile acceleration.*

10.1.2 Statusword in electronic gearing position mode

MSB **LSB**

See 6041h	Following error	Reserved	See 6041h	Target reached	See 6041h
15 14	13	12	11	10	9 0

Table 10.2 *Status Word bits for Electronic Gearing Position Mode*

Name	**Bit**	**Value**	**Description**

Target reached	10	0	Halt = 0: Always 0
			Halt = 1: Drive decelerates

		1	Halt = 0: Always 0 Halt = 1: Velocity of drive is 0
Following error	13	0	No following error
		1	Following error occurred

10.2 Gearing Position Mode Objects

10.2.1 Object 201Eh: Master position

This object is used in order to receive the position from the master, which is used for Electronic Gearing or Camming calculations. The position units are in increments.

Example: if it takes 4000 increments for the motor to do one revolution, these same increments apply for this object.

Object description:

Index	$201E_h$
Name	Master position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RW
PDO mapping	Possible
Units	Increments
Value range	$0 \dots 2^{31}-1$
Default value	-

10.2.2 Object 2012h: Master resolution

This object is used in order to set the master resolution in increments per revolution. This object is valid for the slave axis.

Object description:

Index	2012_h
Name	Master resolution
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	Possible
Units	Increments
Value range	$0 \dots 2^{31}-1$
Default value	80000001h (full range)

10.2.3 Object 2013h: EGEAR multiplication factor

In digital external mode, this object sets the gear ratio, or gear multiplication factor for the slaves. The sign indicates the direction of movement: positive – same as the master, negative – reversed to the master. The slave demand position is computed as the master position increment multiplied by the gear multiplication factor.

Example: if the gear ratio is Slave/Master = 1/3, the following values must be set: 1 in EGEAR ratio numerator (sub-index 1) and 3 in EGEAR ratio denominator (sub-index 2).

Remark: the gear ratio is computed after sub-index 2 is written. So sub-index1 must be written first and then sub-index 2. Even if sub-index 2 has the same value as before, it must be written again to be able to compute the gear ratio.

Object description:

Index	2013_h
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

10.2.4 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 in object 201Eh or from the digital inputs of the drive). The units are increments.

Object description:

Index	2017_h
Name	Master actual position
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	Possible
Value range	-2^{31} … 2^{31}-1
Default value	0

10.2.5 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 encoder increments / sampling loop. The default sampling loop is 1ms.

Object description:

Index	2018_h
Name	Master actual speed
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RO
PDO mapping	Possible
Value range	-32768 … 32767
Default value	0

10.2.6 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	$201D_h$
Name	External Reference Type
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	-

Table 10.3 *External Reference Type bit description*

Value	**Description**
0	Reserved
1	On-line. Received via object $201E_h$.
2	Analogue. In case of External Reference Position / Speed / Torque Modes, select this option in order to read the reference from the dedicated analogue input.
3	Digital[1]. In case of External Reference Position Modes, select this option in order to read the reference from the dedicated digital inputs as set in the setup made using PRO Config / MotionPRO Developer (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 PRO Config / MotionPRO Developer (either 2nd encoder or pulse & direction)
4 … 65535	Reserved

[1] Not all drives and configurations have secondary encoder inputs.

10.3 Electronic gearing through online communication example

Start an Electronic Gearing Slave.

1. Map in a RPDO the object 201Eh Master position to be able to send the drive the position reference every communication cycle. See **2.4 PDOs mapping example** or paragraph **1.3.3** for a TwinCAT PDO mapping example.

   The PDOs must be sent every slow loop period which is by default 0.8ms for Stepper configurations or 1ms for the rest. It is recommended to set the SYNC 0 time equal to the communication cycle and slow loop. For this setting, object 2082h Sync on fast loop must be 0.

2. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06_h.

4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07_h.

5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

6. **External reference type.** Slave receives reference through online communication.

   Send the following message: SDO access to object $201D_h$ 16-bit value 0001_h.

7. **Modes of operation.** Select Electronic Gearing mode (-1).

   Set in **Modes of Operation** mapped in RPDO1 the value FF_h.

8. **Master resolution.** Set the master resolution 2000, assuming a 500 line encoder is used.

   Send the following message: SDO access to object 2012_h 32-bit value $000007D0_h$.

9. **Electronic gearing multiplication factor.**

   Set EG numerator to 1.

   Send the following message: SDO access to object 2013_h, sub-index 1, 16-bit value 0001_h.

   Set EG denominator to 1.

   Send the following message SDO access to object 2013_h,sub-index 2, 16-bit value 0001_h.

10. Set the initial Master position into the associated RPDO where $201E_h$ is mapped.

11. **Enable EG slave** in control word associated RPDO.

    Set in **Control Word** mapped in RPDO1 the value $1F_h$.

12. Start changing the Master position in the RPDO where $201E_h$ is mapped every communication cycle.

The slave motor should start rotating with the same speed as the difference between the master position values received every slow loop.

11 Electronic Camming Position (ECAM) Mode

11.1 Overview

In Electronic Camming Position the drive executes a cam profile function of the position of an electronic camming master. The cam profile is defined by a cam table – a set of (X, Y) points, where X is cam table input i.e. the position of the electronic camming master and Y is the cam table output i.e. the corresponding slave position. Between the points the drive performs a linear interpolation.

The electronic camming slave can get the position information from the electronic camming master in three ways:

1. Via EtherCAT® master, who writes the master position in object **Master position** (index $201E_h$).
2. Via an external digital reference of type pulse & direction or quadrature encoder. Both options have dedicated inputs. The pulse & direction signals are usually provided by an indexer and must be connected to the pulse & direction inputs of the drive. The quadrature encoder signals are usually provided by an encoder on the master and must be connected to the 2nd encoder inputs.
3. From one of the analogue inputs of the drive.

The reference type, i.e. the selection between the online reference received via communication channel and the digital reference read from dedicated inputs is done with object **External Reference Type** (index $201D_h$). The source of the digital reference (pulse & direction or second encoder inputs) is set during drive commissioning.

The electronic camming position mode can be: **relative** (if ControlWord.6 = 0) or **absolute** (if ControlWord.6 = 1).

In the relative mode, the output of the cam table is added to the slave actual position. At each slow loop sampling period the slave computes a position increment **dY = Y – Yold**. This is the difference between the actual cam table output Y and the previous one Yold. The position increment dY is added to the old demand position to get a new demand position. The slave detects when the master position rolls over, from 360 degrees to 0 or vice-versa and automatically compensates in dY the difference between **Ymax** and **Ymin**. Therefore, in relative mode, you can continuously run the master in one direction and the slaves will execute the cam profile once at each 360 degrees with a glitch-free transition when the cam profile is restarted.

When electronic camming is activated in relative mode, the slave initializes **Yold** with the first cam output computed: **Yold = Y = f(X)**. The slave will keep its position until the master starts to move and then it will execute the remaining part of the cam. For example if the master moves from X to Xmax, the slave moves with Ymax – Y.

In the absolute mode, the output of the cam table Y is the demand position to reach.

Remark: *The absolute mode must be used with great care because it may generate abrupt variations on the slave demand position if:*

- Slave position is different from Y at entry in the camming mode
- Master rolls over and Ymax < Ymin

In the absolute mode, you can introduce a maximum speed limit to protect against accidental sudden changes of the positions to reach. The feature is activated by setting ControlWord.5=1 and the maximum speed value in object **Profile Velocity** (index 6081_h).

Typically, the cam tables are first downloaded into the EEPROM memory of the drive by the EtherCAT® master or with MotionPRO Developer. 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.

11.1.1 Controlword in electronic camming position mode

MSB **LSB**

See 6040h	Halt	See 6040h	Abs / Rel	Activate Speed Limitation	Enable Electronic Camming Mode	See 6040h
15 9	8	7	6	5	4	3 0

Table 11.1 *Controlword bits for electronic camming position mode*

Name	Bit	Value	Description
Enable Electronic Camming Mode	4	0	Do not start operation
		0 -> 1	Start electronic camming procedure
		1	Electronic camming mode active
		1 -> 0	Do nothing (does not stop current procedure)
Activate Speed Limitation	5	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	6	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	8	0	Execute the instruction of bit 4
		1	Stop electronic camming motion with the *profile acceleration*

11.1.2 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 11.2 *Statusword bits for electronic camming position mode*

Name	Bit	Value	Description

Target reached	10	0	Halt = 0: Always 0
			Halt = 1: Drive decelerates

		1	Halt = 0: Always 0 Halt = 1: Velocity of drive is 0
Following error	13	0	No following error
		1	Following error occurred

11.2 Electronic Camming Position Mode Objects

11.2.1 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 $201A_h$, for faster processing. The copy is made every time object 2019_h 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	2019_h
Name	CAM table load address
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	UNSIGNED16
Default value	-

11.2.2 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 2019_h).

Object description:

Index	$201A_h$
Name	CAM table run address
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	UNSIGNED16
Default value	-

11.2.3 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 (2017_h) but (master actual position – CAM offset) computed as modulo of master resolution (2012_h) The CAM offset must be set before enabling the electronic camming mode. The *CAM offset* is expressed in increments.

Object description:

Index	$201B_h$
Name	CAM offset
Object code	VAR
Data type	UNSIGNED32

Entry description:

Access	RW
PDO mapping	No
Value range	$0 \ldots 2^{32}$-1
Default value	0

11.2.4 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	$206B_h$
Name	CAM input scaling factor
Object code	VAR
Data type	FIXED32

Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	FIXED32
Default value	00010000_h

11.2.5 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	$206C_h$
Name	CAM output scaling factor
Object code	VAR
Data type	FIXED32

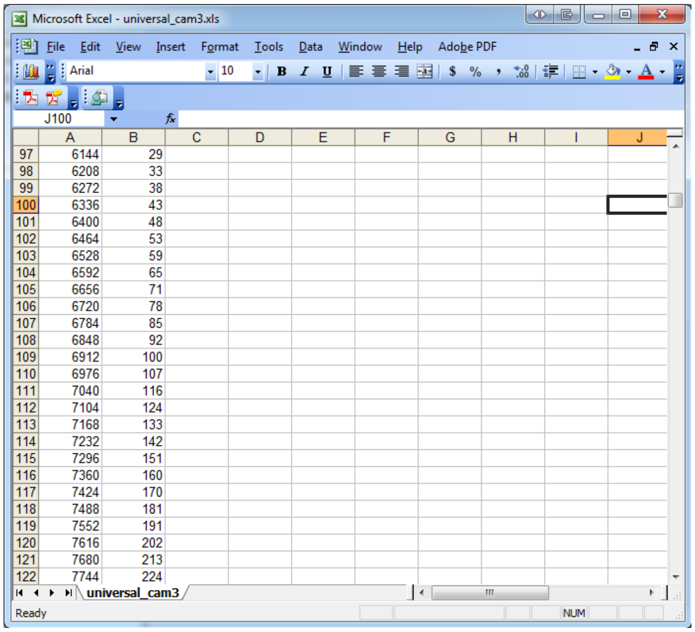
Entry description:

Access	RW
PDO mapping	No
Units	-
Value range	FIXED32
Default value	00010000_h

11.2.6 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 Excell.



	A	B	C	D	E	F	G	H	I	J
97	6144	29								
98	6208	33								
99	6272	38								
100	6336	43								
101	6400	48								
102	6464	53								
103	6528	59								
104	6592	65								
105	6656	71								
106	6720	78								
107	6784	85								
108	6848	92								
109	6912	100								
110	6976	107								
111	7040	116								
112	7104	124								
113	7168	133								
114	7232	142								
115	7296	151								
116	7360	160								
117	7424	170								
118	7488	181								
119	7552	191								
120	7616	202								
121	7680	213								
122	7744	224								

***Figure 11.1** MS Excell interface*

The numbers in the columns represent the input and output of the cam file. They are position points represented in the drive internal units. Let’s say that we have an 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 it turned the rotor 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, 2^7. So let’s 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’s say that 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).

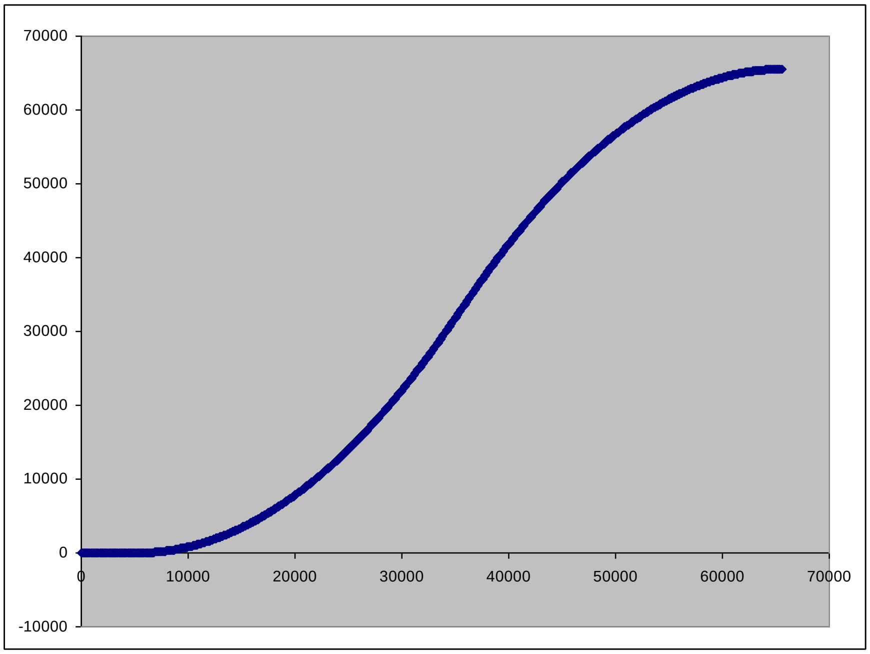
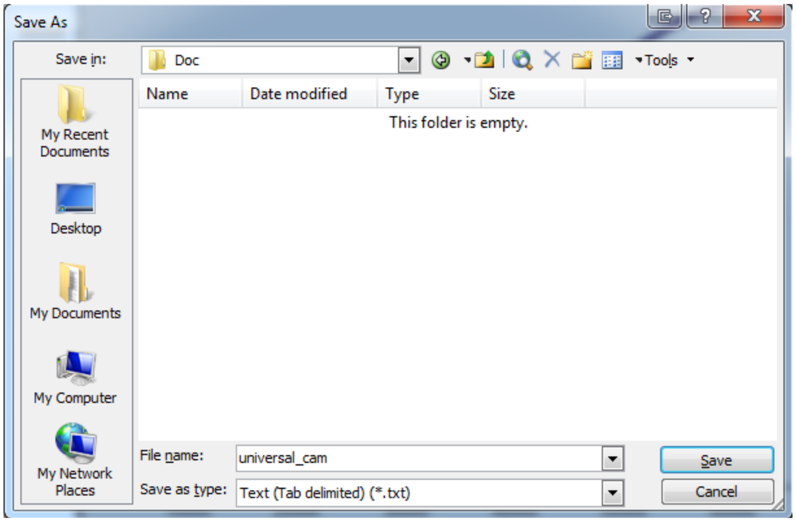


Figure 11.2 *Cam example*

After your cam is ready, save it as Text (Tab delimited) (*.txt) file.

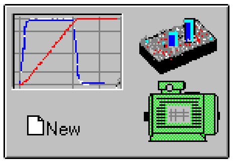


File name: universal_cam

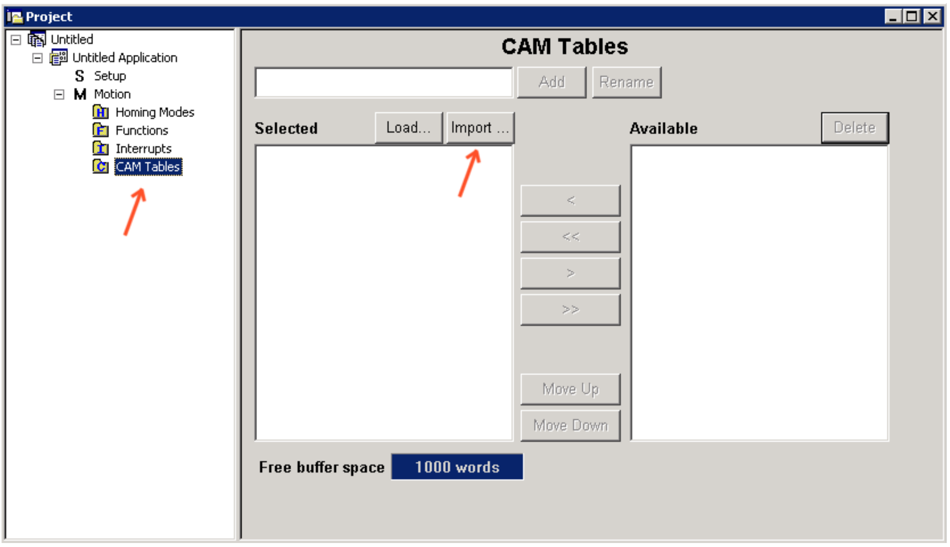
Save as type: Text (Tab delimited) (*.txt)

Figure 11.3 *Save As example.*

Once you have your cam file saved start MotionPRO Developer.

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

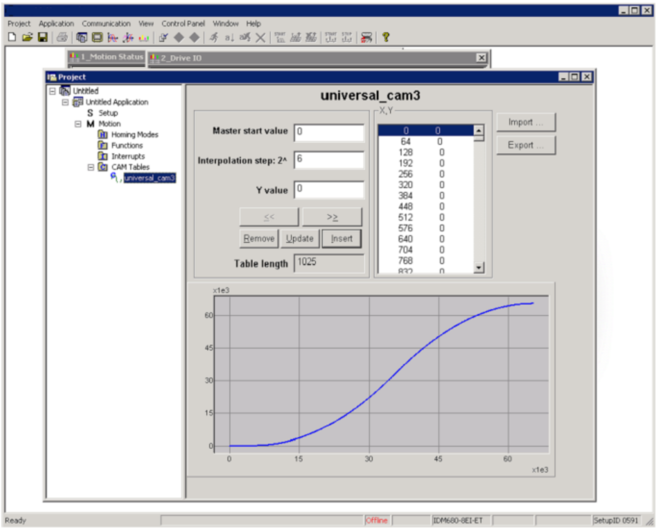
After the project opens, select CAM Tables tab from the left of the screen. Press the import button and choose your recently saved cam file. (See Figure 11.5)



Free buffer space 1000 words

Figure 11.4 *CAM tab.*

If the CAM file loaded it should look like this:



Master start value 0

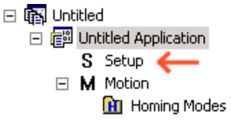
Interpolation step: 2^ 6

Y value 0

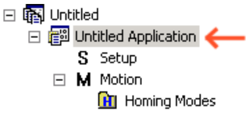
Table length 1025

X	Y
0	0
64	0
128	0
192	0
256	0
320	0
384	0
448	0
512	0
576	0
640	0
704	0
768	0
832	0

Figure 11.5 *CAM file loaded.*

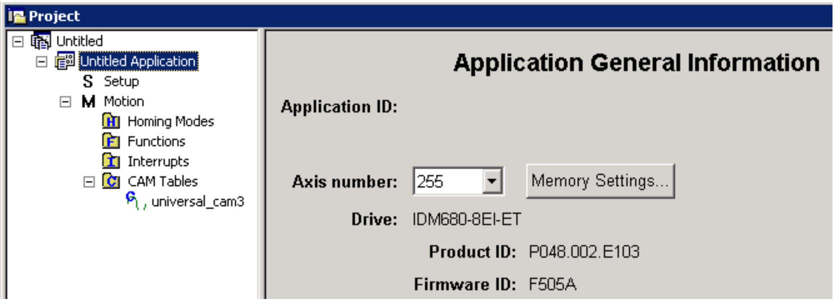


After loading the CAM file successfully click over the Setup tab and load your saved setup made in **Paragraph 1.1.4** .



Click the tab with the name of the application .

Press the memory settings button (like in the figure below).



Application General Information

Application ID:

Axis number: 255

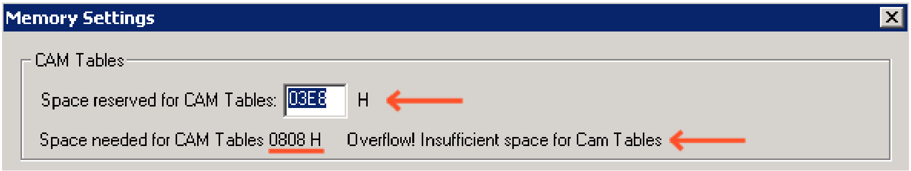
Drive: IDM680-8EI-ET

Product ID: P048.002.E103

Firmware ID: F505A

Figure 11.6 *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)

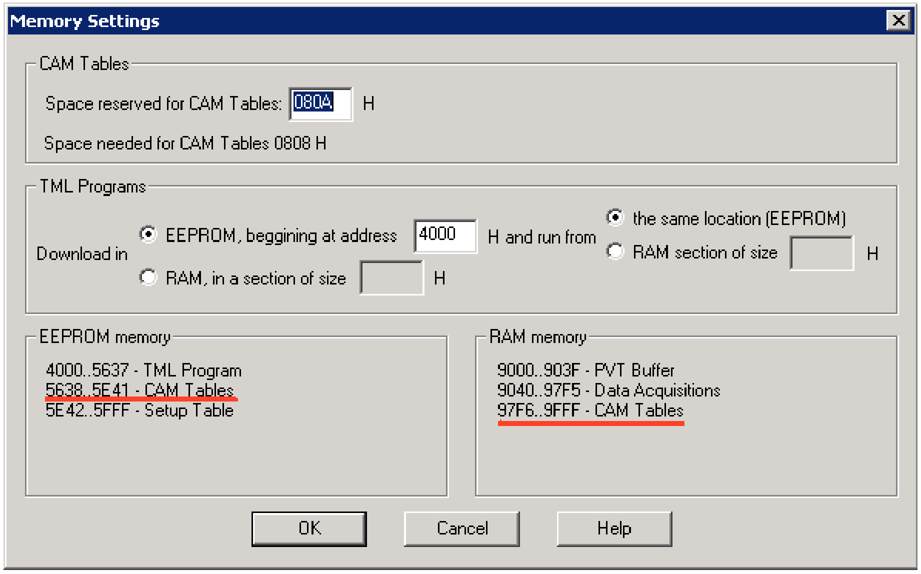


Space reserved for CAM Tables: 03E8 H

Space needed for CAM Tables 0808 H Overflow! Insufficient space for Cam Tables

Figure 11.7 *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** which must be set also in object **2019**$_{\mathbf{h}}$ afterwards.



Memory Settings

CAM Tables

Space reserved for CAM Tables: 080A H

Space needed for CAM Tables 0808 H

TML Programs

Download in ● EEPROM, beggining at address 4000 H and run from ● the same location (EEPROM) ○ RAM section of size H

○ RAM, in a section of size H

EEPROM memory

4000..5637 - TML Program
5638..5E41 - CAM Tables
5E42..5FFF - Setup Table

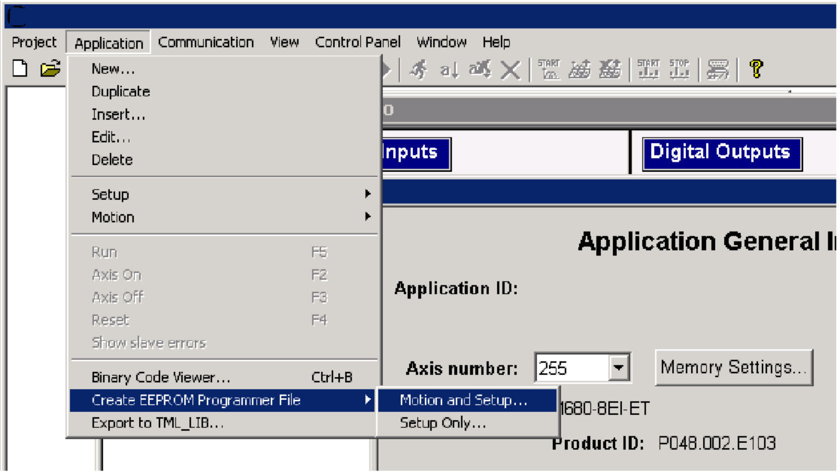
RAM memory

9000..903F - PVT Buffer
9040..97F5 - Data Acquisitions
97F6..9FFF - CAM Tables

Figure 11.8 *Cam table load and run addresses.*

Under the RAM memory section the fist number in CAM Tables is the **cam table Run Address** which must also be set in object **201A$_h$** afterwards.

Save the project and select Application -> Create EEPROM programmer file -> Motion and Setup… like in the figure below. Save the EPPROM file that includes your setup and motion (including CAM data) onto your PC.



Application ID:

Axis number: 255

1680-8EI-ET

Product ID: P048.002.E103

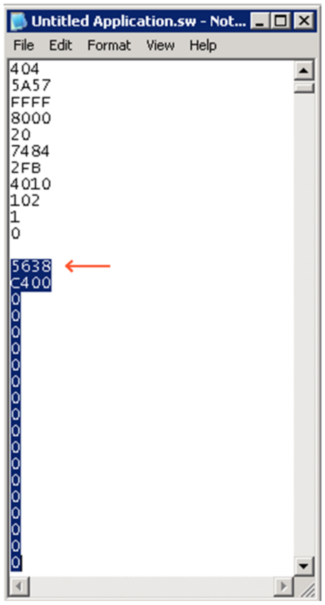
Figure 11.9 *Create .sw file.*

11.2.6.1 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.

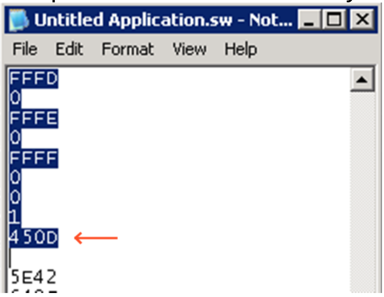
This number shall be delimited by an empty new line just before it (the numbers before it represent the setup data).



```
404
5A57
FFFF
8000
20
7484
2FB
4010
102
1
0

5638
C400
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Select all these numbers that represent the cam file until you find another empty new line.

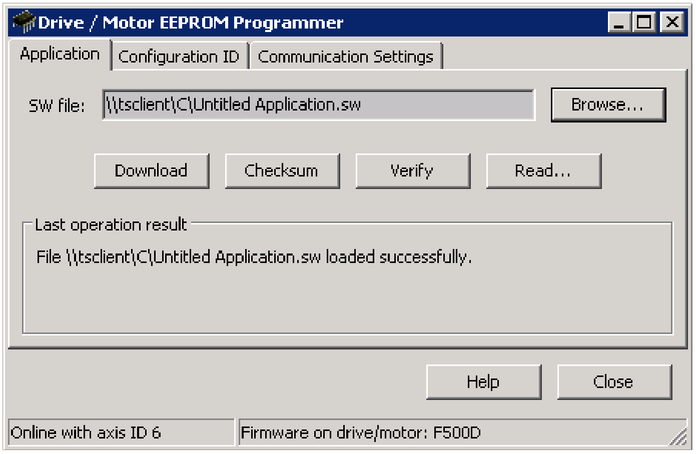


```
FFFD
0
FFFE
0
FFFF
0
0
1
450D

5E42
```

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 the EEPROM Programmer (supplied free with MotionPRO Developer) or load it with the help of **2064$_h$ 2065$_h$** objects explained in next sub chapter.



SW file: \\tsclient\C\Untitled Application.sw

Last operation result

File \\tsclient\C\Untitled Application.sw loaded successfully.

Online with axis ID 6

Firmware on drive/motor: F500D

Figure 11.10 *THS EEPROM Programmer.*

Note: with THS EEPROM programmer you can write the entire setup and motion .sw file, not just the CAM .sw file created in this example.

11.2.6.2 Downloading a CAM .sw file with objects 2064h and 2065h example

- Send the following message: SDO access to object 2064$_h$ 32-bit value xxxx0008$_h$.

Where xxxx is the first 16 bit number found in the CAM .sw file and represents the CAM table load address. The 08 activates writing/reading 16 bit data in EEPROM memory in object 2064$_h$ (see more in the object description).

All the next numbers until the end of the file must be written with the following type of command.

- Send the following message: SDO access to object 2065$_h$ 32-bit value 0000xxxx$_h$.

Where xxxx is the 16 bit number taken from the .sw file (the data to write) after the first one (which is the address at which to first write the data and increment it).

⚠	**Warning!**	When object 2064$_h$ bit 7=0 (auto-incrementing is ON), do not read the object list in parallel with a read/write operation using a script. By reading object 2066h in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

11.3 Electronic gearing through online communication example

Start an Electronic Gearing Slave.

1. Map in a RPDO the object 201Eh Master position to be able to send the drive the position reference every communication cycle. See **2.4 PDOs mapping example** or paragraph **1.3.3** for a TwinCAT PDO mapping example.

   The PDOs must be sent every slow loop period which is by default 0.8ms for Stepper configurations or 1ms for the rest. It is recommended to set the SYNC 0 time equal to the communication cycle and slow loop. For this setting, object 2082_h Sync on fast loop must be 0.

2. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

3. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06_h.

4. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07_h.

5. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value $0F_h$.

6. **External reference type.** Slave receives reference through online communication.

   Send the following message: SDO access to object $201D_h$ 16-bit value 0001_h.

7. **Cam table load address.** Set cam table load address as **5638_h**.

   The cam table load address can be discovered as explained in paragraph 11.2.6 .

   Send the following message: SDO access to object 2019_h 16-bit value 5638_h.

8. **Cam table run address.** Set cam table load address as **$97F6_h$**.

   The cam table load address can be discovered as explained in paragraph 11.2.6 .

   Send the following message: SDO access to object $201A_h$ 16-bit value $97F6_h$.

9. **Modes of operation.** Select Electronic Camming mode (-2).

   Set in **Modes of Operation** mapped in RPDO1 the 8 bit value 0xFE.

10. **Master resolution.** Set the master resolution 2000, assuming a 500 line encoder is used.

    Send the following message: SDO access to object 2012_h 32-bit value $000007D0_h$.

11. **Cam offset.** Set cam offset to 6000 counts (0x1770).

    If the master resolution is 2000 counts/revolution, the slave shall start applying the cam when the master is at position 6000 + CamX value.

    Send the following message: SDO access to object 201B$_h$ 32-bit value 00001770$_h$.

12. **Cam input scaling factor.** Set it to 1.

    Send the following message: SDO access to object 206B$_h$ 32-bit value 00000001$_h$.

13. **Cam output scaling factor.** Set it to 1.

    Send the following message: SDO access to object 206C$_h$ 32-bit value 00000001$_h$.

13. Set the initial Master position into the associated RPDO where 201E$_h$ is mapped.

14. **Enable EG slave** in control word associated RPDO.

    Set in **Control Word** mapped in RPDO1 the value 3F$_h$ to start electronic gearing and activate speed limitation.

15. Start changing the Master position in the RPDO where 201Eh is mapped every communication cycle.

The slave motor should start rotating. After the master reports the position 6000 IU (cam offset), the slave motor shall rotate depending on the set cam values.

12 Cyclic synchronous position mode

12.1 Overview

The overall structure for this mode is shown in Figure 12.1. With this mode, the trajectory generator is located in the control device, not in the drive device. In cyclic synchronous manner, it provides a target position to the drive device, which performs position control, velocity control and torque control. Measured by sensors, the drive provides actual values for position, velocity and torque to the control device.

The cyclic synchronous position motion is limited to a maximum velocity by setting a number in object 6081_h Profile velocity. By default this object has the value 0. This means that the motion will not start, until an acceptable velocity value is set.

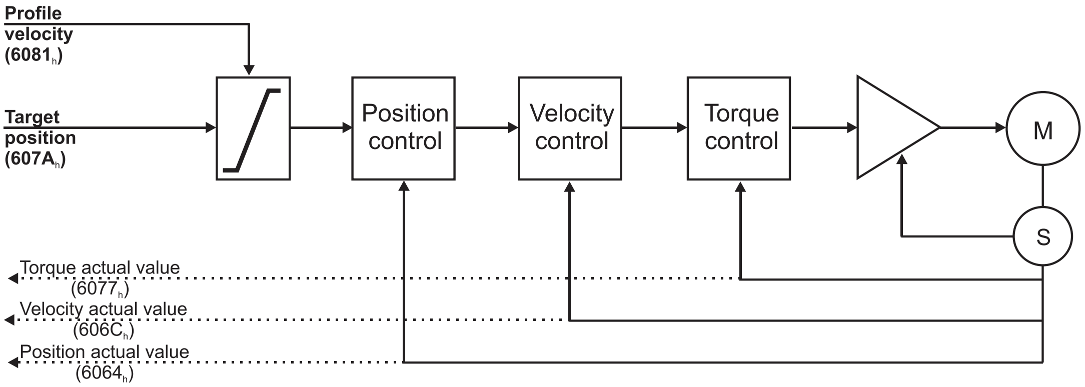


Figure 12.1 *Cyclic synchronous position mode overview*

12.1.1 Controlword in synchronous position mode

MSB LSB

See 6040h	Halt	See 6040h	Abs / rel	Reserved	Reserved	See 6040h
15 9	8	7	6	5	4	3 0

Table 12.1 *Control Word bits description for Interpolated 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 found in object $607A_h$. By sending this value periodically and setting the correct interpolation period time in object $60C2_h$, it will be similar to working in Cyclic Synchronous Velocity (CSV) mode without the speed loop active.

12.1.2 Statusword in synchronous position mode

MSB **LSB**

See 6041h	Following error	Target position ignored	See 6041h	Reserved	See 6041h
15 14	13	12	11	10	9 0

Table 12.2 *Status Word bit description for External Reference 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

12.1.3 Object 2086h: Limit speed/acceleration for CSP/CSV

This object is used to set a maximum velocity/acceleration during CSP or CSV modes of operation.

Object description:

Index	2086_h
Name	Limit speed/acceleration for CSP/CSV
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Yes
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. During CSV mode, the maximum acceleration will be the one defined in object 6083_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. The same scenario applies to the CSV mode.

12.2 Cyclic synchronous position profile example

1. **Start remote node**.

   Enter **Pre-Operational** state.

   Enter **Safe-Operational** state.

   Enter **Operational** state.

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

   Set in **Control Word** mapped in RPDO1 the value 06$_h$.

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

   Set in **Control Word** mapped in RPDO1 the value 07$_h$.

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

   Set in **Control Word** mapped in RPDO1 the value 0F$_h$.

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

   Set in **Modes of Operation** mapped in RPDO1 the value 08$_h$.

6. **Send position points.** The drive will execute a new motion with every new value it receives in RxPDO2 variable Target position which is object 607A$_h$.

   Set in **Target position** mapped in RPDO2, the 32bit value xxxxxxxx$_h$.

13 Cyclic synchronous velocity mode

13.1 Overview

The overall structure for this mode is shown in Figure 13.1. 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 PRO Config 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.

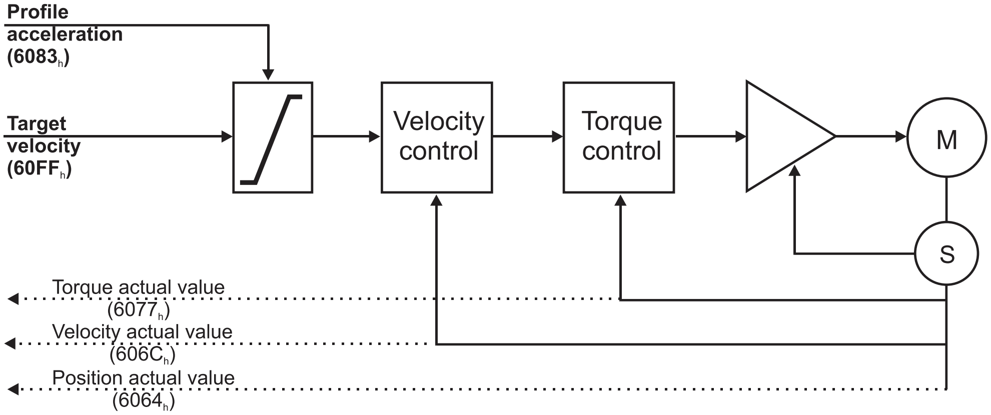

Profile acceleration (6083h)
Target velocity (60FFh)
Velocity control
Torque control
M
S
Torque actual value (6077h)
Velocity actual value (606Ch)
Position actual value (6064h)

Figure 13.1 Cyclic synchronous velocity mode overview

13.1.1 Controlword in synchronous velocity mode

The cyclic synchronous velocity mode uses no mode specific bits of the controlword. See 6040h.

13.1.2 Statusword in synchronous velocity mode

MSB LSB

See 6041h	Reserved	Target velocity ignored	See 6041h	Reserved	See 6041h
15 14	13	12	11	10	9 0

Table 13.1 *Status Word bit description for External Reference Speed Mode*

Name	Value	Description
Bit 10	0	Reserved
	1	Reserved
Speed	0	Target velocity ignored
	1	Target velocity shall be used as input to velocity control loop
Bit 13	0	Reserved
	1	Reserved

13.2 Cyclic synchronous velocity profile example

1. **Start remote node**.

   Enter **Pre-Operational** state.

2. **Disable the Sync Manager $1C12_h$ Subindex 0 to 0.**

   Send the following message: SDO access to object $1C12_h$ 8-bit value 00_h.

3. **Map RPDO4 to Sync Manager $1C12_h$ Subindex 3.** The RPDO4 has mapped by default, the object $60FF_h$ Target velocity.

   Send the following message: SDO access to object $1C12_h$, Subindex 3, 16-bit value 1603_h.

4. **Enable the Sync Manager $1C12_h$ by setting Subindex 0 to 3.**

   Send the following message: SDO access to object $1C12_h$ 8-bit value 03_h.

5. **Enter Operational state.**

   Enter **Safe-Operational** state.

   Enter **Operational** state.

6. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06_h.

7. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07$_h$.

8. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value 0F$_h$.

9. **Modes of operation.** Select cyclic synchronous velocity mode.

   Set in **Modes of Operation** mapped in RPDO1 the value 09$_h$.

10. **Send velocity points.** The drive will execute a new motion with every new value it receives in RxPDO4 variable Target velocity which is object 60FF$_h$.

   Set in **Target velocity** mapped in RPDO4, the 32bit value xxxxxxxx$_h$.

Remark: By default, without the Factor Group set, the Target velocity structure is 16.16. Meaning the integer part of the speed in IU is set in the MSB and the fractional is set in the LSB.

14 Cyclic synchronous torque mode

14.1 Overview

The overall structure for this mode is shown in Figure 14.1. With this mode, the trajectory generator is located in the control device, not in the drive device. In cyclic synchronous manner, it provides a target torque to the drive device, which performs torque control.

Measured by sensors, the drive device provides actual values for position, velocity and torque to the control device.

The cyclic synchronous torque mode covers the following sub-functions:

- demand value input;
- torque capture;
- torque control function with appropriate input and output signals;
- limitation of torque demand.

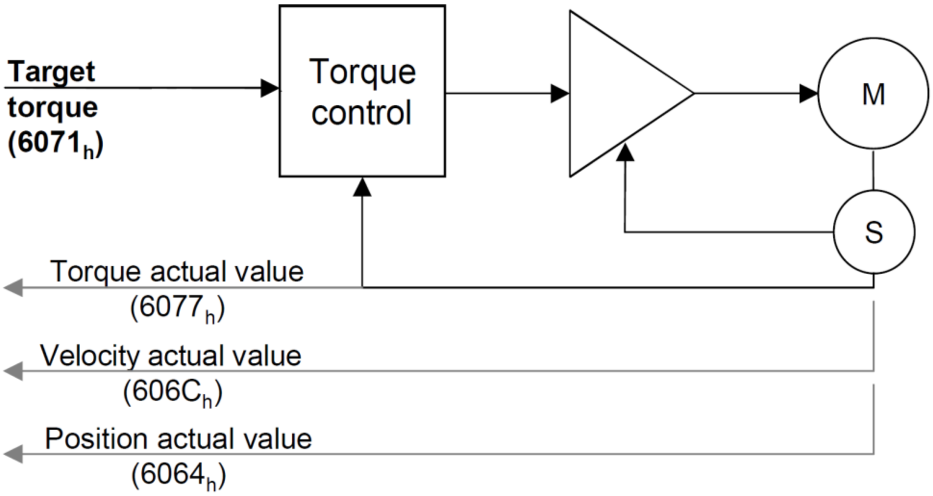

Target torque (6071_h)
Torque control
M
S
Torque actual value (6077_h)
Velocity actual value ($606C_h$)
Position actual value (6064_h)

Figure 14.1 Cyclic synchronous torque mode overview

14.1.1 Controlword in cyclic synchronous torque mode

The cyclic synchronous torque mode uses no mode specific bits of the Controlword. See 6040h.

14.1.2 Statusword in cyclic synchronous torque mode

MSB **LSB**

See 6041h	Reserved	Target torque ignored	See 6041h	Reserved	See 6041h
15 14	13	12	11	10	9 0

Table 14.1 *Status Word bit description for Cyclic Synchronous Torque Mode*

Name	**Value**	**Description**
Bit 10	0	Reserved
	1	Reserved
Target torque ignored	0	Target torque ignored
	1	Target torque shall be used as input to torque control loop
Bit 13	0	Reserved
	1	Reserved

14.1.3 Object 6071h: Target torque

This is used to indicate the configured input value for the torque controller in profile torque mode. The unit for this object is given in IU.

Object description:

Index	6071_h
Name	Target torque
Object code	VAR
Data type	INTEGER16

Entry description:

Access	RW
PDO mapping	Yes
Value range	UNSIGNED16
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 *Ipeak* is the peak current supported by the drive and *current[IU]* is the command value for object 6071$_h$.

14.1.4 Object 6077h: Torque actual value

This is used to provide the actual value of the torque. 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	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 *Ipeak* is the peak current supported by the drive and *current[IU]* is the read value from object 6077$_h$.

14.2 Cyclic synchronous velocity profile example

1. **Start remote node**.

   Enter **Pre-Operational** state.

2. **Disable the Sync Manager 1C12$_h$ Subindex 0 to 0.**

   Send the following message: SDO access to object 1C12$_h$ 8-bit value 00$_h$.

3. **Map RPDO3 to Sync Manager 1C12$_h$ Subindex 3.** The RPDO3 has mapped by default, the object 6071$_h$ Target torque.

   Send the following message: SDO access to object 1C12$_h$, Subindex 3, 16-bit value 1602$_h$.

4. **Enable the Sync Manager 1C12$_h$ by setting Subindex 0 to 3.**

   Send the following message: SDO access to object 1C12$_h$ 8-bit value 03$_h$.

5. **Enter Operational state.**

   Enter **Safe-Operational** state.

   Enter **Operational** state.

6. **Ready to switch on.** Change the node state from *Switch on disabled* to *Ready to switch on* by sending the shutdown command.

   Set in **Control Word** mapped in RPDO1 the value 06$_h$.

7. **Switch on.** Change the node state from *Ready to switch on* to *Switch on* by sending the switch on command.

   Set in **Control Word** mapped in RPDO1 the value 07$_h$.

8. **Enable operation.** Change the node state from *Switch on* to *Operation enable* by sending the enable operation command.

   Set in **Control Word** mapped in RPDO1 the value 0F$_h$.

9. **Modes of operation.** Select cyclic synchronous torque mode.

   Set in **Modes of Operation** mapped in RPDO1 the value 0A$_h$.

10. **Send target torque points.** The drive will apply a new current value with every new command it receives in RxPDO3 variable Target torque which is object 6071$_h$.

   Set in **Target torque** mapped in RPDO4, the 16bit value xxxx$_h$.

15 Touch probe functionality

15.1 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.

15.2 Touch probe objects

15.2.1 Object 60B8h: 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

The *Touch probe function* has the following bit assignment:

Table 15.1 *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 captured position will always be from the load feedback input
- 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.

15.2.2 Object 60B9h: Touch probe status

This object provides the status of the touch probe.

Object description:

Index	$60B9_h$
Name	Touch probe status
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

The *Touch probe status* has the following bit assignment:

Table 15.2 *Bit Assignment of the Touch probe status*

Bit	Value	Description
11 to 15	-	Reserved
10	0	Touch probe 2 no negative edge value stored
	1	Touch probe 2 negative edge position stored in object $60BD_h$
9	0	Touch probe 2 no positive edge value stored
	1	Touch probe 2 positive edge position stored in object $60BC_h$
8	0	Touch probe 2 is switched off
	1	Touch probe 2 is enabled
7	-	Reserved
6	0	Limit switch functionality enabled.
	1	Limit switch functionality disabled.
3 to 5	-	Reserved
2	0	Touch probe 1 no negative edge value stored
	1	Touch probe 1 negative edge position stored in object $60BB_h$
1	0	Touch probe 1 no positive edge value stored
	1	Touch probe 1 positive edge position stored in object $60BA_h$
0	0	Touch probe 1 is switched off
	1	Touch probe 1 is enabled

Note: Bit 1 and bit 2 are set to 0 when touch probe 1 is switched off (object $60B8_h$ bit 0 is 0). Bit 9 and 10 are set to 0 when touch probe 2 is switched off (object $60B8_h$ bit 8 is 0). Bits 1,2,9 and 10 are set to 0 when object $60B8_h$ bits 4,5,12 and 13 are set to 0.

15.2.3 Object 60BAh: Touch probe 1 positive edge

This object provides the load position value of the touch probe 1 at positive edge.

Object description:

Index	$60BA_h$
Name	Touch probe 1 positive edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}…2^{31}-1$
Default value	-

15.2.4 Object 60BBh: Touch probe 1 negative edge

This object provides the load position value of the touch probe 1 at negative edge.

Object description:

Index	$60BB_h$
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	-

15.2.5 Object 60BCh: Touch probe 2 positive edge

This object provides the load 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	$-2^{31}\ldots2^{31}-1$
Default value	-

15.2.6 Object 60BDh: Touch probe 2 negative edge

This object provides the load 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	$-2^{31}\ldots2^{31}-1$
Default value	-

15.2.7 Object 2104h[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	UNSIGNED16

Entry description:

Access	RW
PDO mapping	Yes
Value range	0 … 65535
Default value	0

The *auxiliary feedback touch probe function* has the following bit assignment:

Table 15.3 *Bit Assignment of the Auxiliary encoder function*

Bit	Value	Description
15..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**

[1] Object 2104h applies only to drives which have a secondary feedback input with an index signal

The position cannot be captured on both positive and negative edges simultaneously using the zero impulse signal as a trigger.

15.2.8 Object 2105h[1]: Auxiliary encoder status

This object provides the status of the auxiliary feedback touch probe.

Object description:

Index	2105_h
Name	Auxiliary encoder status
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

The *auxiliary feedback touch probe status* has the following bit assignment:

Table 15.4 *Bit Assignment of the Auxiliary encoder status*

Bit	Value	Description
15 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 2107_h
1	0	Auxiliary feedback touch probe no positive edge value stored
	1	Auxiliary feedback touch probe positive edge position stored in object 2106_h
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 2104_h bit 0 is 0). Bits 1 and 2 are set to 0 when object 2104_h bits 4 and 5 are set to 0.

[1] Object 2105h applies only to drives which have a secondary feedback input with an index signal

15.2.9 Object 2106h[1]: Auxiliary encoder captured position positive edge

This object provides the position value of the auxiliary feedback captured at positive edge.

Object description:

Index	2106_h
Name	Auxiliary encoder captured positive edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}…2^{31}-1$
Default value	-

15.2.10 Object 2107h[2]: Auxiliary encoder captured position negative edge

This object provides the position value of the auxiliary feedback captured at negative edge.

Object description:

Index	2107_h
Name	Auxiliary encoder captured position negative edge
Object code	VAR
Data type	INTEGER32

Entry description:

Access	RO
PDO mapping	YES
Value range	$-2^{31}…2^{31}-1$
Default value	-

[1] Object 2106h applies only to drives which have a secondary feedback input with an index signal

[2] Object 2107h applies only to drives which have a secondary feedback input with an index signal

16 Data Exchange between EtherCAT® master and drives

16.1 Checking Setup Data Consistency

During the configuration phase, a EtherCAT® master can quickly verify using the checksum objects and a reference **.sw** file whether the non-volatile EEPROM memory of the drive contains the right information. If the checksum reported by the drive doesn't match the one computed from the **.sw** file, the EtherCAT® master can download the entire **.sw** file into the drive EEPROM using the communication objects for writing data into the drive EEPROM.

In order to be able to inspect or to program any memory location of the drive, as well as for downloading of a new MPL 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.

16.2 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 then 4 hexadecimal digits are shown, the value must be right justified. For example 92 represents 0x0092.

The **.sw** software files can be generated either from PRO Config or from MotionPRO Developer.

In PRO Config 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 MotionPRO Developer 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, MPL programs, cam tables (if present) and the drive/motor configuration ID. The option **Setup Only** produces a **.sw** file identical with that produced by PRO Config i.e. having only the setup data and the configuration ID.

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

- from a EtherCAT® master, using the communication objects for writing data into the drive EEPROM
- using the EEPROM Programmer tool, which comes with PRO Config but may also be installed separately. The EEPROM Programmer was specifically designed for repetitive fast and easy programming of **.sw** files into the ElectroCraft drives during production.

16.3 Data Exchange Objects

16.3.1 Object 2064h: 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 16.1 *Read/Write Configuration Register bit description*

Bit	Value	Description
31…16	x	16-bit memory address for the next read/write operation
15…8	0	Reserved (always 0)
7	0	Auto-increment the address after the read/write operation
	1	Do not auto-increment the address after the read/write operation
6…4	0	Reserved (always 0)
3,2	00	Memory type is program memory
	01	Memory type is data memory
	10	Memory type is EEPROM memory
	11	Reserved
1	0	Reserved (always 0)
0	0	Next read/write operation is with a 16-bit data
	1	Next read/write operation is with a 32-bit data

Warning! When bit 7=0 (auto-incrementing is ON), do not read the object list in parallel with a read/write operation using a script. By reading object 2066h in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

16.3.2 Object 2065h: 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

16.3.3 Object 2066h: Read 16/32 bits data from address set in Read/Write Configuration Register

This object is used to read 16 or 32-bit values with parameters that are specified in object 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

16.3.4 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 2064$_h$ – Read/Write Configuration Register. The rest of the bits in object 2064$_h$ 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	2067$_h$
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

16.3.5 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 206A$_h$.

Object description:

Index	2069$_h$
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

16.3.6 Object 206Ah: Checksum read register

This object shows the latest computed checksum.

Object description:

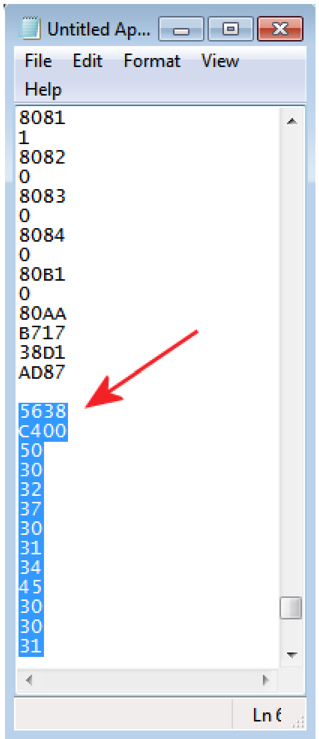
Index	206A$_h$
Name	Checksum read register
Object code	VAR
Data type	UNSIGNED16

Entry description:

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

16.4 Downloading an image file (.sw) to the drive via CoE commands

The structure of an image file (.sw) is described in paragraph ***16.2*** and shown in ***Figure 15.1***.



```
8081
1
8082
0
8083
0
8084
0
80B1
0
80AA
B717
38D1
AD87

5638
C400
50
30
32
37
30
31
34
45
30
30
31
```

Figure 15.1 *.sw file structure example*

In order to download the data block pointed by the red arrow, first the block start address i.e. 5638$_h$ must be set using an SDO access to object 2064$_h$.

- Send the following message: SDO access to object **2064**$_h$, 32-bit value **5638**0008$_h$.

The above configuration command also indicates that next read or write operation shall be executed with drive's EEPROM memory using 16-bit data and auto increment of address. All the numbers from the lines after 5638$_h$ until the following blank line represents data to write in the EEPROM memory at consecutive addresses starting with 5638$_h$. The data writes are done using an SDO access to object 2065$_h$. First data word C400$_h$ is written using:

- Send the following message: SDO access to object **2065**$_h$, 32-bit value 0000**C400**$_h$.

-the number 0000 will not be written; object 2064$_h$ was configured to write only 16 bit data

Next data word 0050$_h$ is written with:

- Send the following message: SDO access to object **2065**$_h$, 32-bit value 0000**0050**$_h$.

and so on, until the next blank line from the .sw file. As 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 2066h in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

16.5 Checking and loading the drive setup via .sw file and CoE commands example.

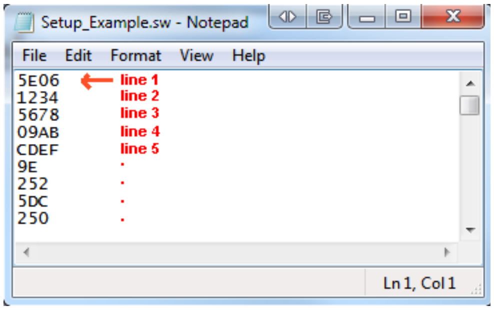
Check the integrity of the setup data on a drive and update it if needed.

Before reading this example, please read **paragraph 16.4.**

To create a .sw file containing only the setup data do the following:

- In MotionPRO Developer, go to Application (in the menu bar at the top)-> Create EEPROM Programmer File -> Setup Only… . Choose where to save the .sw file.
- In PRO Config, 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 ElectroCraft 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:



```
5E06
1234
5678
09AB
CDEF
9E
252
5DC
250
```

1. **Compute the checksum in the .sw file**. Let’s suppose that the computed checksum is 0x1234.
2. **Access object 2069_h in order to compute the checksum of the setup table located on the drive.** Write the value 0x**5EFF5E06**

   Send the following message: SDO write to object 2069_h sub-index 0, 32-bit value $5EFF5E06_h$.

Following the reception of this message, the drive will compute the checksum of the EEPROM locations 0x**5E06** to 0x**5EFF**. The result is stored in the object $206A_h$.

3. **Read the computed checksum from object $206A_h$.**

   Read by SDO protocol the value of object $206A_h$.

   Let’s assume the drive returns the following message (Object $206A_h$ = 0x2345):

As the returned checksum (0x2345) does not match the checksum computed from the .sw file, the setup table has to be configured from the .sw file.

4. **Prepare the Read/Write Configuration Register for EEPROM write**. Let’s assume the address 0x**5E06** is the first 16 bit number found in the .sw file where setup data begins**.** Write the value 0x**5E06**0009 into the object 2064_h (write 32-bit data at EEPROM address 0x**5E06** and auto-increment the address after the write operation).

   Send the following message: SDO write to object 2064_h sub-index 0, 32-bit value $5E060009_h$.

5. **Write the sw file data 32 bits at a time**. Supposing that the next 2 entries in the .sw file after the start address 0x**5E06** are 0x**1234** and 0x**5678**, you have to write the value 0x**56781234** into object 2065_h.

   Send the following message (SDO write to object 2065_h sub-index 0, 32-bit value 56781234_h):

The number 0x**1234** will be written at address 0x**5E06** and 0x**5678** will be at 0x**5E07**.

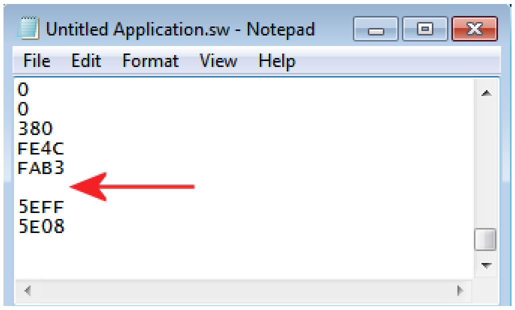
6. Assuming the next data after 0x**5678** will be 0x**09AB** and 0x**CDEF**, write the value 0x**CDEF09AB** into object 2065_h.

   Send the following message (SDO write to object 2065_h sub-index 0, 32-bit value $CDEF09AB_h$):

The number 0x**09AB** will be written at address 0x**5E08** and 0x**CDEF** will be at 0x**5E09**.

7. **Repeat step 5 until a blank line is found in the .sw file.**

This means that all the setup data is written, even if there is more data after the blank line.



```
0
0
380
FE4C
FAB3

5EFF
5E08
```

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.**

⚠	**Warning!**	When object 2064_h bit 7=0 (auto-incrementing is ON), do not read the object list in parallel with a read/write operation using a script. By reading object 2066h in parallel with another application, the target memory address will be incremented and will lead to incorrect data writing or reading.

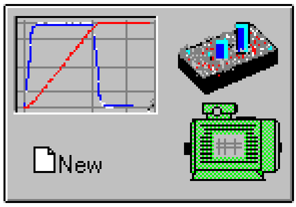
17 Advanced features

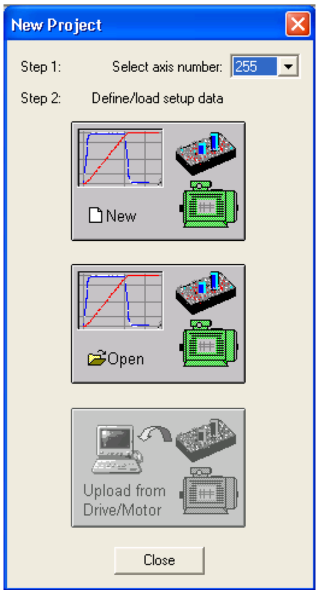
Due to its embedded motion controller, a ElectroCraft programmable drive/motor offers many programming solutions that may simplify a lot the task of a EtherCAT® master. This paragraph overviews a set of advanced programming features which can be used when combining MPL programming at drive level with EtherCAT® master control. All features presented below require usage of MotionPRO Developer as MPL programming tool

17.1 Using MotionPRO Developer

17.1.1 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 MotionPRO Developer as for PRO Config.

Press **New** button . A new window will appear.



New Project

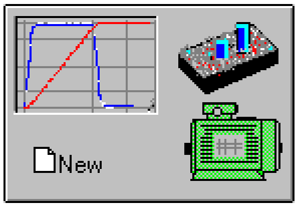
Step 1: Select axis number: 255

Step 2: Define/load setup data

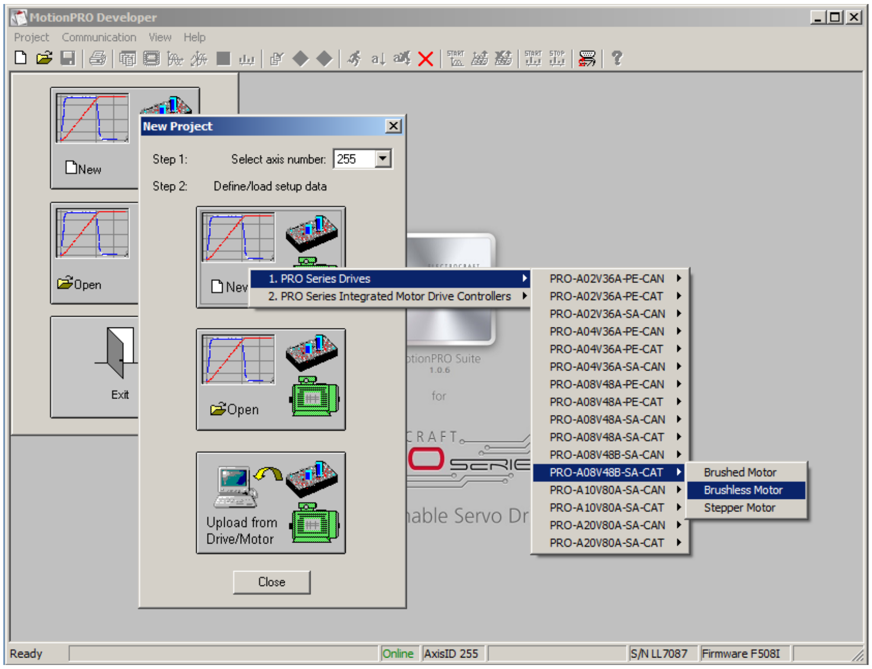
Step 1, selects the axis number for your drive. By default the drive is delivered with axis number 255.

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.

17.1.2 Choosing the drive, motor and feedback configuration

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

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).



Step 1: Select axis number: 255

Step 2: Define/load setup data

AxisID 255

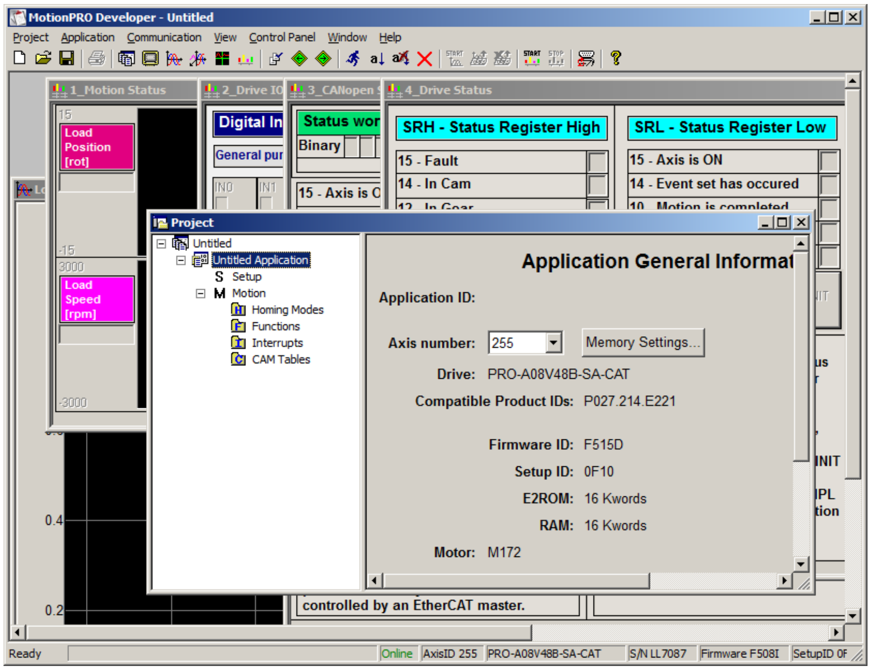
S/N LL7087

Firmware F508I

Figure 16.1 *MotionPRO Developer – 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.



Application General Informat

Application ID:

Axis number: 255

Drive: PRO-A08V48B-SA-CAT

Compatible Product IDs: P027.214.E221

Firmware ID: F515D

Setup ID: 0F10

E2ROM: 16 Kwords

RAM: 16 Kwords

Motor: M172

Ready | Online | AxisID 255 | PRO-A08V48B-SA-CAT | S/N LL7087 | Firmware F508I | SetupID 0F

***Figure 16.2** MotionPRO Developer – Project information*

To edit the setup, click *View / Modify* button.

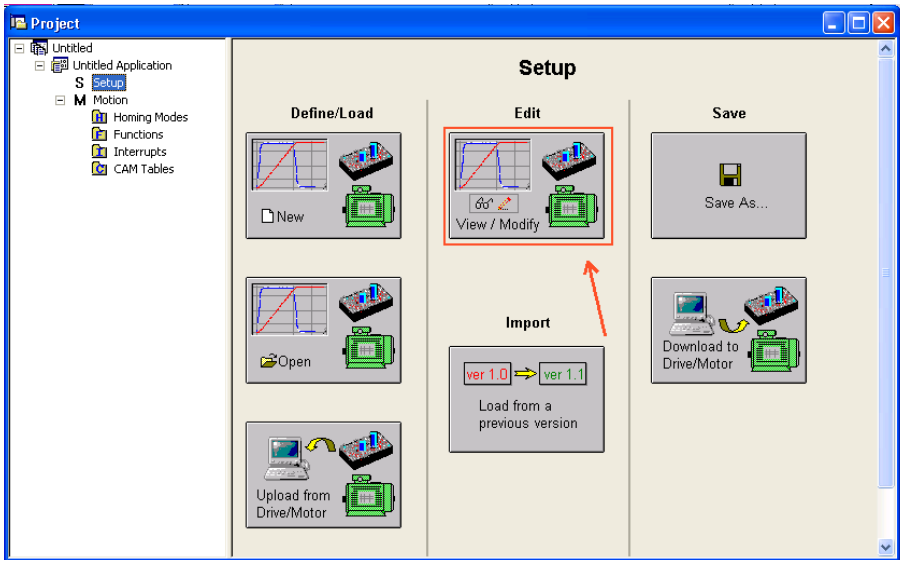


***Figure 16.3** MotionPRO Developer – Editing drive setup*

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 0** and **1.1.5 .**

17.1.3 Downloading setup data to drive/motor

Closing the Drive setup dialogue with **OK**, keeps the new settings only in the MotionPRO Developer project. In order to store the new settings into the drive you need to press the **Download to Drive/Motor** button Download to Drive/Motor or the ◆ button on the menu toolbar. This downloads the entire setup data in the drive EEPROM memory. The new settings become effective after the next power-on, when the setup data is copied into the active RAM memory used at runtime.

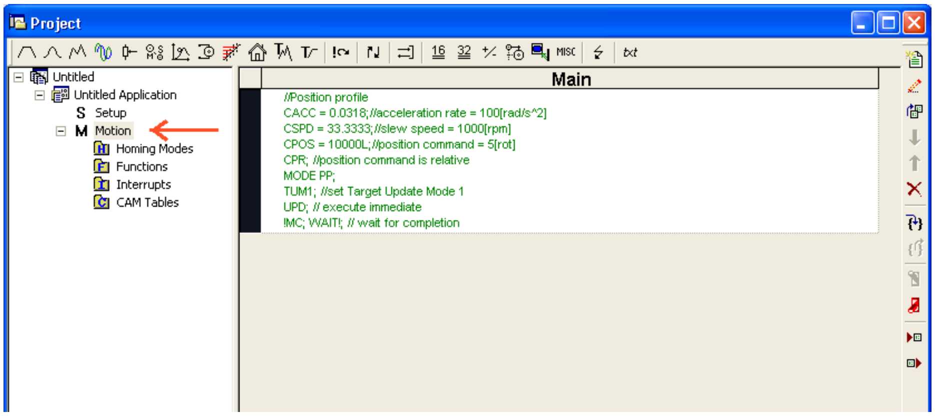
17.2 Using MPL Functions to Split Motion between Master and Drives

With ElectroCraft programmable drives you can really distribute the intelligence between an EtherCAT® master and the drives in complex multi-axis applications. Instead of trying to command each step of an axis movement, you can program the drives using MPL to execute complex tasks and inform the master when these are done. Thus for each axis, the master task may be reduced at: calling MPL program / functions (with possibility to abort their execution) stored in the drives EEPROM and waiting for a message, which confirms the finalization of the MPL motion / functions execution.

17.2.1 Build the MPL program within MotionPRO Developer

The following steps describes how to create a MPL program with MotionPRO Developer

1. **Define the MPL program.** Open the MotionPRO Developer project and select the Motion entry from the project tree.



```
//Position profile
CACC = 0.0318;//acceleration rate = 100[rad/s^2]
CSPD = 33.3333;//slew speed = 1000[rpm]
CPOS = 10000L;//position command = 5[rot]
CPR; //position command is relative
MODE PP;
TUM1; //set Target Update Mode 1
UPD; // execute immediate
!MC; WAIT!; // wait for completion
```

Figure 16.4 *MotionPRO Developer project window – Motion edit window*

2. **Add the MPL code.** To add MPL code click on the buttons in the Motion wizard bar to introduce new motion profiles, events or I/O commands.

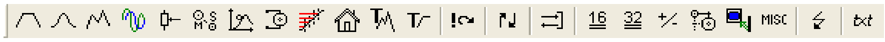
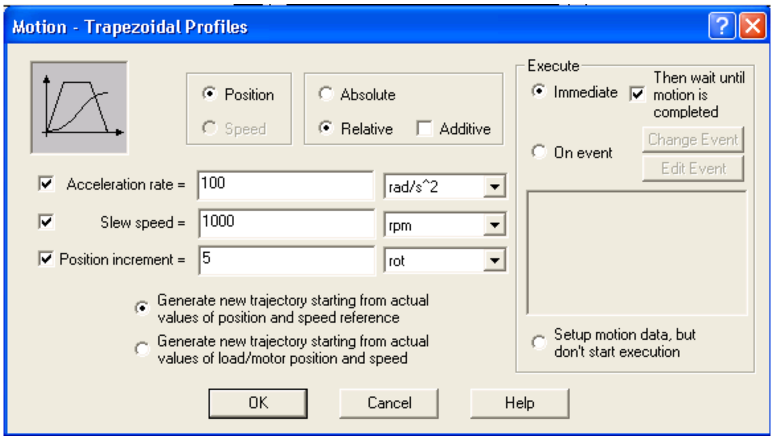


Figure 16.5 *MotionPRO Developer – Motion wizard bar*

Each button represents a new interactive command.



● Position ○ Speed

○ Absolute ● Relative ☐ Additive

☑ Acceleration rate = 100 rad/s^2

☑ Slew speed = 1000 rpm

☑ Position increment = 5 rot

● Generate new trajectory starting from actual values of position and speed reference

○ Generate new trajectory starting from actual values of load/motor position and speed

Execute

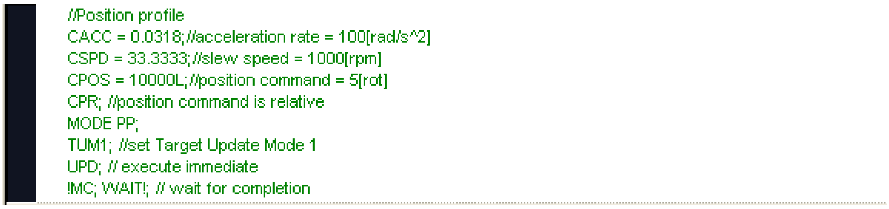
● Immediate ☑ Then wait until motion is completed

○ On event

○ Setup motion data, but don't start execution

Figure 16.6 *MotionPRO Developer – Trapezoidal Profile menu*

After clicking **OK** button, the command is converted into code that will be later downloaded to the drive.



```
//Position profile
CACC = 0.0318;//acceleration rate = 100[rad/s^2]
CSPD = 33.3333;//slew speed = 1000[rpm]
CPOS = 10000L;//position command = 5[rot]
CPR; //position command is relative
MODE PP;
TUM1; //set Target Update Mode 1
UPD; // execute immediate
!MC; WAIT!; // wait for completion
```

Figure 16.7 *MotionPRO Developer – Trapezoidal Profile commands*

Remark: T*he data transfer buttons are not compatible with the EtherCAT® drives.*

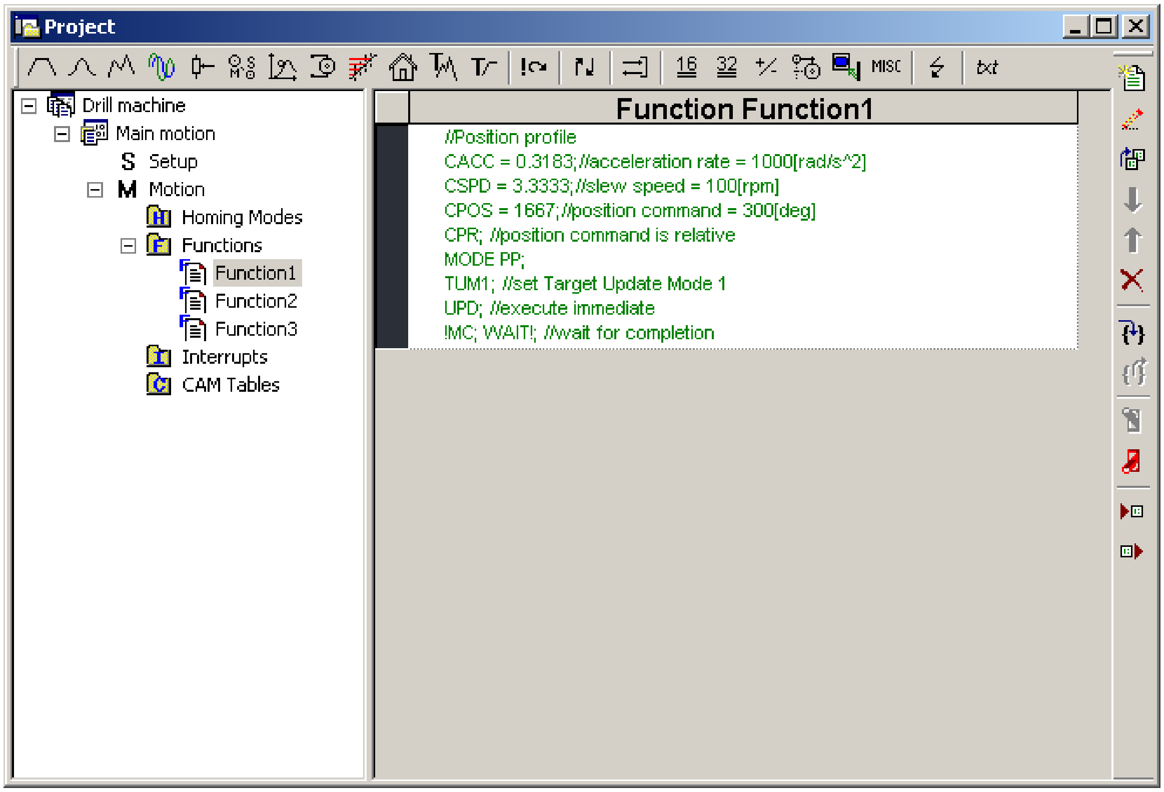
17.2.2 Build MPL functions within MotionPRO Developer

The following steps describes how to create MPL functions with MotionPRO Developer

3. **Define the MPL functions.** Open the MotionPRO Developer project and select the Functions entry from the project tree. On the right side of the project panel add the MPL functions executed by the drive. You may also remove, rename and change the functions download order.

Remark: *You can call up to 10 MPL functions using the associated CANopen object.*

4. **Add the MPL code.** The added functions are listed in the project tree under the **Functions** entry. Select each function from the list and add the MPL code that will be executed by the function. Adding MPL code is presented in **Paragraph 17.2.1.**



```
//Position profile
CACC = 0.3183;//acceleration rate = 1000[rad/s^2]
CSPD = 3.3333;//slew speed = 100[rpm]
CPOS = 1667;//position command = 300[deg]
CPR; //position command is relative
MODE PP;
TUM1; //set Target Update Mode 1
UPD; //execute immediate
!MC; WAIT!; //wait for completion
```

Figure 16.8 *MotionPRO Developer project window – functions edit view*

17.2.3 Downloading MPL program and functions within MotionPRO Developer

Download the MPL 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 MPL code into the drive memory.

17.3 MPL Function Objects

17.3.1 Executing MPL programs

The distributed control concept allows you to prepare and download into a drive a complete MPL program including functions, homing procedures, etc. The MPL program execution can be started simply by writing a value in the dedicated object.

Note: The ElectroCraft EtherCAT® drives, currently do not allow the execution of a PT/PVT motion using MPL code. These types of motions work only via online CoE commands.

17.3.2 Object 2077h: Execute MPL program

This object is used in order to execute the MPL program from either EEPROM or RAM memory. The MPL program is downloaded using the MotionPRO Developer software or by the EtherCAT® master using the .sw file created in MotionPRO Developer.

Writing any value in this object (through the SDO protocol) will trigger the execution of the MPL program in the drive.

Object description:

Index	2077_h
Name	Execute MPL program
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Value range	UNSIGNED16
Default value	-

17.3.2.1 Object 2006h: Call MPL Function

The object allows the execution of a previously downloaded MPL function. When a write is performed to this object, the MPL function with the index specified in the value provided is called. The MPL function body is defined using MotionPRO Developer and saved in the EEPROM memory of the drive. The function index represents an offset in a predefined table of MPL callable functions.

It is not possible to call another MPL function, while the previous one is still running. In this case bits 7 (warning) from the Status Word and 14 (command error) from Motion Error Register are set, and the function call is ignored. The execution of any called MPL function can be aborted by setting bit 13 in Control Word.

There are 10 MPL functions that can be called through this mechanism (the first 10 MPL functions defined using the MotionPRO Developer 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, but no MPL function is defined in that position, an SDO abort code will be issued: 0800 0020h (Data cannot be transferred or stored to the application).

Object description:

Index	2006_h
Name	Call MPL function
Object code	VAR
Data type	UNSIGNED16

Entry description:

Access	WO
PDO mapping	No
Units	-
Value range	1...10
Default value	-

17.4 Loading Automatically Cam Tables Defined in MotionPRO Developer

Apart from the standard modes of operation of CIA 402, the ElectroCraft drives offer 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:

a) The master downloads the cam points into the drive active RAM memory after each power on;

b) The cam points are stored in the drive EEPROM and the master commands their copy into the active RAM memory

c) 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 MotionPRO Developer and are included in the information stored in the EEPROM together with the setup data and the MPL programs/functions.

Remark: *The cam tables are included in the* ***.sw*** *file generated with MotionPRO Developer. Therefore, the master can check the cam presence in the drive EEPROM using the same procedure as for testing of the setup data.*

17.4.1 CAM table structure

The cam tables are arrays of X, Y points, where X is the cam input i.e. the master position and Y is the cam output i.e. the slave position. The X points are expressed in the master internal position units, while the Y points are expressed in the slave internal position units. Both X and Y points 32-bit long integer values. The X points must be positive (including 0) and equally spaced at: 1, 2, 4, 8, 16, 32, 64 or 128 i.e. having the interpolation step a power of 2 between 0 and 7. The maximum number of points for one cam table is 8192.

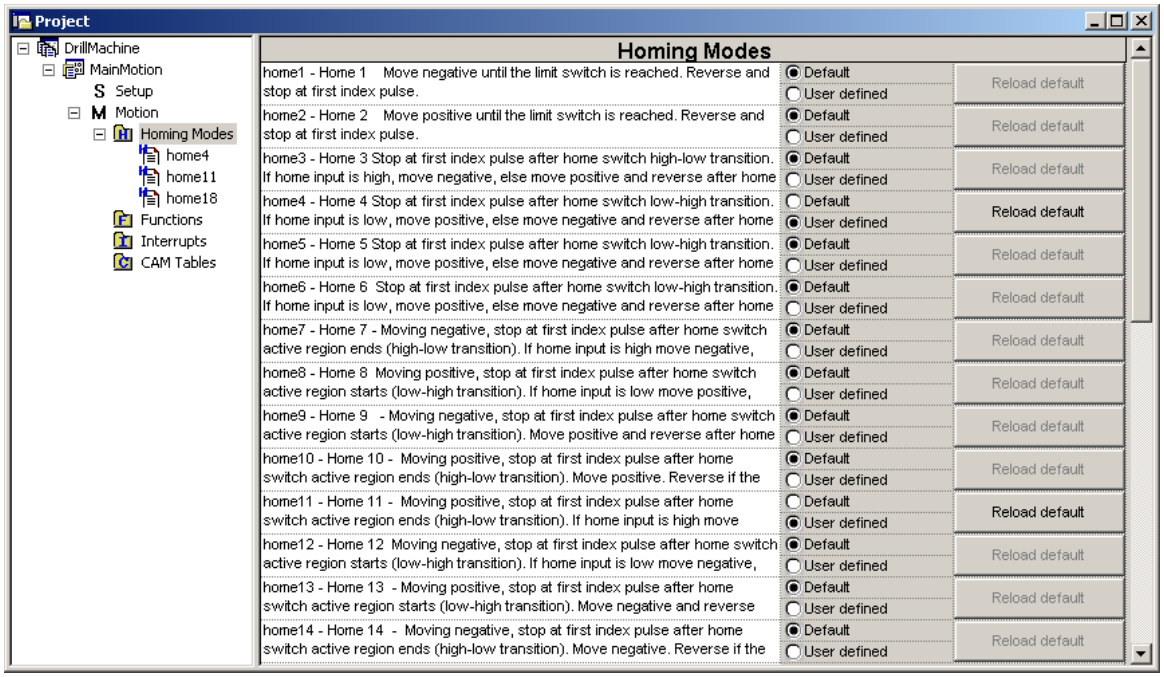
As cam table X points are equally spaced, they are completely defined by two data: the **Master start value** or the first X point and the **Interpolation step** providing the distance between the X points. This offers the possibility to minimize the cam size, which is saved in the drive/motor in the following format:

- 1st word (1 word = 16-bit data):
  - Bits 15-13 – the power of 2 of the interpolation step. For example, if these bits have the binary value 010 (2), the interpolation step is 2^2 = 4, hence the master X values are spaced from 4 to 4: 0, 4, 8, 12, etc.
  - Bits 12-0 – the length -1 of the table. The length represents the number of points (one point occupies 2 words)
- 2nd and 3rd words: the Master start value (long), expressed in master position units. 2nd word contains the low part, 3rd word the high part
- 4th and 5th words: Reserved. Must be set to 0
- Next pairs of 2 words: the slave Y positions (long), expressed in position units. The 1st word from the pair contains the low part and the 2nd word from the pair the high part

Last word: the cam table checksum, representing the sum modulo 65536 of all the cam table data except the checksum word itself.

17.5 Customizing the Homing Procedures

The homing methods defined by the CiA 402 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 MotionPRO Developer 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.



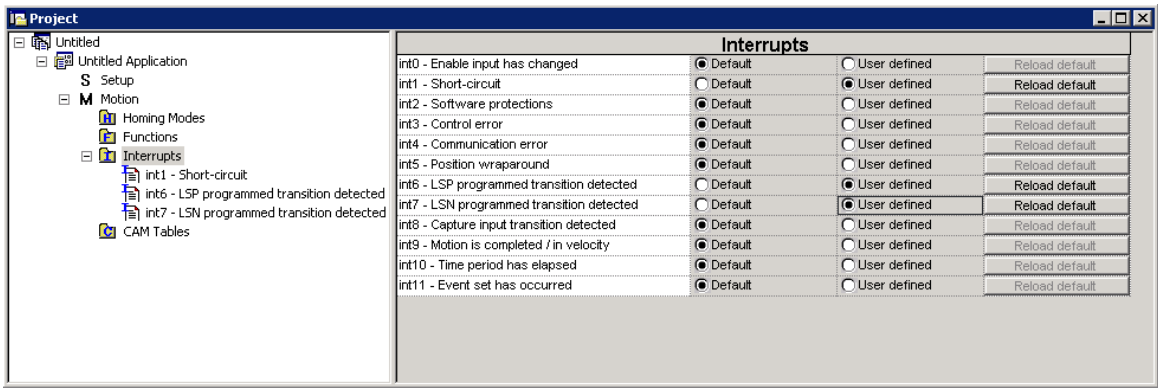
Homing Modes

home1 - Home 1 Move negative until the limit switch is reached. Reverse and stop at first index pulse.	● Default ○ User defined	Reload default
home2 - Home 2 Move positive until the limit switch is reached. Reverse and stop at first index pulse.	● Default ○ User defined	Reload default
home3 - Home 3 Stop at first index pulse after home switch high-low transition. If home input is high, move negative, else move positive and reverse after home	● Default ○ User defined	Reload default
home4 - Home 4 Stop at first index pulse after home switch low-high transition. If home input is low, move positive, else move negative and reverse after home	○ Default ● User defined	Reload default
home5 - Home 5 Stop at first index pulse after home switch low-high transition. If home input is low, move positive, else move negative and reverse after home	● Default ○ User defined	Reload default
home6 - Home 6 Stop at first index pulse after home switch low-high transition. If home input is low, move positive, else move negative and reverse after home	● Default ○ User defined	Reload default
home7 - Home 7 - Moving negative, stop at first index pulse after home switch active region ends (high-low transition). If home input is high move negative,	● Default ○ User defined	Reload default
home8 - Home 8 Moving positive, stop at first index pulse after home switch active region starts (low-high transition). If home input is low move positive,	● Default ○ User defined	Reload default
home9 - Home 9 - Moving negative, stop at first index pulse after home switch active region starts (low-high transition). Move positive and reverse after home	● Default ○ User defined	Reload default
home10 - Home 10 - Moving positive, stop at first index pulse after home switch active region ends (high-low transition). Move positive. Reverse if the	● Default ○ User defined	Reload default
home11 - Home 11 - Moving positive, stop at first index pulse after home switch active region ends (high-low transition). If home input is high move	○ Default ● User defined	Reload default
home12 - Home 12 Moving negative, stop at first index pulse after home switch active region starts (low-high transition). If home input is low move negative,	● Default ○ User defined	Reload default
home13 - Home 13 - Moving positive, stop at first index pulse after home switch active region starts (low-high transition). Move negative and reverse	● Default ○ User defined	Reload default
home14 - Home 14 - Moving negative, stop at first index pulse after home switch active region ends (high-low transition). Move negative. Reverse if the	● Default ○ User defined	Reload default

If you click on the *HomeX* procedure, on the right side you'll see the MPL function implementing it. The homing routine can be customized according to your application needs. Its calling name and method remain unchanged.

17.6 Customizing the Drive Reaction to Fault Conditions

Similarly to the homing modes, the default service routines for the MPL interrupts can be customized according to your application needs. However, as most of these routines handle the drive reaction to fault conditions, <u>it is mandatory to keep the existent functionality while adding your application needs, in order to preserve the correct protection level of the drive</u>. The procedure for modifying the MPL interrupts is similar with that for the homing modes.



- Untitled
  - Untitled Application
    - S Setup
    - M Motion
      - Homing Modes
      - Functions
      - Interrupts
        - int1 - Short-circuit
        - int6 - LSP programmed transition detected
        - int7 - LSN programmed transition detected
      - CAM Tables

Interrupts

int0 - Enable input has changed	● Default	○ User defined	Reload default
int1 - Short-circuit	○ Default	● User defined	Reload default
int2 - Software protections	● Default	○ User defined	Reload default
int3 - Control error	● Default	○ User defined	Reload default
int4 - Communication error	● Default	○ User defined	Reload default
int5 - Position wraparound	● Default	○ User defined	Reload default
int6 - LSP programmed transition detected	○ Default	● User defined	Reload default
int7 - LSN programmed transition detected	○ Default	● User defined	Reload default
int8 - Capture input transition detected	● Default	○ User defined	Reload default
int9 - Motion is completed / in velocity	● Default	○ User defined	Reload default
int10 - Time period has elapsed	● Default	○ User defined	Reload default
int11 - Event set has occurred	● Default	○ User defined	Reload default

17.7 Saving the program, functions and setup to an image file (.sw)

Saving the motion, functions and setup to an image file, allows the EtherCAT® master to download them later to the drive(s) using CoE commands without the need for MotionPRO Developer software.

Use the menu command menu command **Application | EEPROM Programmer File | Motion and Setup…** to the MPL code and setup as a .sw file.

To download an image file (.sw) to the drive, please read **Paragraph 16.4**.

Appendix A Object Dictionary by Index

Index	Sub-index	Description
1000h	00h	Device type
1001h	00h	Error register
1002h	00h	Manufacturer status register
1008h	00h	Manufacturer device name
1009h	00h	Manufacturer Hardware Version
100Ah	00h	Manufacturer software version
1018h		Identity Object
	00h	Number of entries
	01h	Vendor ID
	02h	Product Code
	03h	Revision Number
	04h	Serial number
1600h		RPDO1 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6040h – controlword
	02h	2nd mapped object – 6060h – modes of operation
1602h		RPDO2 mapping parameters
	00h	Number of entries
	02h	1st mapped object – 607Ah – target position
1602h		RPDO3 mapping parameters
	00h	Number of entries
	02h	1st mapped object – 6071h – target torque
1603h		RPDO4 mapping parameters
	00h	Number of entries
	02h	1st mapped object – 60FFh – target velocity
1A00h		TPDO1 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 6041h – statusword
1A01h		TPDO2 mapping parameters
	00h	Number of entries
	02h	1st mapped object – 6061h – modes of operation display
1A02h		TPDO3 mapping parameters
	00h	Number of entries
	02h	1st mapped object – 6064h – position actual value
1A03h		TPDO4 mapping parameters
	00h	Number of entries
	01h	1st mapped object – 606Ch – velocity actual value
1C00h		Sync Manager Com. Type
	00h	Number of entries

	01h	Communication Type Sync Manager 0
	02h	Communication Type Sync Manager 1
	03h	Communication Type Sync Manager 2
	04h	Communication Type Sync Manager 3
1C12h		Sync Manager Channel 2 (Process Data Output)
	00h	Number of entries
	01h	PDO Mapping object index of assigned RxPDO : 1st mapped object
	02h	PDO Mapping object index of assigned RxPDO : 2nd mapped object
1C13h		Object 1C13h: Sync Manager Channel 2 (Process Data Input)
	00h	Number of entries
	01h	PDO Mapping object index of assigned TxPDO : 1st mapped object
	02h	PDO Mapping object index of assigned TxPDO : 2nd mapped object
2000h	00h	Motion Error Register
2001h	00h	Motion Error Register mask
2002h	00h	Detailed Error Register
2005h	00h	Max slippage time out
2006h	00h	Call MPL function
2012h	00h	Master resolution
2013h		EGEAR multiplication factor
	00h	Number of entries
	01h	EGEAR ratio numerator (slave)
	02h	EGEAR ratio denominator (master)
2017h	00h	Master actual position
2018h	00h	Master actual speed
2019h	00h	CAM table load address
201Ah	00h	CAM table run address
201Bh	00h	CAM offset
201Dh	00h	External reference type
201Eh	00h	Master position
2022h	00h	Control effort
2023h	00h	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 a MPL 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	Time factor Numerator
	02h	Time factor 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 MPL 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
207Fh	00h	Current limit
2080h	00h	Reset drive
2081h	00h	Set/Change actual position
2082h	00h	Sync on fast loop
2083h	00h	Encoder resolution
2084h	00h	Stepper resolution
2085h	00h	Position triggered outputs
2086h	00h	Limit speed/acceleration for CSP/CSV

2087h	00h	Actual internal velocity from sensor on motor
2088h	00h	Actual internal position from sensor on motor
2089h	00h	Synchronization test config
208Ah	00h	Save setup status
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
2100h	00h	Number of Steps per Revolution
2101h	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
6007h	00h	Abort connection option code
6040h	00h	Controlword
6041h	00h	Statusword
605Ah	00h	Quick stop option code
605Bh	00h	Shutdown option code
605Ch	00h	Disable operation option code
605Dh	00h	Halt 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 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
6069h	00h	Velocity sensor actual value
606Bh	00h	Velocity demand value
606Ch	00h	Velocity actual value
606Fh	00h	Velocity threshold
6071h	00h	Target torque

6077h	00h	Torque actual value
607Ah	00h	Target position
607Ch	00h	Home offset
607Dh		Software position limit
	00h	Number of entries
	01h	Minimal position limit
	02h	Maximal position limit
6081h	00h	Profile velocity
6083h	00h	Profile acceleration
6085h	00h	Quick stop deceleration
6086h	00h	Motion profile type
6089h	00h	Position notation index
608Ah	00h	Position dimension index
608Bh	00h	Velocity notation index
608Ch	00h	Velocity dimension index
608Dh	00h	Acceleration notation index
608Eh	00h	Acceleration dimension index
6093h		Position factor
	00h	Number of entries
	01h	Numerator
	02h	Divisor
6094h		Velocity encoder factor
	00h	Number of entries
	01h	Numerator
	02h	Divisor
6097h		Acceleration factor
	00h	Number of entries
	01h	Numerator
	02h	Divisor
6098h	00h	Homing method
6099h	00h	Homing speeds
609Ah	00h	Homing acceleration
60B8h	00h	Touch probe function
60B9h	00h	Touch probe status
60BAh	00h	Touch probe 1 positive edge
60BBh	00h	Touch probe 1 negative edge
60BCh	00h	Touch probe 2 positive edge
60BDh	00h	Touch probe 2 negative edge
60C0h	00h	Interpolation sub mode select
60C1h		Interpolation Data Record

	00h	Number of entries
	01h	The first parameter
	0nh	The n-th parameter
60C2h		Interpolation time period
	00h	Number of entries
	01h	Interpolation time period value
	02h	Interpolation time index
60F4h	00h	Following error actual value
60F8h	00h	Max slippage
60FCh	00h	Position demand value
60FDh	00h	Digital inputs
60FEh	00h	Digital outputs
60FFh	00h	Target velocity
6502h	00h	Supported drive modes

ElectroCraft®

powering innovation™