



Ether**CAT**® 

EtherCAT on AccurET

User's Manual

Version F

ETEL

THIS PAGE IS INTENTIONALLY LEFT BLANK

Table of contents

1	Introduction	8
1.1	Safety	8
1.1.1	Principle	8
1.2	General operating conditions	9
1.3	EtherCAT availability	9
1.4	AccurET configuration for EtherCAT	9
1.4.1	Caution	9
1.5	AccurET's features restrictions	10
2	Electrical interface	11
2.1	Communication connectors	11
2.1.1	USB communication (connector X00)	11
2.1.2	EtherCAT input (connector X01)	11
2.1.3	EtherCAT output (connector X02)	11
3	EtherCAT communication	12
3.1	EtherCAT vs TransnET	12
3.2	EtherCAT technical overview	12
3.3	EtherCAT frame	13
3.4	EtherCAT explicit device identification	13
3.5	CANOpen over EtherCAT (CoE)	14
3.5.1	CoE device architecture	14
3.5.2	CoE datagrams	15
3.6	PDO mapping	15
3.6.1	Setup procedure of PDO mapping	16
3.6.2	Default PDO mappings	16
3.6.3	Configurable PDO Mapping	17
3.6.4	Related objects	17
3.7	State machine	18
3.8	Synchronization with distributed clock	18
3.8.1	Setup DC mode	19
3.8.2	Setup FreeRun	19
3.8.3	Related objects	20
3.9	EtherCAT master	21
3.10	AccurET setting	22
3.10.1	General description	22
3.10.2	Related objects	22
4	Application	23

4.1	Controller configuration	23
4.1.1	General description	23
4.1.2	Related objects	23
4.2	Device control	24
4.2.1	General description	24
4.2.2	Related objects	25
4.3	Modes of operation	28
4.3.1	General description	28
4.3.2	Related objects	29
4.4	Profile Position mode	29
4.4.1	General description	29
4.4.2	Related objects	32
4.5	Cyclic Synchronous Position mode	35
4.5.1	General description	35
4.5.2	Related objects	36
4.6	Homing	38
4.6.1	General description	38
4.6.2	Related objects	39
4.7	Sequence management	42
4.7.1	General description	42
4.7.2	Related objects	42
4.8	Digital inputs outputs management	43
4.8.1	General description	43
4.8.2	Related objects	43
4.9	Touch Probe	44
4.9.1	Related objects	44
4.9.2	Example	48
4.10	Controller register access	49
4.10.1	General description	49
4.10.2	Related objects	49
4.11	RTV equivalent	51
4.11.1	General description	51
4.11.2	Related objects	51
4.12	Information and diagnosis data	53
4.12.1	General description	53
4.12.2	Related objects	53
4.13	Other objects	54
4.13.1	General description	54
4.13.2	Related objects	54
5	Object dictionary	56

6	Appendixes	62
6.1	Configuration	62
6.1.1	AccurET configuration for EtherCAT	62
6.1.2	Controlword and statusword source	62
6.1.3	TwinCAT configuration	63
6.1.4	Cyclic synchronous position mode	64
6.1.5	Profile position mode	66
6.1.6	Create custom PDO	67
6.1.7	RTV configuration	68
6.1.8	SDO / PDO access with TwinCAT through a C application	70
6.2	ETEL library	73
6.3	Errors reference list	74
6.4	Units conversion	74
6.5	EtherCAT certification	75
7	Service and support	76

THIS PAGE IS INTENTIONALLY LEFT BLANK

Record of revisions:

Document revisions		
Version	Date	Main modifications
Ver A	30.11.18	First version
Ver B	11.06.19	Updated version: - New Touch Probe function (refer to §4.9)
Ver C	05.12.19	Updated version: - AccurET configuration updated (refer to §1.4.1) - EtherCAT explicit device identification (refer to §3.4) - PDO mapping updated (refer to §3.6) - Configurable PDO Mapping (refer to §3.6.3) - RTV equivalent (refer to §4.11) - Object dictionary updated (refer to §5) - Create custom PDO (refer to §6.1.6) - RTV configuration (refer to §6.1.7)
Ver D	07.09.20	Updated version: - New object: Max motor speed (refer to §4.4.2.6)
Ver E	25.01.21	Updated version: - New related object (refer to §4.13.2) - Register K375 (refer to §6.1.2)
Ver F	10.09.21	Updated version: - Minor changes (refer to modification strokes)

Documentation concerning EtherCAT on AccurET:

- | | |
|--|---|
| <ul style="list-style-type: none"> • EtherCAT on AccurET • AccurET Modular Operation & Software Manual • AccurET Modular 48 Hardware Manual • AccurET Modular 300 Hardware Manual • AccurET Modular 400-600 Hardware Manual • ComET User's Manual | <ul style="list-style-type: none"> • AccurET programming with EtherCAT • AccurET setup, use & programming manual • Specifications & electrical interfaces • Specifications & electrical interfaces • Specifications & electrical interfaces • Commissioning software |
|--|---|

Acronyms

- **ADS:** Automation Device Specification
- **CoE:** CANOpen over EtherCAT
- **CSP:** Cyclic Synchronous Position
- **DC:** Distributed Clock
- **ESC:** EtherCAT Slave Controller
- **EtherCAT** Ethernet for Control Automation Technology
- **LSB:** Less significant BYTE
- **MLTI:** Manager Loop Time Interrupt
- **MSB:** Most significant BYTE
- **PDO:** Process Data Object
- **PLC:** Programmable Logic Controller
- **PP:** Profile Position
- **SDO:** Service Data Object

1 Introduction

This document concerns the digital position controller of ETEL's AccurET family also called 'controller' in this document.

The purpose of this manual is to give details about the use of EtherCAT protocol in AccurET controllers. EtherCAT is a standardized fieldbus system for real-time application in automation field.

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



For more information about functioning, installation, tuning, functions and programming possibilities, refer to the '**AccurET Operation & Software Manual**'.

For electrical specifications, interfaces and hardware items, refer to the corresponding '**AccurET Hardware Manual**'.

Remark: The updates between two successive versions are highlighted with a modification stroke in the margin of the manual.



1.1 Safety

1.1.1 Principle

	The user must have read and understood this documentation as well as those listed page 7 before carrying out any operation on the system. Please contact ETEL S.A. or authorized distributors in case of missing information or question regarding the installation procedures, safety or any other issue.
	ETEL S.A. disclaims all responsibility for accidents and damages if the safety instructions, the procedures and the usage described in this manual are not followed (including the ones given in the manuals listed in page 7).

- Never use the controller in operating conditions and for purposes other than those described in this manual.
- A competent and trained technician must install and operate the controller, in accordance with all specific regulations of the respective country concerning both safety and EMC aspects.
- High voltage may be present on the power and motor connectors.
- The customer must provide at all time the appropriate protections against electrical hazard and moving parts of the connected system. Operating the controller will make the motor move.
- Before connecting or disconnecting a cable on one of these connectors or touching the controller, turn off all the power supplies and wait 5 minutes to allow the internal Direct Circuit bus capacitors to discharge.
- In the controller, the leakage current through the protective conductor to the GND is greater than a.c. 3.5mA.
- The safety symbols placed on the controller or written in the manuals ([page 7](#)) must be respected.
- If the controller is integrated into a machine, the manufacturer of this machine must establish that it fulfills the 2004/108EC directive on EMC before operating the controller.

	Signals a danger of electrical shock to the operator Can be fatal for a person.
--	--

	Signals a danger for the controller. Can be destructive for the material. A danger for the operator can result from this.
	Indicates electrostatic discharges (ESD), dangerous for the controller. The components must be handled in an ESD protected environment only.

1.2 General operating conditions

The controller is designed to operate in a non-aggressive and clean environment, with a humidity rate ranging between 10% and 85%, an altitude < 2000m (6562 ft), and a temperature ranging between +15°C (59°F) and +40°C (104°F). This controller must be connected to an electrical network of overvoltage category 3 (refer to EN 61800-5-1 and UL 840 standards for more information) and is suitable for use on a circuit capable of delivering not more than 5000 Arms, symmetrical amperes, 400 volts maximum. The electronics must be in an enclosure respecting a pollution degree of 2 (refer to UL 508C and EN 61800-5-1 standards for more information). The controller is not designed or intended for use in the on-line control of air traffic, aircraft navigation and communications as well as critical components in life support systems or in the design, construction, explosive atmosphere, operation and maintenance of any nuclear facility.

1.3 EtherCAT availability

AccurET comes in two hardware versions:

- One dedicated to TransnET protocol only, with no EtherCAT compatibility.
(TransnET is the ETEL proprietary protocol)
- One dedicated to both EtherCAT and TransnET protocols.
This version (with firmware 3.16A or above) is hardware signed and is able to switch to EtherCAT protocol at boot-up.

The AccurET Modular which support EtherCAT capability are:

- AccurET Modular 48
- AccurET Modular 300
- AccurET Modular 400
- AccurET Modular 600

1.4 AccurET configuration for EtherCAT

AccurET for EtherCAT version is plug-and-play and can be used directly with any master following the EtherCAT protocol. All basic EtherCAT functionalities have been implemented and compatibility with EtherCAT is ensured.

To make AccurET controller working with EtherCAT, two conditions must be met:

- The signature or P/N of the controller **must have suffix 2**. For example, the product number must look like: 828309-**02**
- The register **C1** must be set to **2**. Do not forget to save the parameter with **SAV.x=4** and reset the controller with **RSD.x=255** for taking changes into account (refer to [§6.1.1](#) for more information).

1.4.1 Caution

Since the MLTI is working at 500 µs in EtherCAT mode and not at 400 µs (TransnET and TCP/IP mode), registers with ISO conversion (speed, acceleration, etc.) based on the MLTI period will have different values at 500 µs MLTI than at 400 µs MLTI when converted in increments.

If register C1=2 only the USB connection can be used for drive commissioning and MLTI period is 500 µs.

If TransnET or TCP/IP is used for controller setting (then C1=0 or C1=1), the MLTI period is 400 µs and registers set using ISO values during commissioning will be converted into increments using a 400 µs MLTI.

This will impact drive setting if the user switches to EtherCAT mode, with C1=2, without modifying the registers value to take into account for a different MLTI period of 500 µs.

To update registers for a 500 μ s MLTI:

1. Upload all register in ISO format to host PC.
2. Open and modify C1 to 2 in the register file.
3. Download modified file to AccurET EtherCAT

1.5 AccurET's features restrictions

Several standard features of AccurET are not available in EtherCAT mode. The missing features are:



- Triggers
- Single and Dual Axis Force Control
- RTV
- Gantry Control (level 1 and 2)
- Stage Mapping

If an application requires one of these features, it is mandatory to use TransnET (C1=0) as main communication bus, or TCP/IP (C1=1) if the Triggers or Force Control are needed.

In addition, it is worth mentioning that the EtherCAT CiA-402 limits the position on 32 bits. If a 48-bit resolution is required, TransnET or TCP/IP must be used.

2 Electrical interface

2.1 Communication connectors

	Signals are not insulated from Protective Earth (PE).
	The communication connectors must be handled in an ESD protected environment.

The communication between a host (PC) and a controller is obtained via USB protocol (connector X00). The USB communication is the one which must be used to commission an AccurET with ComET tool when the EtherCAT communication is established. To establish the EtherCAT communication, the connectors X01 and X02 are used to make a daisy chain with standard RJ-45 cables.

For more information about available connectors, refer to the Hardware Manual of the relevant **AccurET Modular XXX** ([page 7](#)).

2.1.1 USB communication (connector X00)

USB 2.0 (full speed) communication is used for controller setting and monitoring. The USB connector is a «Type B» connector.

2.1.2 EtherCAT input (connector X01)

The EtherCAT input, labeled «COM IN» (COMmunication INput) on the front panel of the controller, is used to connect the input cable of the EtherCAT communication. The Ethernet connection is used to directly connect a master (PC) to a single controller.

Remark: The RJ-45 cable must meet the following characteristics: 1:1 shielded cable, category 5e according to EN50173 or ISO/IEC 11801 using 4 wires for signal transfer. The distance between two EtherCAT nodes (master to AccurET and AccurET to AccurET) is 100m maximum.

2.1.3 EtherCAT output (connector X02)

The EtherCAT output, labeled «COM OUT» (COMmunication OUTput) on the front panel of the controller is used to connect the output cable of EtherCAT communication. For the last controller, this connector is not used as incoming and outgoing data run through the same cable.

Remark: The RJ-45 cable must meet the following characteristics: 1:1 shielded cable, category 5e according to EN50173 or ISO/IEC 11801 using 4 wires for signal transfer. The distance between two EtherCAT nodes (master to AccurET and AccurET to AccurET) is 100m maximum.

3 EtherCAT communication

3.1 EtherCAT vs TransnET

Any device wishing to communicate with the outside world needs to define a communication protocol between itself and remote peers. This is always merely a way to organize bytes sent over the network cables so both end of the communication can understand each other. Every message consists of bytes packed and orderly sent on the wire in a structure called frame. EtherCAT and ETEL's TransnET are no different to that.

TransnET is an ETEL proprietary protocol and so bytes order traveling the network and their meaning is not documented neither released outside of ETEL premises.

To the contrary EtherCAT is also a proprietary protocol but has been standardized and is now open and documented to any users wishing to create their own device supporting it or create their own master to manage any EtherCAT slaves.

In EtherCAT world, controllers (also named drives) are called slaves (or sometimes stations). There can be up to 65535 slaves but exactly one master is needed. This master can be a dedicated embedded PC, a PLC, a real-time software master or any device complying with EtherCAT masters specifications. Each EtherCAT AccurET controller is a single controller that can drive two motors.

Within TransnET configuration, the master is an ETEL UltimET. It is possible to link up to 31 TransnET AccurET (i.e controlling 62 motors).

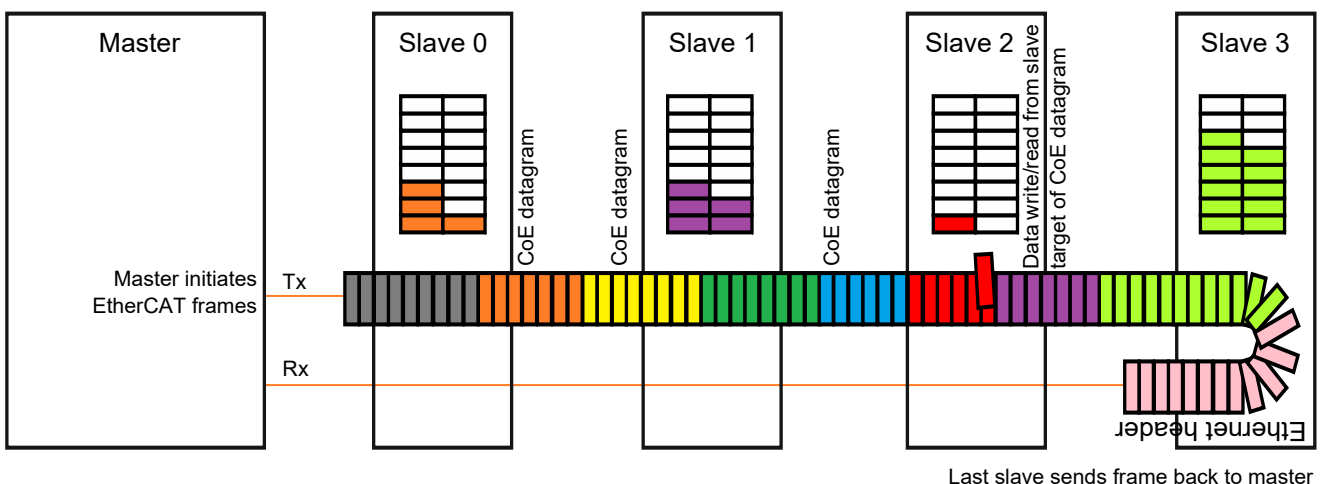
EtherCAT AccurET controllers can also be used stand-alone the same way a TransnET AccurET can be used without UltimET or PC. In this case though, all functionalities have to be programmed using ETEL sequences and saved in the controller memory. No EtherCAT communication is used.

3.2 EtherCAT technical overview

EtherCAT basic principle are EtherCAT frames created at the master level, sent over the network to the first slave, which passes them to the next slave, until frames reach the last slave which sends them back to the master as shown in the following figure for one frame. Only the master can initiate a new frame.

Every slave is free to read data from a frame or modify the frame by adding information to it. It cannot remove information added by other slaves.

Every slave connected to an EtherCAT network receives EtherCAT frames and act upon accordingly to user data or requests.



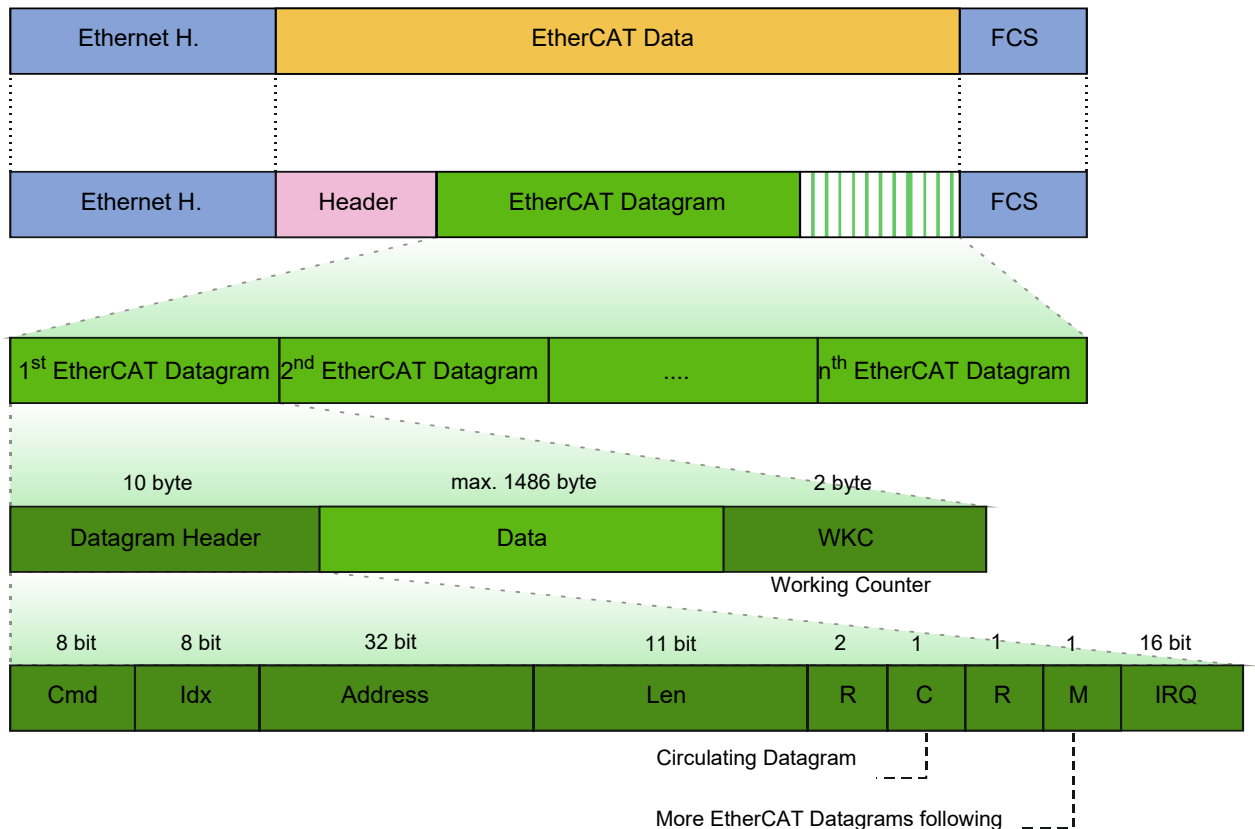
Note that EtherCAT frames are broadcasted to all slaves on the same network. Slave Controllers read every CoE datagrams, pick up only CoE objects that target them (see on the figure the red datagram dedicated to Slave 2) and forward the frame to the next node. The slave may have written some additional PDO or SDO data to the frame before it moves to the next node.

It is important to understand that EtherCAT protocol is not a way to dialog with a controller but a way to distribute data to a controller. Using a freight rail network analogy Ethernet would be the tracks, EtherCAT frames would be the always on time trains, and datagrams are the freight.

3.3 EtherCAT frame

EtherCAT frames are structured as shown in the following figure. Slaves all have an EtherCAT Slave Controller (ESC) ASIC or FPGA whose task is to take care of all EtherCAT formatting: Ethernet Header, EtherCAT header, EtherCAT datagrams header, EtherCAT datagrams data, frame checksum (FCS). Space dedicated to user data and requests is made of so called **EtherCAT Datagrams** which are short messages targeting one EtherCAT slave.

EtherCAT Datagrams are short messages with their own headers and specific commands whose task is only to get/deliver data payload from/to targeted slaves by the master.



3.4 EtherCAT explicit device identification

The use of EtherCAT device identification is to identify an EtherCAT slave explicitly. This is necessary for the following use cases:

- Hot connect applications
Within some applications it might be useful to connect or disconnect parts of the network. In this case the master must have the possibility to identify which part of the network is available.
- Prevention against cable swapping
If at least two identical devices are used in one application, it might be necessary to prevent the mix-up of these devices by cable swapping. Example: Within a machining center there might be two identical drives to work in X and Y direction. To avoid that the drives receive wrong process data, for example after a device replacement, an explicit identification of the devices can be used.

For setting EtherCAT device identification, the user must either set the DIP switch or set the ESC configured station alias register (0x0012).

DIP switch	ESC configured station alias register (0x0012)	Comment
0	0	Free to the master to choose an address
0	1..FFFFh	The master must use the address set in register
1..63	0	The master must use the address set with DIP switch
1..63	1..FFFFh	The slave generates an error

Caution: Configured station alias register is saved in the AccurET register C375, which is in persistent flash memory. Changing configured station alias register at CoE level will trigger a write of C375 to flash memory. Since all Cx registers are stored in one single flash memory block, writing to C375 will also save all others Cx registers. To avoid unintentionally saving Cx registers modified only for testing purposes, do not parametrize AccurET registers and configured station alias at the same time.

Remark: In TransnET and TCP/IP modes, there are two ways to set axes number:

- DIP switch
- AXI command

The address set with AXI command is used when all switches are set to 1, otherwise the address set with the DIP switch is used.

When USB communication is used the axis number seen by the user will be defined by:

- DIP switch or AXI command as defined above in TransnET and TCP/IP modes.
- AXI command in EtherCAT mode.

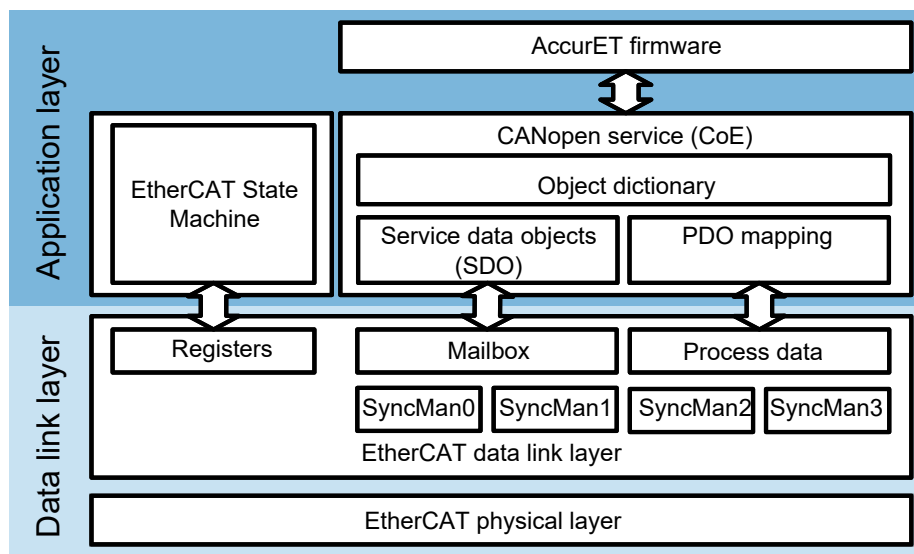
For more information about axes numbering, refer to the “Operation & Software Manual”.

3.5 CANOpen over EtherCAT (CoE)

EtherCAT can carry many embedded protocols as datagrams, Sercos over EtherCAT (SoE), CANOpen over EtherCAT (CoE), FSoE, etc. Only CoE datagrams can be used with AccurET.

3.5.1 CoE device architecture

The following figure shows the device architecture of the AccurET CANOpen over EtherCAT (CoE).



In the controller, EtherCAT network model is composed of two parts:

- **Application layer** is CANopen (CiA402) communication protocol. It provides a set of objects which represent controller's parameters, application data and so on. The master access these objects to set the controller working on the desired way. The next chapters will provide more details about the PDO mapping, the SDO, the State Machine and the objects dictionary.

- **Data link layer** is in charge of the data transmission. This layer is not described in this manual. Refer to the EtherCAT documentation for more information.

3.5.2 CoE datagrams

The communication to the AccurET objects can be achieved through two services:

- **Process data object (PDO)**
PDO are data read / written cyclically by the AccurET from / to an EtherCAT frame, and synchronized with EtherCAT Distributed Clock (DC) (refer to [§3.8](#) for more information).
This allows to process data such as real position, tracking error, new target position being continuously shared over the EtherCAT network between a slave and the master only. Direct communication between two slaves is not available.
- **Mailbox communication (SDO)**
SDO (Service Data Object) are asynchronous datagrams that allow to access any object from the CoE Object Dictionary. This service is acyclic and cannot be synchronized neither with PDO data nor with distributed clock. These objects can be read / written and are accessed with an index address and a sub-index number. Every AccurET functionalities must be addressed using a CoE object. For more information about the addresses and documentation related to CoE objects, refer to [§5](#)

For example, to read an AccurET register (steps below are simplified, refer to [§4.10](#) for full control steps), do as follows:

1. From the master, use CoE object with index address 4008h. Fill requested register number and type in sub-indexes 5 and 4. Send a SDO datagram over EtherCAT.
2. Wait to receive a SDO datagram with 4008h CoE object. Sub-indexes 7 - 8 are the data part filled with register content.

AccurET provides two EtherCAT modules (0 and 1) which represent respectively axes 0 and 1. The objects can be either 'controller specific' or 'module specific':

- 'controller specific' means it concern the AccurET device (axes 0 and 1)
- 'module specific' means the object is for module 0 (axis 0) and the user must add an offset to access to the equivalent object of module 1 (axis 1).

In the current documentation, all objects are described for module 0. To get objects for module 1 (axis 1), the user must add 800h to each SDO object and 10h to each PDO mapping.

3.6 PDO mapping

PDO mapping defines the relation from object dictionary to PDO's application objects (real-time process data). The user can choose among defined PDO. To configure PDO user's application specific, uses PDO 160F and 1A0F which can be modified as user wish.

In AccurET, several PDOs are available which are:

RxPDO	TxPDO	Definition
1600h	1A00h	Profile Position (PP) mode motion
1601h	1A01h	Cyclic Synchronous Position (CSP) mode motion
1602h	1A02h	Sequence management
1603h	1A03h	Digital IO management
1604h	1A04h	Fast Digital IO management
	1A05h	Motion Information
1605h		Feed Forward
1706h	1B06h	CSP mode motion with latch capability (used for TwinCAT homing compatibility)
1607h	1A07h	Touch Probe management

RxPDO	TxPDO	Definition
1608h	1A08h	Modes of operation
...
160Fh	1A0Fh	Custom definition

These PDOs are 'module specific' and only for module 0 (axis 0). To get PDO for module 1 (axis 1) the developer must add 10h to base PDO.

This is not applied to PDOs 1604h and 1A04h since these PDOs are 'controller specific'.

3.6.1 Setup procedure of PDO mapping

To initiate the controller with PDO, follow the procedure:

- Disable the assignment of the Sync manager with 1C12h:00 = 0 and 1C13h:00 = 0
- Set all sub-objects of 1C12h and 1C13h to the desired PDO
- Enable the assignment of the Sync manager with 1C12h:00 = number of added PDOs in objects in 1C12h
- Enable the assignment of the Sync manager with 1C13h:00 = number of added PDOs in objects in 1C13h

3.6.2 Default PDO mappings

The following table shows the default PDO configuration of the module 0. To get PDO for module 1 (axis 1) the developer must add 10h to base PDO address. To get object contained in PDO for module 1 (axis1) the developer must add 800h to object address.

PDO	Mapping
RxPDO (1600h)	Control Word (6040h)
	Target Position (607Ah)

PDO	Mapping
RxPDO (1601h)	Control Word (6040h)
	Target Position (607Ah)

PDO	Mapping
RxPDO(1602h)	Sequence Control Word (4004h)

PDO	Mapping
RxPDO(1603h)	Digital Outputs Demanded (4007h: 02h)

PDO	Mapping
RxPDO(1604h)	Fast Digital Outputs Demanded (4006h: 02h)

PDO	Mapping
RxPDO(1605h)	Torque Offset (60B2h)

PDO	Mapping
TxPDO (1A00h)	Status Word (6041h)
	Actual Position (6064h)

PDO	Mapping
TxPDO (1A01h)	Status Word (6041h)
	Actual Position (6064h)

PDO	Mapping
TxPDO (1A02h)	Sequence Status Word (4003h)

PDO	Mapping
TxPDO (1A03h)	Digital Inputs (4007h: 01h)
	Digital Outputs Actual (4007h: 03h)

PDO	Mapping
TxPDO (1A04h)	Fast Digital Inputs (4006h: 01h)
	Fast Digital Actual (4006h: 03h)

PDO	Mapping
TxPDO (1A05h)	Velocity Actual Value (606Ch)
	Current Actual Value (6078h)
	Torque Actual Value (6077h)
	Following Error Actual Value (60F4h)
	Position demand internal value (60FCh)

PDO	Mapping
RxPDO (1706h)	Control Word (6040h)
	Target Position (607Ah)
	Latch Control Word (2802h)

PDO	Mapping
RxPDO (1607h)	Touch Probe
	Touch Probe Function (60B8h)

PDO	Mapping
RxPDO (1608h)	Mode Of Operation
	Mode Of Operation (6060h)

PDO	Mapping
TxPDO (1B06h)	Status Word (6041h)
	Actual Position (6064h)
	Latch Status Word (2901h)
	Latch Position (2902)

PDO	Mapping
TxPDO (1A07h)	Touch Probe
	Touch Probe Function(60B8h)
	Touch Probe Status (60B9h)
	Touch Probe 1 Positive Edge (60BAh)
	Touch Probe 1 Negative Edge (60BBh)
	Touch Probe 2 Positive Edge (60BCh)
	Touch Probe 2 Negative Edge (60BDh)
	Touch Probe 1 Positive Edge Counter (60D5h)
	Touch Probe 1 Negative Edge Counter (60D5h)
	Touch Probe 2 Positive Edge Counter (60D7h)
	Touch Probe 2 Negative Edge Counter (60D8h)

PDO	Mapping
TxPDO (1A08h)	Mode Of Operation
	Mode Of Operation Display (6061h)
	Supported Drive Modes (6502h)

3.6.3 Configurable PDO Mapping

The PDOs RxPDO (160Fh) and TxPDO (1A0Fh) are ARRAY where the user can set up to 32 CoE objects. The user can use these objects to create a full custom PDO.

PDO	Mapping
RxPDO (160Fh)	User selection [1]
	...
	User selection [32]

PDO	Mapping
TxPDO (1A0Fh)	User selection [1]
	...
	User selection [32]

3.6.4 Related objects

To configure the controller to work with one or more PDO, objects 1C12h and 1C13h must be set. These objects represent an array of selected PDO the application demands to work with.

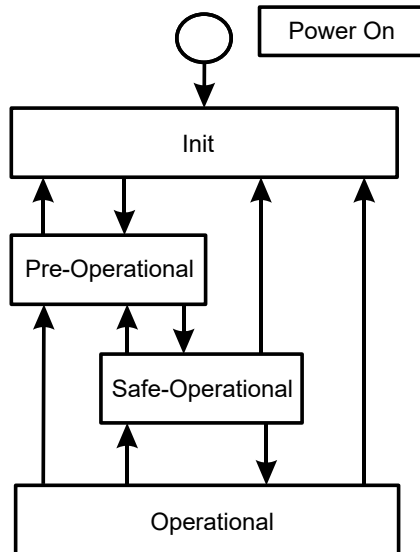
These objects are 'controller specific'.

Index	Sub-Index	Name	Data Type	Access	Value	Example
1C12h	00h	RxPDO assign			Nb RxPDO	2
	01h	RxPDO 1	UINT16	RW		1600h
	02h	RxPDO 2	UINT16	RW		1603h
	...		UINT16	RW		
1C13h	00h	TxPDO assign			Nb TxPDO	
	01h	TxPDO 1	UINT16	RW		1A00h
	02h	TxPDO 2	UINT16	RW		1A03h
	...		UINT16	RW		

Added PDOs is application dependent. It is advised to only add required PDOs to optimize the EtherCAT frame size.

3.7 State machine

The following figure shows the state machine used to coordinate the EtherCAT communication between the master and the slaves. State switching is normally initiated by the master. The slaves can deny state switching if some errors are detected.



State	Description
Init	<ul style="list-style-type: none"> No mailbox communication No PDO communication
Pre-Operational	<ul style="list-style-type: none"> Mailbox communication No PDO communication
Safe-Operational	<ul style="list-style-type: none"> Mailbox communication Only Inputs PDO communication
Operational	<ul style="list-style-type: none"> Mailbox communication PDO communication

This state machine is standard to all EtherCAT slaves.

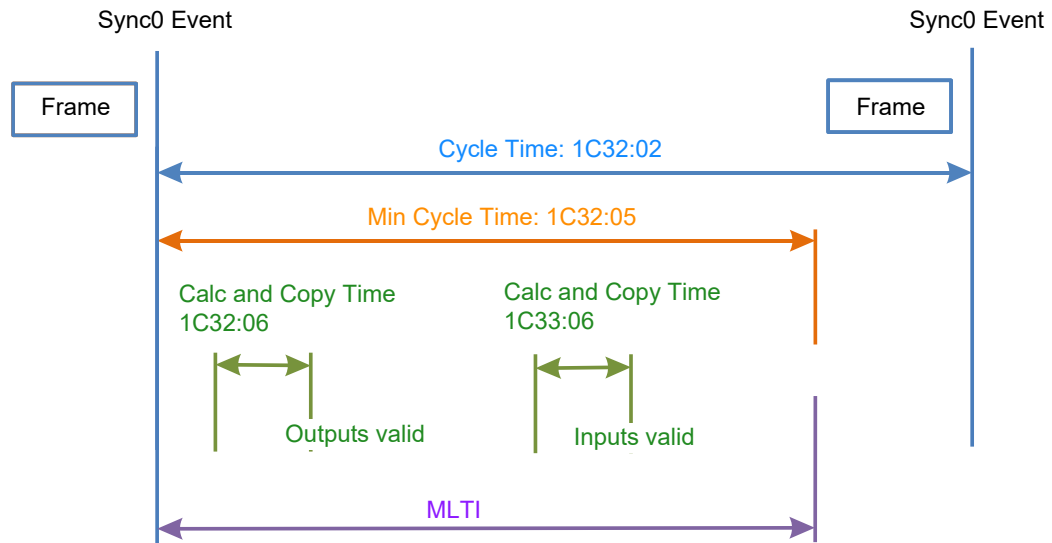
3.8 Synchronization with distributed clock

EtherCAT network can run in two modes: Distributed Clock or FreeRun.

- Distributed Clock (DC)** is the way to synchronize the EtherCAT communication. This mechanism synchronizes slaves through *Sync0 Event* which represents the internal application reference clock. EtherCAT specifies a time-shift inferior to 100ns between *Sync0 Event* of each slave on the segment. The controller is able to work at cycle time 500µs, 1ms, 2ms or 4ms. However, an AccurET controller works with a MLTI (Manager Loop Time Interrupt) of 500µs in any case. This MLTI is synchronized to *Sync0 Event* from EtherCAT frame. Whatever the frame runs at 500µs, 1ms, 2ms or 4ms, it will adjust at least one of the 500µs MTLIs to a *Sync0 Event* from EtherCAT. It is advised to set the cycle time to 500µs for best time stability.
- EtherCAT *FreeRun* mode can also be used, though in that case, there is no way to synchronize devices to each other. The MLTI still operates at about 500µs but no synchronization with frames is done. In FreeRun mode, the Master may send two frames at 500µs but AccurET controller may not execute them on time in CSP mode creating time-shift and ultimately undefined behavior. FreeRun mode can only be used to read / write data to AccurET controller using SDO datagrams. **It cannot be used with CSP mode.**

3.8.1 Setup DC mode

To initiate this mode, the developer must specify ESC register: 0x980 = 0x0300.



ESC processes EtherCAT frames on the fly. When a new frame has been processed, it raises a Sync0 signal on a dedicated line to the AccurET controller. This signal will be used for synchronization tasks. For the AccurET the Sync0 signal is raised every 500µs and it cannot be changed to any other value.

To generate this signal accurately and synchronize all slaves of a network (for instance when synchronized movement is needed), EtherCAT uses the concept of DC. The DC automatically synchronizes and takes into account delays over an EtherCAT network so that every ESC raises the Sync0 signal to its slave with a maximum jitter of 100ns.

AccurET controller triggers a new MLTI (Manager Loop Time Interrupt) to each Sync0 signal issued from the ESC. During each MLTI, a new CSP or PP target position can be entered, a register can be read/written through a SDO datagram (refer to figure in [§3.8](#) for more information).

3.8.2 Setup FreeRun

FreeRun mode is also supported but **must not be used** with CSP mode.

To initiate it, simply specify ESC register: 0x980 = 0x0000.

3.8.3 Related objects

These objects are 'controller specific'.

Index	Sub-Index	Name	Data Type	Access	Value
1C32h	00h	SM Output Parameter			32
	01h	Synchronization Type	UINT16	RO	1: FreeRun 2: DC Mode
	02h	Cycle Time	UINT32	RO	500000
	03h	Shift Time	UINT32	RW	0
	04h	Synchronization Types Supported	UINT16	RO	5
	05h	Minimum Cycle Time	UINT32	RO	500000
	06h	Calc and Copy Time	UINT32	RO	0
	08h	Get Cycle Time	UINT16	RW	0
	09h	Delay Time	UINT32	RO	0
	0Ah	Synch0 Cycle Time	UINT32	RO	500000
	0Bh	SM-Event Missed	UINT16	RO	0
	0Ch	Cycle Time Too Small	UINT16	RO	0
	20h	Sync Error	UINT8	RO	0
1C33h	00h	SM Input Parameter			32
	01h	Synchronization Type	UINT16	RO	1: FreeRun 2: DC Mode
	02h	Cycle Time	UINT32	RO	500000
	03h	Shift Time	UINT32	RW	0
	04h	Synchronization Types Supported	UINT16	RO	5
	05h	Minimum Cycle Time	UINT32	RO	500000
	06h	Calc and Copy Time	UINT32	RO	0
	08h	Get Cycle Time	UINT16	RW	0
	09h	Delay Time	UINT32	RO	0
	0Ah	Synch0 Cycle Time	UINT32	RO	500000
	0Bh	SM-Event Missed	UINT16	RO	0
	0Ch	Cycle Time Too Small	UINT16	RO	0
	20h	Sync Error	UINT8	RO	0

3.8.3.1 Object 1C32h: Sync manager channel 2

This object describes various information about mode of synchronization and timing for output objects.

3.8.3.1.1 Sub-object 01h: Synchronization type

This object indicates which mode of synchronization is enabled between the EtherCAT master and the AccurET.

0: FreeRun
2: DC mode

3.8.3.1.2 Sub-object 02h: Cycle time

This object indicates the current DC synchronization time set in nanosecond.

3.8.3.1.3 Sub-object 03h: Shift Time

This object sets a delay between SYNC0 event and data management in firmware. It might be used when a master has jitter in the frame sending.

This is only used with object 1C32h. It does not have any meaning in 1C33h object, but it has been kept for EtherCAT specification compliance.

3.8.3.1.4 Sub-object 04h: Synchronization types supported

This object indicates the list of all synchronization modes supported by the AccurET. Currently, this value is set to 5 and might only evolve with new version of firmware.

Bit 0 set to 1: *FreeRun* is supported.
Bit 2 set to 1: *DC mode* is supported.

3.8.3.1.5 Sub-object 05h: Minimum cycle time

This object indicates the minimum cycle time supported by AccurET in nanosecond. Currently this time is set to 500'000 ns. This is only relevant in DC mode.

3.8.3.1.6 Sub-object 06h: Calc and copy time

This object indicates the time required by the AccurET to copy the process data from the Sync Manager to the local memory and perform calculations if necessary before the data is sent to the process. This is only relevant in DC mode.

3.8.3.1.7 Sub-object 08h: Get cycle time

When this object is set to 1, the firmware makes time measurement and automatically update the object *Calc and Copy Time* (object 1C32:06h).

3.8.3.1.8 Sub-object 09h: Delay time

Not relevant in AccurET. It is set to 0.

3.8.3.1.9 Sub-object 0Ah: Synch0 cycle time

Not relevant in AccurET.

3.8.3.1.10 Sub-object 0Bh: SM-Event missed

This object indicates the error counter of missing frames by the AccurET. This is only relevant in DC mode.

3.8.3.1.11 Sub-object 0Ch: Cycle time too small

This object indicates the error counter when the cycle time is too small.

3.8.3.1.12 Sub-object 20h: Sync error

This object indicates there are some synchronization errors in the system.

3.8.3.2 Object 1C33h: Sync manager channel 3

This object describes various information about mode of synchronization and timing for input objects. All sub-objects have the same meaning as object 1C32h.

3.9 EtherCAT master

Multiple EtherCAT masters are available. They can be software based like TwinCAT, firmware based like PLCs, or hardware based like FPGAs. Whatever the format chosen, any customer loop must be fast enough to run within a constrained cycle time: 500µs (in CSP mode). This ensures a new target position will be taken into account by the AccurET every Sync0 event.

3.10 AccurET setting

3.10.1 General description

The EtherCAT master MUST set the following objects at startup so the drive becomes usable:

- Set up required PDO's with object 0x1C12 and object 0x1C13 (refer to [§3.6.2](#))
- Set up the *Configured Module Ident List* (object F030h)
- Set up the *Mode of Operation* of each module (objects 6060h and 6860h) (refer to [§4.3](#))

3.10.2 Related objects

Index	Sub-Index	Name	Data Type	Access	Value
F030h	00h	Configured Module Ident List	UINT8	RO	2
	01h	Ident Module 0	UINT32	RW	1 or 2
	02h	Ident Module 1	UINT32	RW	1 or 2

With:

- 1: Cyclic Synchronous Position mode at startup
- 2: Profile Position mode at startup

4 Application

4.1 Controller configuration

4.1.1 General description

The controller provides several objects which inform the application about AccurET state.

4.1.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Value
4000h	0	Motor Information			
	1	Type	UINT16	RO	0: linear 1: rotative
4001h	0	Encoder Information			
	1	Period	UINT32	RO	
	2	Interpolation	UINT32	RO	
4002h	0	Controller Information			
	1	Controller Status 1	UINT32	RO	Reg. M60
	2	Controller Status 2	UINT32	RO	Reg. M61
	3	Controller Status	UINT32	RO	Reg. M63
	4	MLTI [10*ns]	UINT32	RO	
	5	Movement Limit Min (low-part)	INT32	RO	Reg KL34
	6	Movement Limit Min (high-part)	INT32	RO	Reg KL34
	7	Movement Limit Max (low-part)	INT32	RO	Reg KL35
	8	Movement Limit Max (high-part)	INT32	RO	Reg KL35
	9	Timeout delayed power off (ms)	UINT32	RW	

4.1.2.1 Object 4000h: Motor information

This object indicates information about the motor.

Sub-object 1 "Type" can have two values: 0 or 1. The values represent respectively a linear or a rotary motor.

4.1.2.2 Object 4001h: Encoder information

This object indicates information about the encoder.

Sub-objects 1 and 2 represents respectively the Period and the Interpolation of the encoder.

4.1.2.3 Object 4002h: Controller information

This object indicates information about the state of the controller.

Sub-objects 1 to 3 provide the status of the controller. These objects represent respectively the registers M60, M61 and M63 of the AccurET.

Sub-object 4 "MLTI [10*ns]" represents the MLTI time of AccurET.

Sub-objects 5 to 8 represents the software position limits set in the AccurET (registers MINSL, MAXSL or equivalently KL34 and KL35).

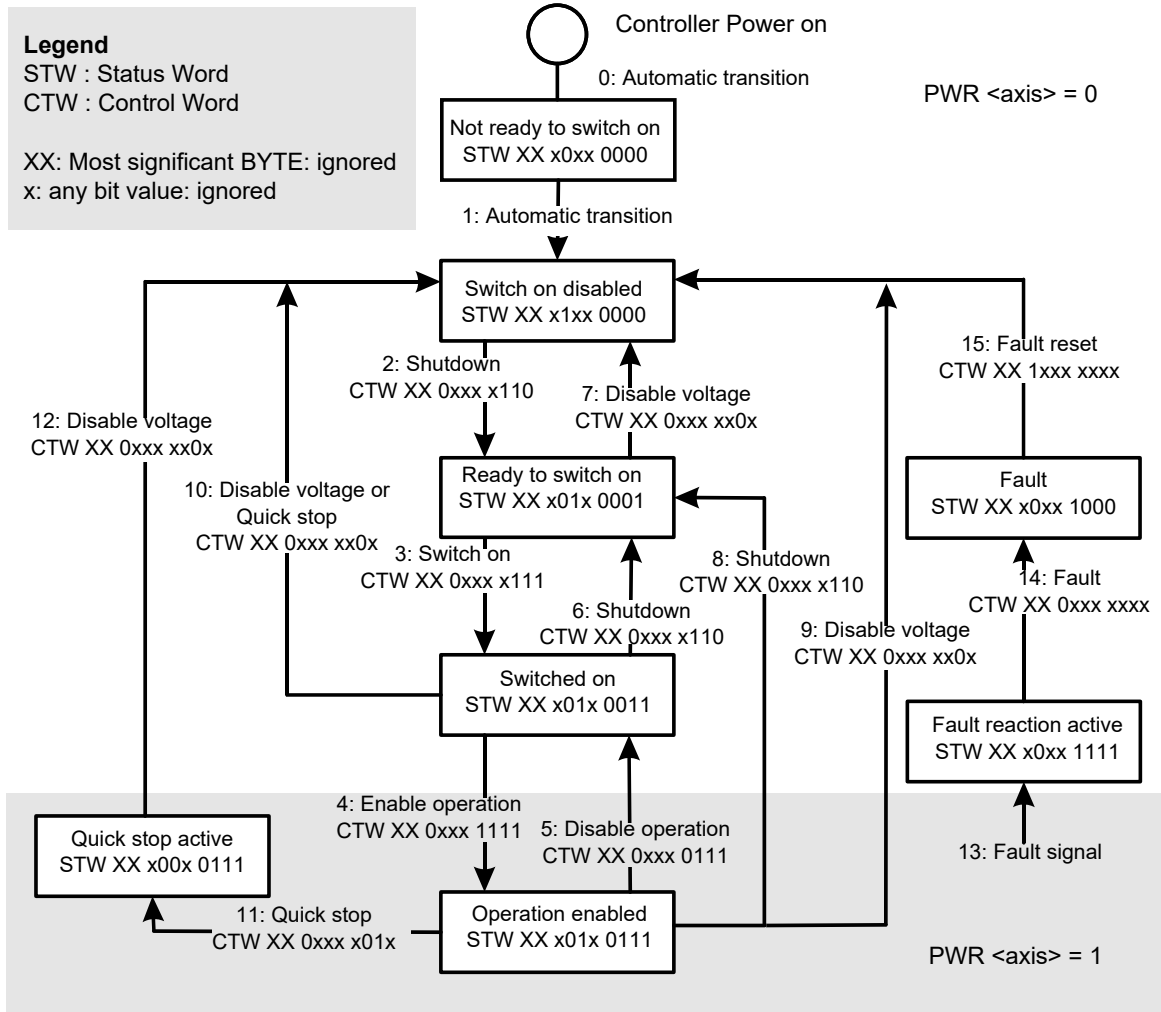
Sub-object 9 is the timeout of power off of the motor. This aims to keep the motor powered on while changing mode of operation (refer to [§4.2](#) for more information).

4.2 Device control

4.2.1 General description

The CiA402 state machine is controlled through the *Control Word* (object 6040h) and the status is obtained through *Status Word* (object 6041h).

The following figure describes the CiA402 state machine, when the EtherCAT state machine is in "Operational" state (refer to §3.7):



Transition	Command	Actions
0	Automatic transition after power-on or reset application	None
1	Automatic transition	None
2	Shutdown command	None
3	Switch on command	None
4	Enable operation	Power supplied to the motor
5	Disable operation	Slow down with slow down ramp and then power off the motor
6	Shutdown command	None
7	Disable voltage	None
8	Shutdown	Slow down with slow down ramp and then power off the motor
9	Disable voltage	None
10	Disable voltage or Quick Stop	None

Transition	Command	Actions
11	Quick stop	Perform action set in <i>Quick Stop Option Code (object 605Ah)</i> .
12	Disable voltage	Power off the motor
13	Fault signal	The configured fault reaction is executed, then the automatic transition 14 is performed
14	Fault	Ensure motor is not moving, no command is still running, Control Word bit 7 is 0 and power off
15	Fault reset	A reset of the fault condition is carried out.

If in *Quick Stop Active* state, the quick stop option code is set to 1, 2, 3, the controller device automatically leaves this state to go to *Switch on disabled* state when the motor is stopped.

4.2.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Value
6040h	0	Control Word	UINT16	RW	
6041h	0	Status Word	UINT16	RO	
605Ah	0	Quick Stop Option Code	INT16	RW	
605Dh	0	Halt Option Code	INT16	RW	

4.2.2.1 Object 6040h: Control word

This object indicates the command controlling the AccurET. It is structured as defined in the following figure. Bits 7, 3, 2, 1, and 0 are supported. Other bits may be supported depending on the selected mode of operation.

The AccurET supports the following modes of operation:


- *Profile Position* mode
- *Cyclic Synchronous Position* mode
- *Homing* mode

Refer to corresponding section for more information about *Control Word* for each mode.


Control Word is defined as follows:

15	12	11	10	9	8	7	6	4	3	2	1	0
Not used			Dp	r	oms	h	fr	oms	eo	qs	ev	so
MSB												LSB

dp	delayed motor power off
r	reserved
oms	operation mode specific
h	halt
f	fault reset
eo	enable operation
qs	quick stop
ev	enable voltage
so	switch on

Command	Bits of the Control Word					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
Enable operation	0	1	1	1	1	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
Fault reset		X	X	X	X	15

X represents an ignored bit value.

 : active on rising edge

Bits 9, 6, 5, and 4 of the *Control Word* are operation mode specific.

The halt function (bit 8) behavior is operation mode specific. If the bit is set to 1, the commanded motion is interrupted and the AccurET behaves as defined in the *Halt Option Code (object 605Dh)*. After releasing the halt function, the commanded motion continues if possible.

If bit 11 is 1, motor power on is still active after transition 9. In that case, the application must go in state "Operation Enabled" before the timeout (object 4002:09) expires. If it does not, the AccurET goes into error state. The aim of this functionality is to have a way to change the *Mode of Operation (object 6060h)* with motor powered on.

Bits 10, 12 - 15 are not used.

4.2.2.2 Object 6041h: Status word

This object provides AccurET status. The object is structured as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not used	oms	ila	tr	Not used	ms	w	sod	qs	ve	f	oe	so	rtso		
MSB													LSB		

oms	operation mode specific
ila	internal limit active
tr	target reached
w	warning
sod	switch on disabled
qs	quick stop
ve	voltage enabled
f	fault
oe	operation enabled
so	switched on
rtso	ready to switch on

Status Word	AccurET state
xxxx xxxx x0xx 0000 _b	Not ready to switch on
xxxx xxxx x1xx 0000 _b	Switch on disabled
xxxx xxxx x01x 0001 _b	Ready to switch on

Status Word	AccurET state
xxxx xxxx x01x 0011 _b	Switched on
xxxx xxxx x01x 0111 _b	Operation enabled
xxxx xxxx x00x 0111 _b	Quick stop active
xxxx xxxx x0xx 1111 _b	Fault reaction active
xxxx xxxx x0xx 1000 _b	Fault

x represents an ignored bit value.

If bit 4 (voltage enabled) of *Status Word* is 1, this indicates that the power voltage (Vpower) is inside the required range.

If bit 5 (quick stop) of *Status Word* is 0, this indicates that the AccurET is reacting on a quick stop request.

If bit 7 (warning) of *Status Word* is 1, this indicates the presence of a warning condition. Warning is not an error. The AccurET status does not change. Warning cause might be given in the *Error Code (object 603Fh)* and is equivalent to bit 23 of M63 register.

If bit 10 (target reached) of *Status Word* is 1, this indicates that the AccurET has reached the set-point. The set-point is operation mode specific and is defined in detail in the corresponding section of this document.

Bit 10 is also set to 1, if operation mode has been changed. Software target value change modifies this bit.

If *Quick Stop Option Code (object 605Ah)* is 5, 6, 7, bit 10 is set to 1 when quick stop operation is finished and the AccurET is halted. If halt occurred and the AccurET actually halted then bit 10 is set to 1.

If bit 11 (internal limit active) of *Status Word* is 1, this indicates that an internal limit is active (example: position range limit).

Bit 13 and bit 12 of *Status Word* are operation mode specific.

Bits 19, 4 and 15 are not used.

4.2.2.3 Object 605Ah: Quick stop option code

This object indicates what action is performed when the quick stop function is executed. The slow down ramp is the deceleration value of the current used mode of operation.

The following table specifies the value definition.

Value	Definition
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	Slow down on current limit and transit into "Switch On Disabled"
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	Slow down on current limit and stay in "Quick Stop Active"

4.2.2.4 Object 605Dh: Halt option code

This object indicates what action is performed when the halt function is executed. The slow down ramp is the deceleration value of the used mode of operation.

The following table specifies the value definition.

Value	Definition
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	Slow down on current limit and stay in "Operation Enabled"

4.3 Modes of operation

4.3.1 General description

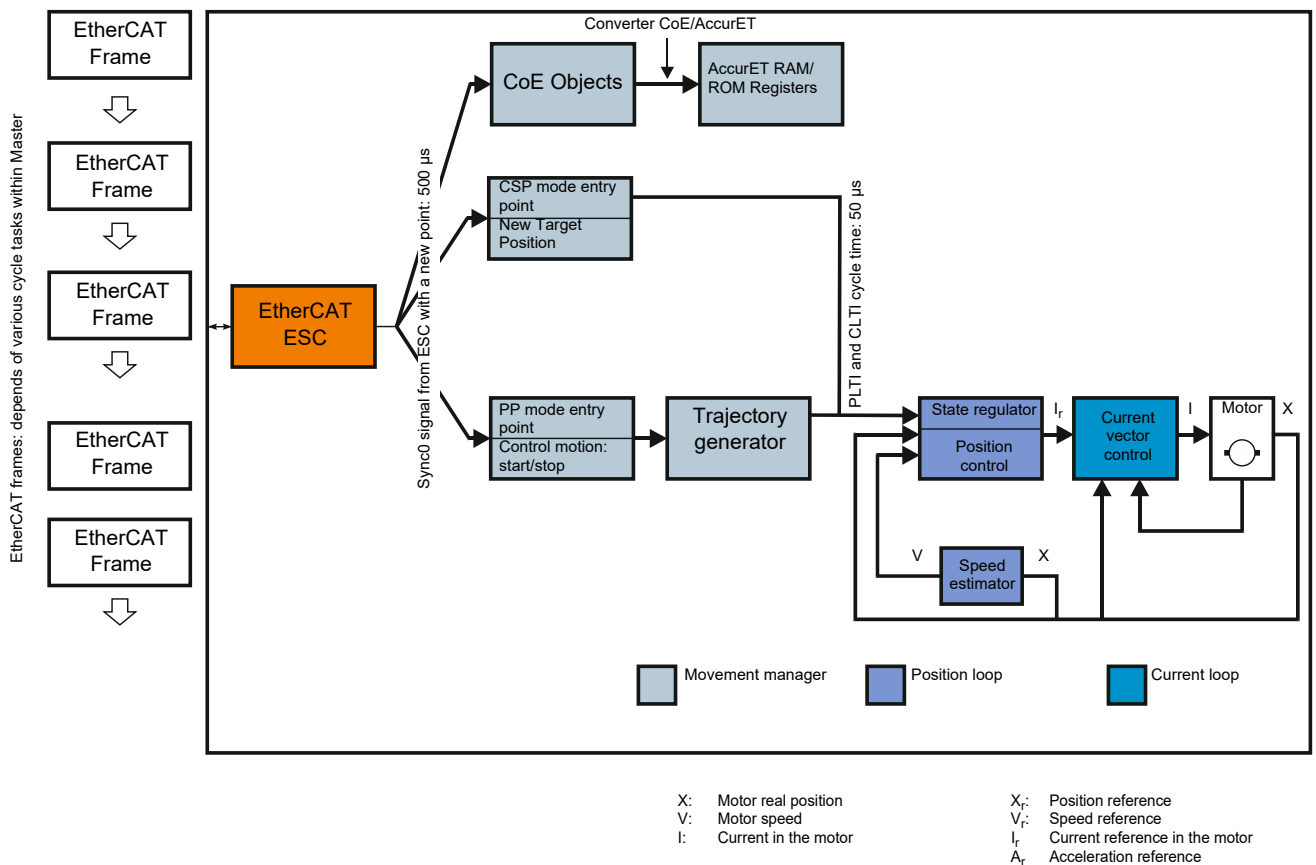
The AccurET supports the following modes of operation:

- *Profile Position* mode (PP)
- *Cyclic Synchronous Position* mode (CSP)
- *Homing* mode (HM)

In the PP mode, the user specifies once the target position, the speed and the acceleration. The trajectory generator of the AccurET manages then the whole motion.

With the CSP mode, the EtherCAT master manages the full motion and dispatches position references every EtherCAT cycle to the AccurET. This is similar to the UltimET Interpolation mode.

The following figure explains the command flow for both modes:



Refer to the [§4.4](#), [§4.5](#) and [§4.6](#) for more details about these modes.

4.3.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Value
6060h	0	Mode of Operation	INT8	RW	
6061h	0	Mode of Operation Display	INT8	RO	
6502h	0	Supported Drive Modes	UINT32	RO	

4.3.2.1 Object 6060h: Mode of operation

This object indicates the requested operation mode. The table below indicates authorized values.

Value	Definition
1	Profile Position mode
6	Homing mode
8	Cyclic Synchronous Position mode

4.3.2.2 Object 6061h: Mode of operation display

This object provides the current operation mode. The table above indicates possible values.

4.3.2.3 Object 6502h: Supported drive modes

This object provides information about supported controller modes.

31	8	7	5	4	1	0
0	csp	0	hm	0	pp	
MSB						LSB

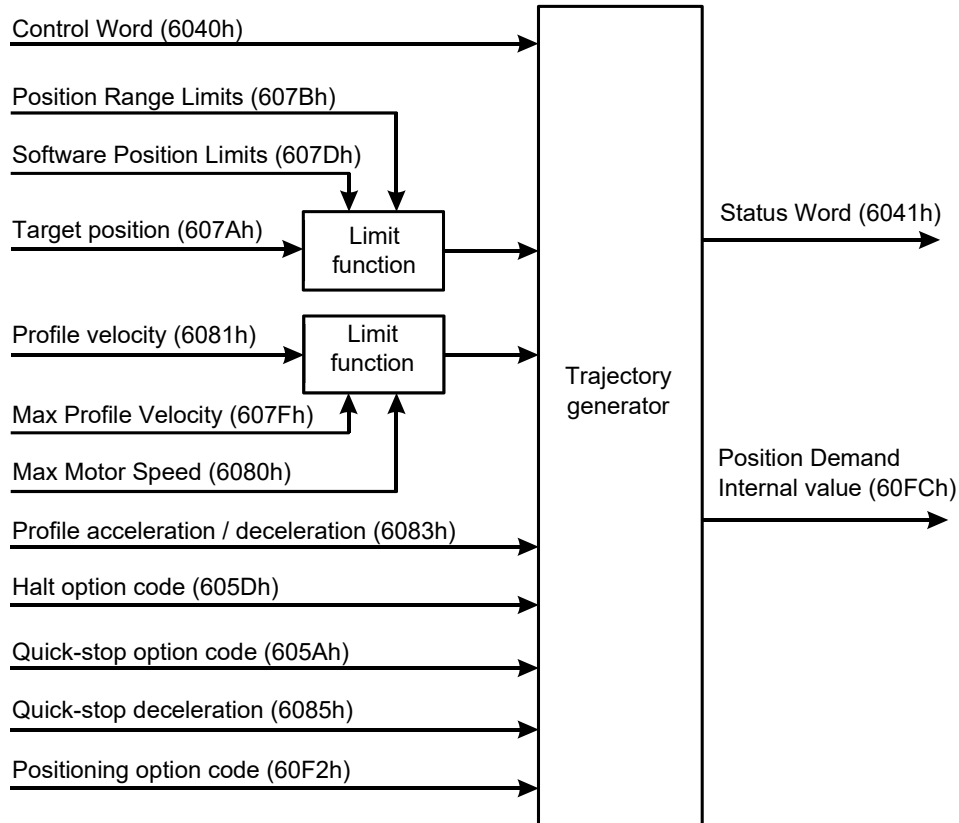
This object always returns 0xA1. Each set bit indicates the supported operation modes. This value might only change with the new releases of AccurET firmware.

4.4 Profile Position mode

4.4.1 General description

Profile Position mode is used to start positioning to *Target Position (object 607Ah)* with *Profile Velocity (object 6081h)* and *Profile Acceleration (object 6083h)*. In that case, the trajectory generator is located in the controller.

The following figure shows the *Profile Position* mode block diagram.



Trajectory generator set-points are managed with *Control Word (object 6040h)* using the four operation modes specific bits for *Profile Position* mode:

Control Word:

- Bit 4: new set-point
- Bit 5: change set immediately
- Bit 9: change on set-point

Status Word:

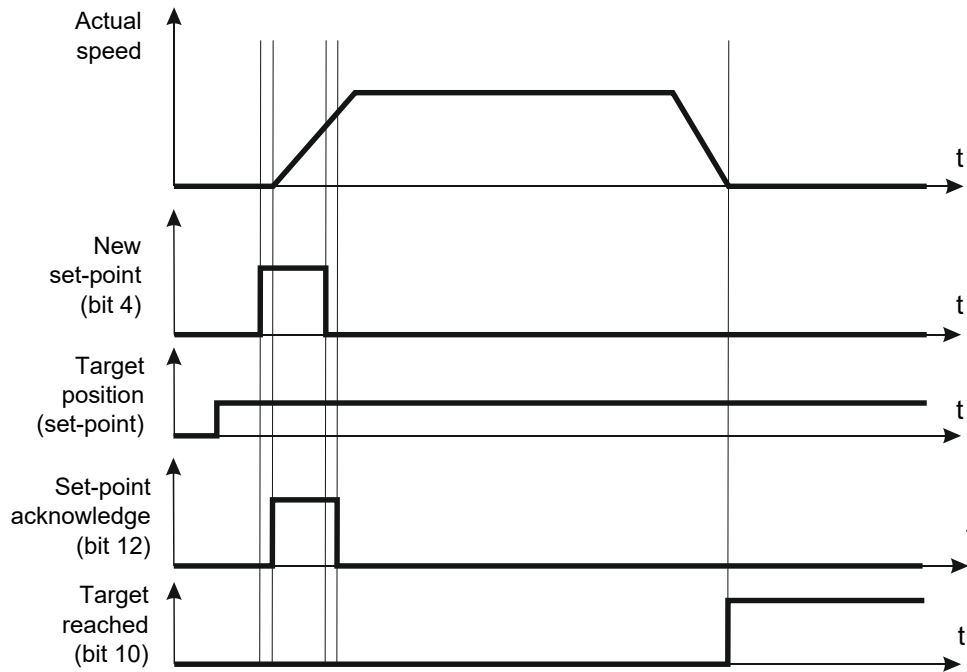
- Bit 12: set-point acknowledge

If the *change set immediately* bit 5 of *Control Word (object 6040h)* is set to 1, the new movement starts immediately. The controller calculates the new position trajectory and the motor switches from one movement to the new one without going through 0 speed.

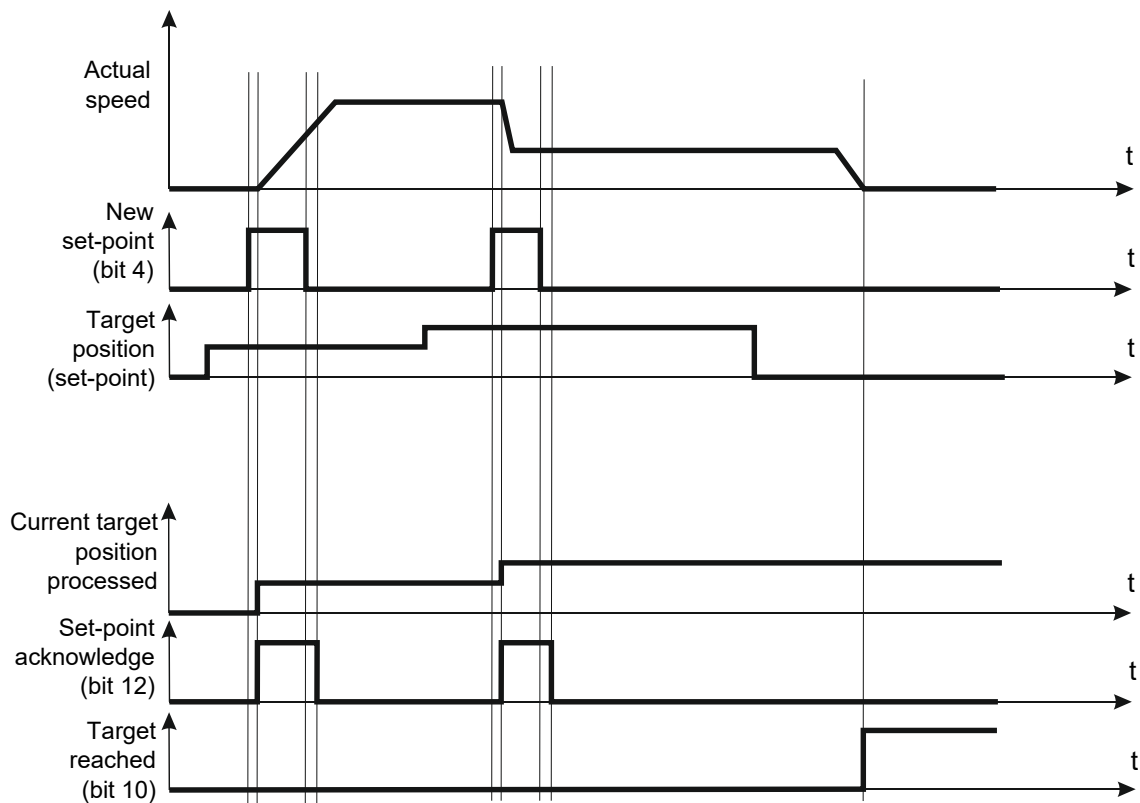
If the *change set immediately* bit 5 of *Control Word (object 6040h)* is set to 0, the new movement is going to start as soon as the current one has reached its target. In that case, two different behaviors can occur depending on the state of *change on set-point* bit 9 in the *Control Word (object 6040h)*:

- If bit is set to 0, the current movement is ending with profile deceleration, the speed goes to 0, and then, the new movement starts.
- If bit is set to 1, the current movement continues at its current speed until it reaches the target position without deceleration, and then, the new movement starts without going through 0 speed.

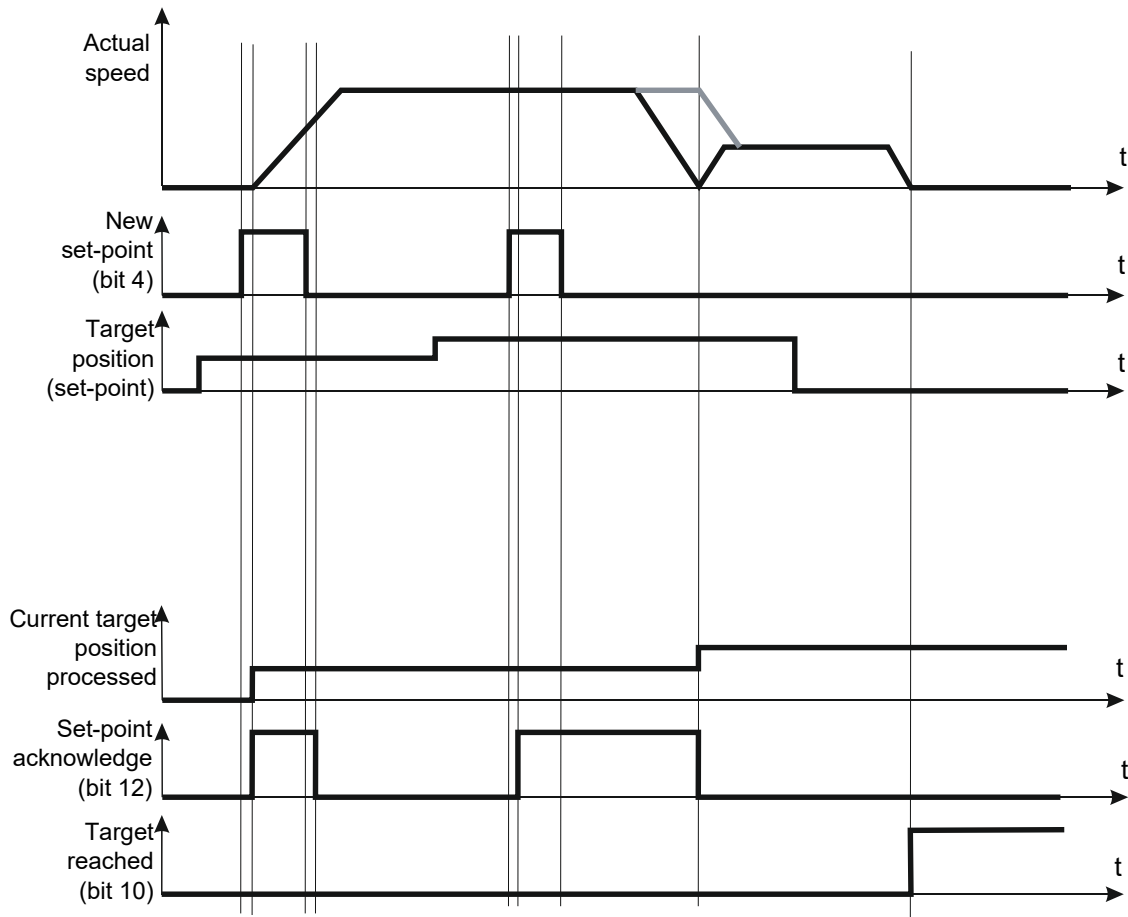
When the master wants to apply a new set-point to the controller, the master signals that the set-point is valid by a rising edge of the *new set-point* bit 4 in the *Control Word (object 6040h)*. The slave sets the *set-point acknowledge* bit 12 in the *Status Word (object 6041h)* to 1 in return. When *new set-point* bit 4 in the *Control Word (object 6040h)* falls down to 0, AccurET is then available to receive a new movement request and *set-point acknowledge* bit 12 is set to 0. An example is shown in the following time diagram.



When a set-point is in progress and a new set-point is validated by the *new set-point* (bit 4) in the *Control Word* (object 6040h), bits 5 and 9 of the *Control Word* (object 6040h) are evaluated to determine the behavior of the movement. If bit 5 is set to 1, the new set-point is processed immediately. This handshaking procedure is shown in the following time diagram.



When a set-point is in progress and a new set-point is validated by the *new set-point* (bit 4) in the *Control Word* (object 6040h), bits 5 and 9 of the *Control Word* (object 6040h) are evaluated to determine the behavior of the movement. If bit 5 is set to 0, the new set-point is processed only after the previous one has reached its targeted position. This handshaking procedure is shown in the following time diagram. The additional gray line segment in the graph 'actual speed' shows the actual speed if *change of set point* bit (bit 9) is set to 1.



In each case, a new set-point can only be taken into account if *set-point acknowledge* bit 12 in *Status Word* (object 6041h) is cleared.

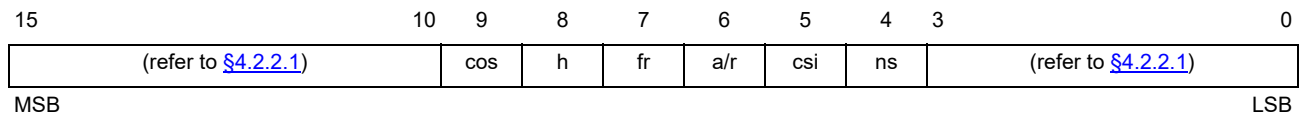
4.4.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).




Index	Sub-Index	Name	Data Type	Access	Unit
6040h	0	Control Word	UINT16	RW	
6041h	0	Status Word	UINT16	RO	
6064h	0	Position Actual Value	INT32	RO	Inc
607Ah	0	Target position	INT32	RW	Inc
607Fh	0	Max Profile Velocity	UINT32	RW	Inc/s
6080h	0	Max Motor Speed	UINT32	RW	Inc/s
6081h	0	Profile Velocity	UINT32	RW	Inc/s
6083h	0	Profile Acceleration / Deceleration	UINT32	RW	
6085h	0	Quick Stop Deceleration	UINT32	RW	
60F2h	0	Positioning Option Code	UINT16	RW	
60FCh	0	Position Demand Internal Value	INT32	RO	Inc

4.4.2.1 Object 6040h: Control word

In case of *Position Profile* mode, the *Control Word (object 6040h)* is modified to be structured as follows:



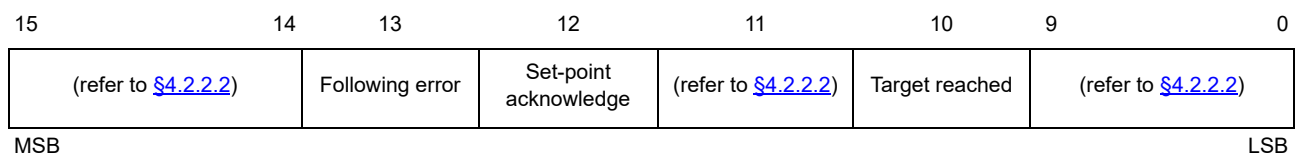
cos: change on set-point
h: halt
fr: fault reset
a/r: absolute / relative
csi: change set immediately
ns: new set-point

Bit 9	Bit 5	Bit 4	Definition
0	0		Positioning is completed (target reached) before the next one gets started
1	0		Positioning with the current profile velocity up to the current set-point is proceeded and then next positioning is applied
X	1		Next positioning starts immediately

Bit	Value	Definition
6	0	Target position is an absolute value
	1	Target position is a relative value (depending on object 60F2h)
8	0	Positioning is executed or continued
	1	Axis is stopped accordingly to halt option code (object 605Dh)

4.4.2.2 Object 6041h: Status word

In case of *Position Profile* mode, the *Status Word (object 6041h)* is modified to be structured as follows:



Bit	Value	Definition
10	0	Halt (bit 8 in Control Word) = 0: Target position not reached
		Halt (bit 8 in Control Word) = 1: Axis decelerates
	1	Halt (bit 8 in Control Word) = 0: Target position reached
		Halt (bit 8 in Control Word) = 1: Velocity of axis is 0
12	0	Previous set-point already processed, waiting for new set-point
	1	Previous set-point still in process, set-point overwriting is accepted
13	0	No following error
	1	Following error

4.4.2.3 Object 6064h: Position actual value

This object indicates the current position of the axis.

4.4.2.4 Object 607Ah: Target position

This object indicates the requested position that the controller must move to in *Position Profile* mode using the current settings of motion control parameters such as velocity, acceleration / deceleration, motion profile type, etc.

The value of this object is interpreted as absolute or relative depending on the 'absolute / relative' flag, bit 6 in the *Control Word (object 6040h)*. Its unit is assumed to be in position increments.

4.4.2.5 Object 607Fh: Max profile velocity

This object indicates the configured maximal allowed velocity in either direction during profiled motion. Its unit is assumed to be in increments per second.

4.4.2.6 Object 6080h: Max Motor Speed

This object indicates the configured maximal allowed speed for the motor. Its unit is assumed to be in increments per seconds.

The same behavior as register K31.

For more information, refer to the '**AccurET Operation & Software Manual**' (section Movements limits).

4.4.2.7 Object 6081h: Profile velocity

This object indicates the configured velocity normally reached at the end of the acceleration ramp during a profiled motion and is valid for both directions of motion.

The value is given in increments per second.

4.4.2.8 Object 6083h: Profile acceleration / deceleration

This object indicates the configured acceleration and deceleration. There is no way to get deceleration different of the acceleration. This deceleration is considered as the *slow ramp*.

The value is given in position increments per second squared.

4.4.2.9 Object 6085h: Quick stop deceleration

This object indicates the configured deceleration used to stop the motor when the quick stop function is activated and the *Quick Stop Code (object 605Ah)* is set to 2 or 6. The quick stop deceleration is also used if the *Halt Option Code (object 605Dh)* is set to 2.

The value is given in position increments per second squared.

4.4.2.10 Object 60F2h: Positioning option code

This object indicates the configured positioning behavior when the *relative option* bits of *Control Word (object 6040h)* is set to 1 in *Profile Position* mode. The object is structured as follows:

15	2	1	0
Not used			Relative option
MSB			LSB

Bit 1	Bit 0	Definition
0	0	Positioning moves is performed relative to the preceding (internal absolute) target position (respectively Relative to 0 if there is no preceding target position)
0	1	Positioning moves is performed relative to the actual position demand value (object 60FCh) – output of the trajectory generator

Bit 1	Bit 0	Definition
1	0	Positioning moves is performed relative to the position actual value (object 6064h)
1	1	Reserved

4.4.2.11 Object 60FCh: Position Demand Internal Value

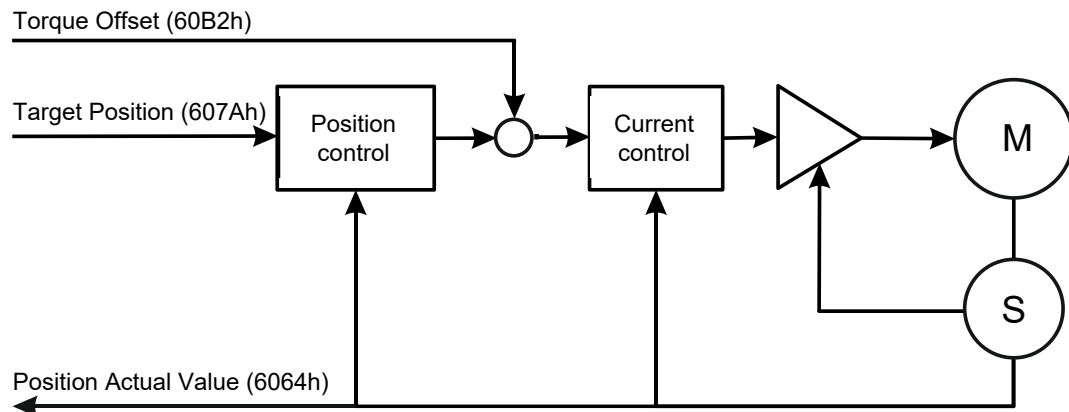
This object provides the output of the trajectory generator in profile position mode.

The value is given in increments.

4.5 Cyclic Synchronous Position mode

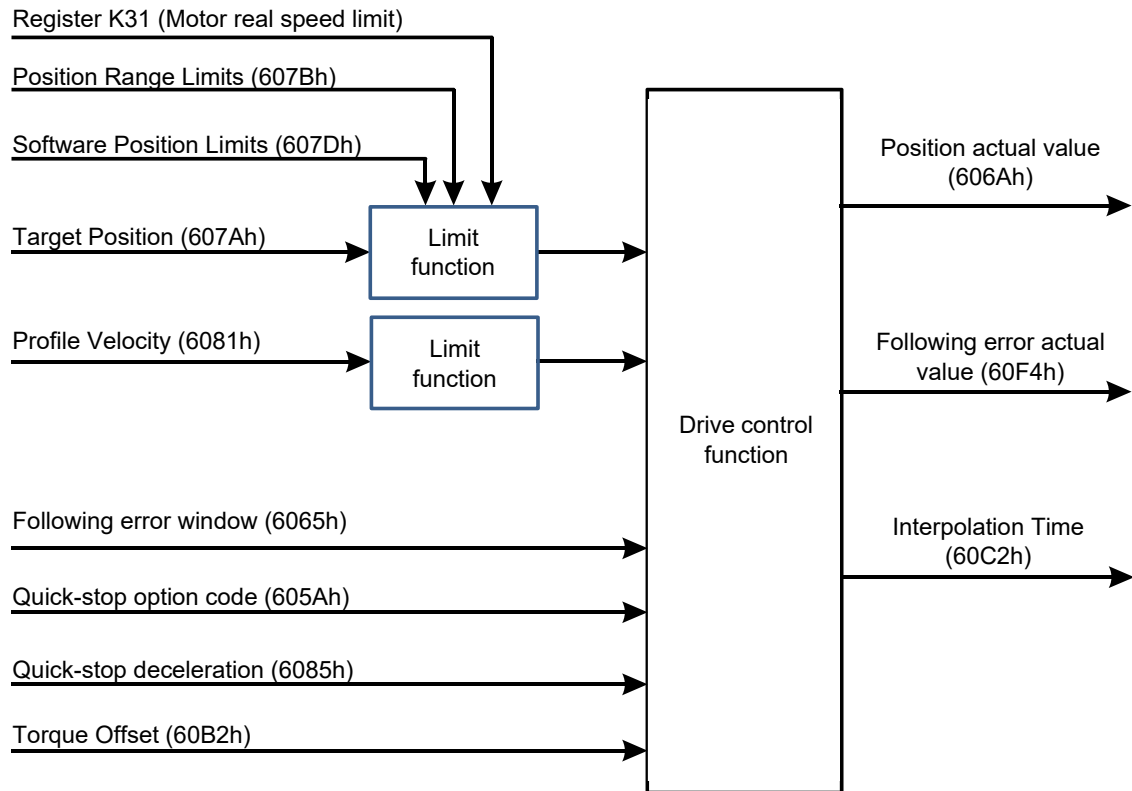
4.5.1 General description

With this mode, the trajectory generator is located in the master, not in the AccurET. The *Cyclic Synchronous Position* mode is used for the interpolated positioning.



The figure above shows the inputs and outputs of the control function for the controller. The input values (from the control function point of view) is the *Target Position (object 607Ah)*. The unit for the target position is in increments. The *Position Actual Value (object 6064h)* is the device control output. It is also in increments.

The following figure shows the *Cyclic Synchronous Position* mode block diagram.



Unlike to the PP mode, the controller monitors here the following error, also called tracking error.

4.5.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Unit
6040h	0	Control Word	UINT16	RW	
6041h	0	Status Word	UINT16	RO	
6064h	0	Position Actual value	INT32	RO	Inc
6065h	0	Following Error Window	UINT32	RW	
607Ah	0	Target Position	INT32	RW	Inc
6085h	0	Quick Stop Deceleration	UINT32	RW	
60B2h	0	Torque Offset	INT16	RW	
60C2h	0	Interpolation Time	-	-	-
	1	Interpolation Period	UINT8	RO	
	2	Interpolation Index	INT8	RO	
60F4h	0	Following Error Actual Value	INT32	RO	
2802h	0	Latch Control Word	UINT16	RW	
2901h	0	Latch Status Word	UINT16	RO	
2902h	0	Latch Position	INT32	RO	

4.5.2.1 Object 6040h: Control word

No mode-specific bits of the *Control Word*. For standard bits, refer to [§4.2.2.1](#).

4.5.2.2 Object 6041h: Status word

In case of *Cyclic Synchronous Position* mode, the *Status Word* is modified and structured as follows:

15	14	13	12	11	10	9	0
(refer to §4.2.2.2)	Following error	Controller follows the command value	(refer to §4.2.2.2)	Reserved	(refer to §4.2.2.2)		
MSB							LSB

Bit	Value	Definition
12	0	Controller does not follow the command value – Target position ignored
	1	Controller follows the command value – Target position is used as input to position control loop
13	0	No following error
	1	Following error

4.5.2.3 Object 6065h: Following error window

This object indicates tolerated position error relative to *Target Position* (object 607Ah).

Allowed Position = Target Position +/- Following Error Window

If the *Position Actual Value* (object 6064h) is out of the following error window, a following error occurs and the controller is blocked.

This value is assumed to be in position increments.

4.5.2.4 Object 6064h: Position actual value

This object indicates the current position of the axis.

4.5.2.5 Object 607Ah: Target position

This object indicates the requested position that the position controller of the AccurET must reach. The behavior is the same as the one when register K61 is set to 4.

This value is assumed to be in position increments.

4.5.2.6 Object 6085h: Quick stop deceleration

This object indicates the configured deceleration used to stop the motor when the quick stop function is activated and the *Quick Stop Code* (object 605Ah) is set to 2 or 6. The quick stop deceleration is also used if the *Halt Option Code* (object 605Dh) is 2.

The value is given in position increments per second squared.

4.5.2.7 Object 60B2h: Torque offset

This provides a feed-forward offset for the current control loop. The value is given per thousand of *Motor Rated Torque* (object 6076h).

4.5.2.8 Object 60C2h: Interpolation time

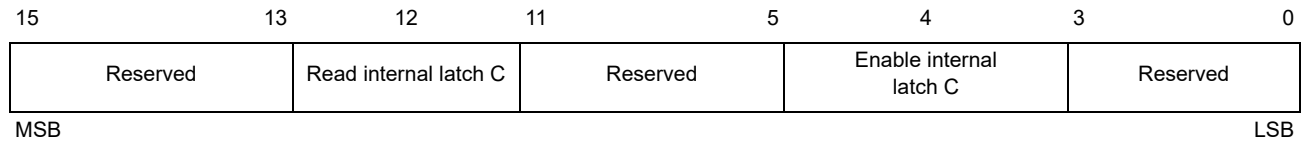
This object indicates the interpolation time between reception of two target position requests. This is a read-only object, the value is automatically computed in the controller.

4.5.2.9 Object 60F4h: Following error actual value

This indicates the actual following error value. This position is provided in increments.

4.5.2.10 Object 2802h: Latch control word

This object is structured as follows:

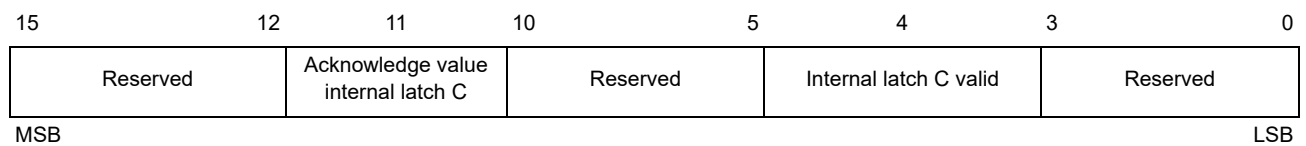


Bit 4 enable detection of C (index) signal of the encoder

Bit 5 ask the controller to set the Latch Position object

4.5.2.11 Object 2901h: Latch status word

This object is structured as follows:



Bit 4 indicates the signal C (index) of the encoder has been detected.

Bit 11 indicates the value in Latch Position object is valid.

4.5.2.12 Object 2902h: Latch position

This object indicates the position where the C (index) signal of the encoder has been detected.

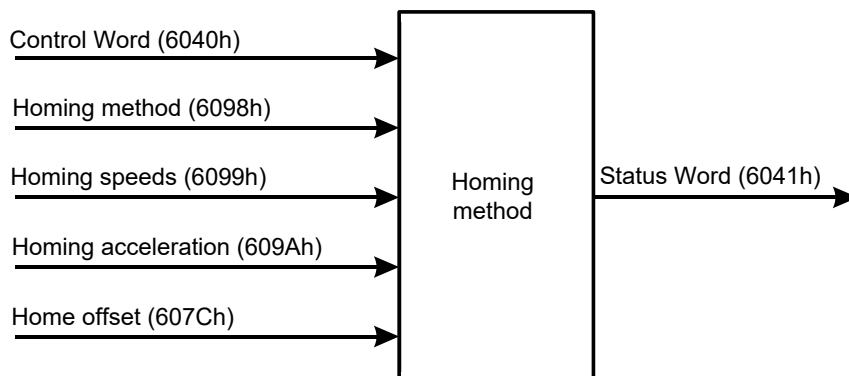
This position is provided in increments.

4.6 Homing**4.6.1 General description**

The following figure shows the defined input objects as well as the output objects. The user may specify the speed, acceleration and the method of homing. There is an object *home offset*, which allows the user to set or move the zero position.

There is no output data except for bits in the *Status Word* that return the status or result of the homing process.

There is only one homing speed available to perform the homing procedure.



In addition to the EtherCAT homing methods described in this chapter, it is possible to use all AccurET specific homing modes thanks to a PLC library (refer to [§4.6.2.7.1](#) for more information).

4.6.2 Related objects

These objects are 'module specific', so it exists an equivalent object at object + 800h to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Unit
6040h	0	Control Word	UINT16	RW	
6041h	0	Status Word	UINT16	RO	
607Ch	0	Home offset	INT32	RW	Inc
6098h	0	Homing Method	INT8	RW	
6099h	0	Homing Speeds	-	-	-
	1	Speed	UINT32	RW	
	2	Not used	UINT32	RW	
609Ah	0	Homing Acceleration	UINT32	RW	
60E3h	0	Supported Homing Methods			
	1	Supported Method 1	INT8	RO	
	2	Supported Method 2	INT8	RO	
	3	Supported Method 3	INT8	RO	
	4	Supported Method 4	INT8	RO	
	5	Supported Method 5	INT8	RO	
	6	Supported Method 6	INT8	RO	

4.6.2.1 Object 6040h: Control word

In case of *Homing* mode, the *Control Word* is modified to be structured as follows:

15	9	8	7	6	5	4	3	0
(refer to §4.2.2.1)	halt	(refer to §4.2.2.1)	Reserved (0)	Homing operation start	(refer to §4.2.2.1)			
MSB								LSB

Bit	Value	Definition
4	0	Do not start homing procedure
	1	Start or continue homing procedure
8	0	Enable bit 4
	1	Stop axis according to halt option code (object 605Dh)

4.6.2.2 Object 6041h: Status word

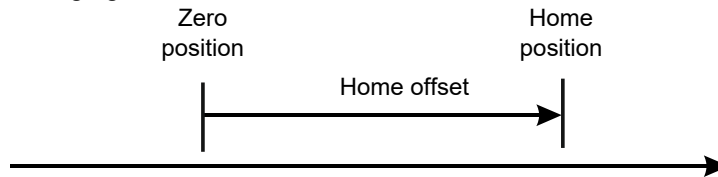
In case of *Homing* mode, the *Status Word* is modified to be structured as follows:

15	14	13	12	11	10	9	0
(refer to §4.2.2.2)	Homing error	Homing attained	(refer to §4.2.2.2)	Target reached	(refer to §4.2.2.2)		
MSB							LSB

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 position is found, but the motor is still moving
0	1	1	Homing procedure 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

4.6.2.3 Object 607Ch: Home offset

This object indicates the difference between the zero position for the application and the machine home position (found during homing). When the homing procedure is completed, the offset is added to the machine home position. And then, all subsequent absolute moves are taken relative to this new zero position. This is illustrated in the following figure.



The unit of the *home offset* is in increments.

4.6.2.4 Object 6098h: Homing method

This object indicates the configured homing method to be used.

4.6.2.5 Object 6099h: Homing speeds

This object indicates the configured speed used during homing procedure. The value is given in increments per second. This object update the AccurET register HSPD (KL41). The second sub-object is not used with AccurET. It has been added to be compliant with CiA402 specifications.

4.6.2.6 Object 609Ah: Homing acceleration

This object indicates the configured acceleration and deceleration to be used during homing operation. The value is given in increments per second squared.

4.6.2.7 Object 60E3h: Supported homing methods

Hereafter, the list of methods supported by the current version of controller.

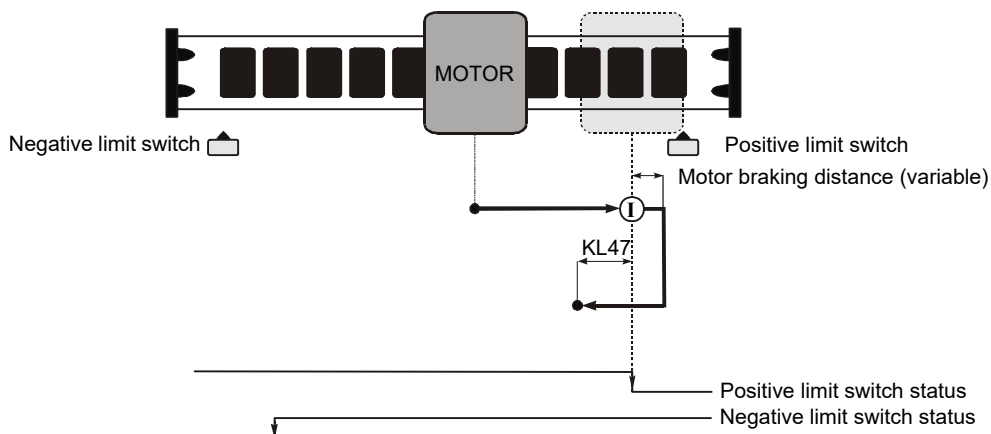
4.6.2.7.1 Method 15

Apply the AccurET specific homing method configured with ComET. In this mode, all other homing objects are ignored.

A PLC library is provided as an example, refer to [§6.2](#) for more information.

4.6.2.7.2 Method 18

Homing on a limit switch with a positive movement. After having found the limit switch, the motor moves back the distance given by parameter KL47.



If at the beginning of the homing the motor is on the positive limit switch, the motor moves back the distance given by parameter KL48.

This mode is equivalent to Homing mode AccurET K40.x = 4.

4.6.2.7.3 Method 17

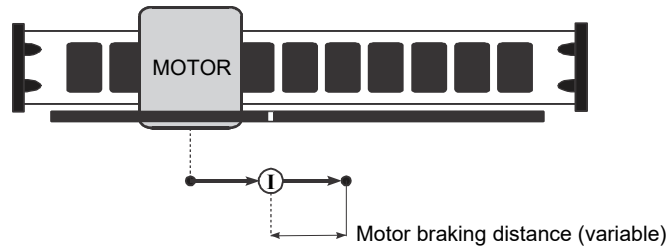
Same as Method 18 with a negative movement.

This mode is equivalent to Homing mode AccurET K40.x = 5.

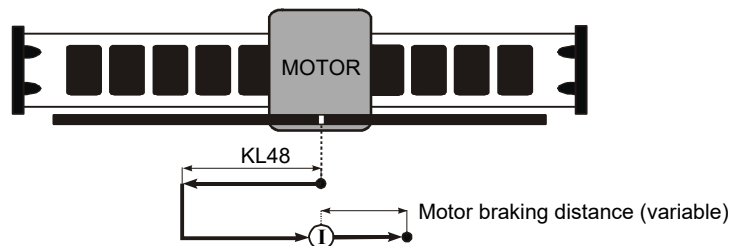
4.6.2.7.4 Method 34

Homing with a mono-reference mark with a positive movement. To have the mono-reference mark always in the same position, the motor must find it when moving in the determined direction. There are three possibilities:

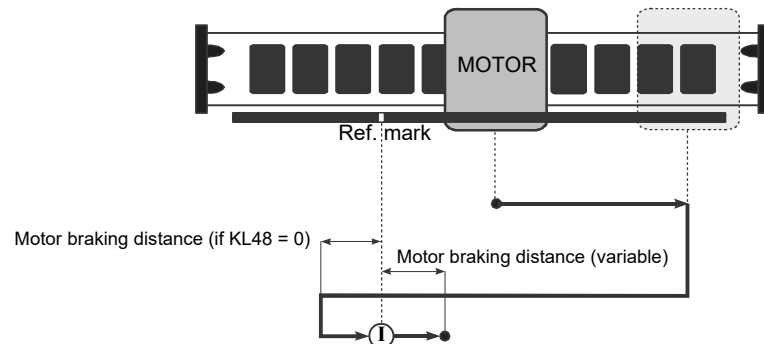
- The motor is on the left of the mono-reference mark at the beginning of the homing. It moves and directly meets the mono-reference mark.



- The motor is on the mono-reference mark at the beginning of the homing. It moves the distance given by parameter KL48 in the opposite direction of the homing to leave the mono-reference mark. Then it moves back towards the mono-reference mark in the right direction. The parameter KL48 must be bigger than the home switch length because if the motor is still on the top of it after the movement given by the parameter KL48, the **LEAVEREF ERROR** error (M64=60) appears.



- The motor is on the right of the mono-reference mark at the beginning of the homing. After having found the mechanical end stop, it comes back. When it finds the mono-reference mark (in the wrong direction), it keeps moving the distance given by the parameter KL48 before changing direction a second time to find the mono-reference mark in the right direction. The parameter KL48 must be bigger than the mono-reference mark length because if the motor is still on the top of it after the movement given by the parameter KL48, the **LEAVEREF ERROR** error (M64=60) appears.



This mode is equivalent to Homing mode AccurET K40.x = 8.

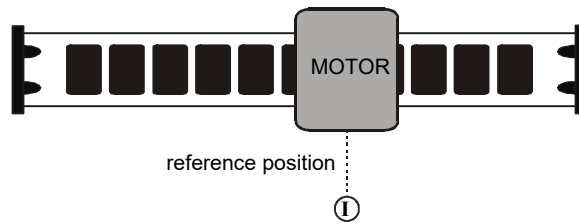
4.6.2.7.5 Method 33

Same as Method 34 with a negative movement.

This mode is equivalent to Homing mode AccurET K40.x = 9.

4.6.2.7.6 Method 37

The present position is the reference position.



This mode is equivalent to Homing mode AccurET K40.x = 22.

4.7 Sequence management

4.7.1 General description

When required by the application, it is possible to write a program that will be executed by the controller itself. This program is called 'sequence' because it is a list of instructions that are executed sequentially by the controller. These instructions can start movements, check the I/O state, change the controller registers, etc. For more information, refer to the '**AccurET Operation & Software Manual**' (section Programming of the controller). The following objects allow the management of sequences.

4.7.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1)

Index	Sub-Index	Name	Data Type	Access	Value
4003h	0	Sequence Status Word	UINT16	RO	
4004h	0	Sequence Control Word	UINT16	RW	
4005h	0	ETEL Sequence Label index			
	1	Label for Thread1	UINT32	RW	0 to 1023
	2	Label for Thread2	UINT32	RW	0 to 1023

4.7.2.1 Object 4003h: Sequence status word

This object provides the status of an ETEL Sequence.

15	10	9	8	7			2	1	0	
reserved				r2	r1	reserved			t2	t1
MSB					LSB					

t1: 1: Acknowledge the start of thread1
0: Acknowledge the stop of thread1

t2: 1: Acknowledge the start of thread2
0: Acknowledge the stop of thread2

r1: 1: Sequence is running on thread1
0: Sequence is stopped on thread1

r2: 1: Sequence is running on thread2
0: Sequence is stopped on thread2

4.7.2.2 Object 4004h: Sequence control word

This object allows to start / stop an ETEL Sequence.

15	8	7	2	1	0
unused				t2	t1
MSB				LSB	

t1: 1: Start sequence on thread 1 with label defined in object 4005:1
0: Stop sequence on thread 1 with label defined in object 4005:1

t2: 1: Start sequence on thread 2 with label defined in object 4005:2
0: Stop sequence on thread 2 with label defined in object 4005:2

Remark: A sequence is started only on rising edge of related Control Word bit.
A sequence is stopped only on falling edge of related Control Word bit.

4.7.2.3 Object 4005h: ETEL Sequence label index

This object allows to choose which label will be executed

4005:1 Label for Thread1
4005:2 Label for Thread2

4.8 Digital inputs outputs management

4.8.1 General description

AccurET controller provides several digital inputs and outputs to the user. All of these IOs can be accessed from EtherCAT master through the following objects.

For more information, refer to the '**AccurET Operation & Software Manual**' (section Digital inputs / outputs).

4.8.2 Related objects

Object 4006h is 'controller specific', therefore there is NO second axis object at [object + 800h].

On the other side, object 4007h is 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Unit
4006h	0	Fast Digital IO	UINT8	RO	
	1	Inputs	UINT32	RO	
	2	Outputs demanded	UINT32	RW	
	3	Outputs actual value	UINT32	RO	
4007h	0	Digital IO	UINT8	RO	
	1	Inputs	UINT32	RO	
	2	Outputs demanded	UINT32	RW	
	3	Outputs actual value	UINT32	RO	

4.8.2.1 Object 4006h: Fast digital IO

This object is an array of three elements which represents a mapping with some AccurET registers. The mapping is structured as follows:

Sub-index	Register	Comment
1	M52	Gives the status of the fast digital inputs of the controller: FDIN1 (bit# 0) to FDIN6 (bit# 5).
2	C5	Activates / deactivates the fast digital outputs of the controller: FDOUT1 (bit#0) to FDOUT4 (bit#3). The FDOUT are common to both axes.
3	M172	Gives the status of the fast digital outputs of the controller at the beginning of the PLTI.

4.8.2.2 Object 4007h: Digital IO

This object is an array of three elements which represents a mapping with some AccurET registers. The mapping is structured as follows:

Sub-index	Register	Comment
1	M50	Gives the status of the digital inputs of the controller per motor: DIN1 (bit#0), to DIN10 (bit# 9).
2	K171	Activates / deactivates the digital outputs of the controller per motor: DOUT1 (bit#0) and DOUT2 (bit# 1).
3	M171	Gives the status of the digital outputs of the controller per motor at the beginning of the PLTI. Takes DOUT (K171) into account

4.9 Touch Probe

Touch Probe is a latching function to capture the position value of the encoder. This capture is triggered by a digital input of the AccurET.

4.9.1 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis1).

Index	Sub-Index	Name	Data Type	Access	Units
60B8h	0	Touch Probe Function	UINT16	RW	
60B9h	0	Touch Probe status	UINT16	RO	
60BAh	0	Touch Probe 1 positive edge	INT32	RO	inc
60BBh	0	Touch Probe 1 negative edge	INT32	RO	inc
60BCh	0	Touch Probe 2 positive edge	INT32	RO	inc
60BDh	0	Touch Probe 2 negative edge	INT32	RO	inc
60D0h	0	Highest sub-index supported	UINT8	c	
	1	Touch Probe 1 source	INT16	RO	
	2	Touch Probe 2 source	INT16	RO	
60D5h	0	Touch Probe 1 positive edge counter	UINT16	RO	
60D6h	0	Touch Probe 1 negative edge counter	UINT16	RO	
60D7h	0	Touch Probe 2 positive edge counter	UINT16	RO	
60D8h	0	Touch Probe 2 negative edge counter	UINT16	RO	
4009h	0	Touch Probe Source Selection	UINT16	c	
	1	Selection of input source for Touch Probe 1	UINT16	RW	
	2	Selection of input source for Touch Probe 2	UINT16	RW	

4.9.1.1 Object 4009h: Touch Probe source selection

This object allows the selection of the digital input used for the Touch Probe function. Only one input per Touch Probe can be chosen.

Index	Sub-Index	Name	Data Type	Access	Units
4009h	0	Touch Probe source selection	UINT8	RO	
	1	Touch Probe 1 source selection	UINT16	RW	
	2	Touch Probe 2 source selection	UINT16	RW	

Depending on the AccurET model, the following digital input can be used:

- On Modular 48 or Modular 300

15	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	Index	DIN10	DIN9	0	DIN2	DIN1	FDIN6	FDIN5	FDIN4	FDIN3	FDIN2	FDIN1	
MSB													LSB

- On Modular 400

15	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	Index	DIN10	DIN9	DIN3	DIN2	DIN1	0	0	FDIN4	FDIN3	FDIN2	FDIN1	
MSB													LSB

Remark: If 2 digital inputs are selected, the **ERR CFG TOUCHPRO** error is raised.
If a digital input is selected while it is already used for another feature, the same **ERR CFG TOUCHPRO** error is raised.

4.9.1.2 Object 60B8h: Touch Probe mode

This object set the configured mode of the Touch Probe.

Bit	Value	Definition
0	0	Switch off Touch Probe 1
	1	Enable Touch Probe 1
1	0	Trigger first event
	1	Continuous
2,3	0	Trigger with Touch Probe on FDIN1
	1	Trigger with zero impulse signal of position encoder
	2	Touch Probe source as defined in object 60D0h, sub-index 01
	3	Reserved
4	0	Switch off sampling at positive edge of Touch Probe 1
	1	Enable sampling at positive edge of Touch Probe 1
5	0	Switch off sampling at negative edge of Touch Probe 1
	1	Enable sampling at negative edge of Touch Probe 1
6,7		Reserved
8	0	Switch off Touch Probe 2
	1	Enable Touch Probe 2
9	0	Trigger first event
	1	Continuous
10,11	0	Trigger with Touch Probe on FDIN2
	1	Trigger with zero impulse signal of position encoder
	2	Touch Probe source as defined in object 60D0h, sub-index 02
	3	Reserved
12	0	Switch off sampling at positive edge of Touch Probe 2
	1	Enable sampling at positive edge of Touch Probe 2

Bit	Value	Definition
13	0	Switch off sampling at negative edge of Touch Probe 2
	1	Enable sampling at negative edge of Touch Probe 2
14,15	-	Reserved

4.9.1.3 Object 60B9h: Touch Probe status

This object provides the status of the Touch Probe. Next table specifies the possible values.

Bit	Value	Definition
0	0	Touch Probe 1 is switched off
	1	Touch Probe 1 is enabled
1	0	Touch Probe 1 no positive edge value stored
	1	Touch Probe 1 positive edge value stored
2	0	Touch Probe 1 no negative edge value stored
	1	Touch Probe 1 negative edge position stored
3 to 5	0	Reserved
6,7	0	Reserved
8	0	Touch Probe 2 is switched off
	1	Touch Probe 2 is enabled
9	0	Touch Probe 2 no positive edge value stored
	1	Touch Probe 2 positive edge value stored
10	0	Touch Probe 2 no negative edge value stored
	1	Touch Probe 2 negative edge position stored
11 to 13	0	Reserved
14,15	0	Reserved

4.9.1.4 Object 60BAh: Touch Probe 1 positive edge

This object provides the position value of the Touch Probe 1 at positive edge. The value is given in user-defined position units.

4.9.1.5 Object 60BBh: Touch Probe 1 negative edge

This object provides the position value of the Touch Probe 1 at negative edge. The value is given in user-defined position units.

4.9.1.6 Object 60BCh: Touch Probe 2 positive edge

This object provides the position value of the Touch Probe 2 at positive edge. The value is given in user-defined position units.

4.9.1.7 Object 60BDh: Touch Probe 2 negative edge

This object provides the position value of the Touch Probe 2 at negative edge. The value is given in user-defined position units.

4.9.1.8 Object 60D0h: Touch Probe source

This object provides the source of the Touch Probe function. The following values are available:

Value	Definition
-11	DIN10
-10	DIN9
-9	DIN3

Value	Definition
-8	DIN2
-7	DIN1
-6	FDIN6
-5	FDIN5
-4	FDIN4
-3	FDIN3
-2	FDIN2
-1	FDIN1
0	Reserved
5	Hardware zero impulse signal of position encoder

4.9.1.9 Object 60D5h: Touch Probe 1 positive edge counter

This object provides a continuous counter that is incremented with each positive edge at Touch Probe 1. The counter is only valid if the Touch Probe input is enabled (60B8h, bit 0 = 1b).

For single event measuring only the value of bit 0 shall be evaluated. For continuous measuring the value is an unsigned 16-bit value with overflow.

4.9.1.10 Object 60D6h: Touch Probe 1 negative edge counter

Similar to previous object 60D5h, but sensitive to the negative edge.

4.9.1.11 Object 60D7h: Touch Probe 2 positive edge counter

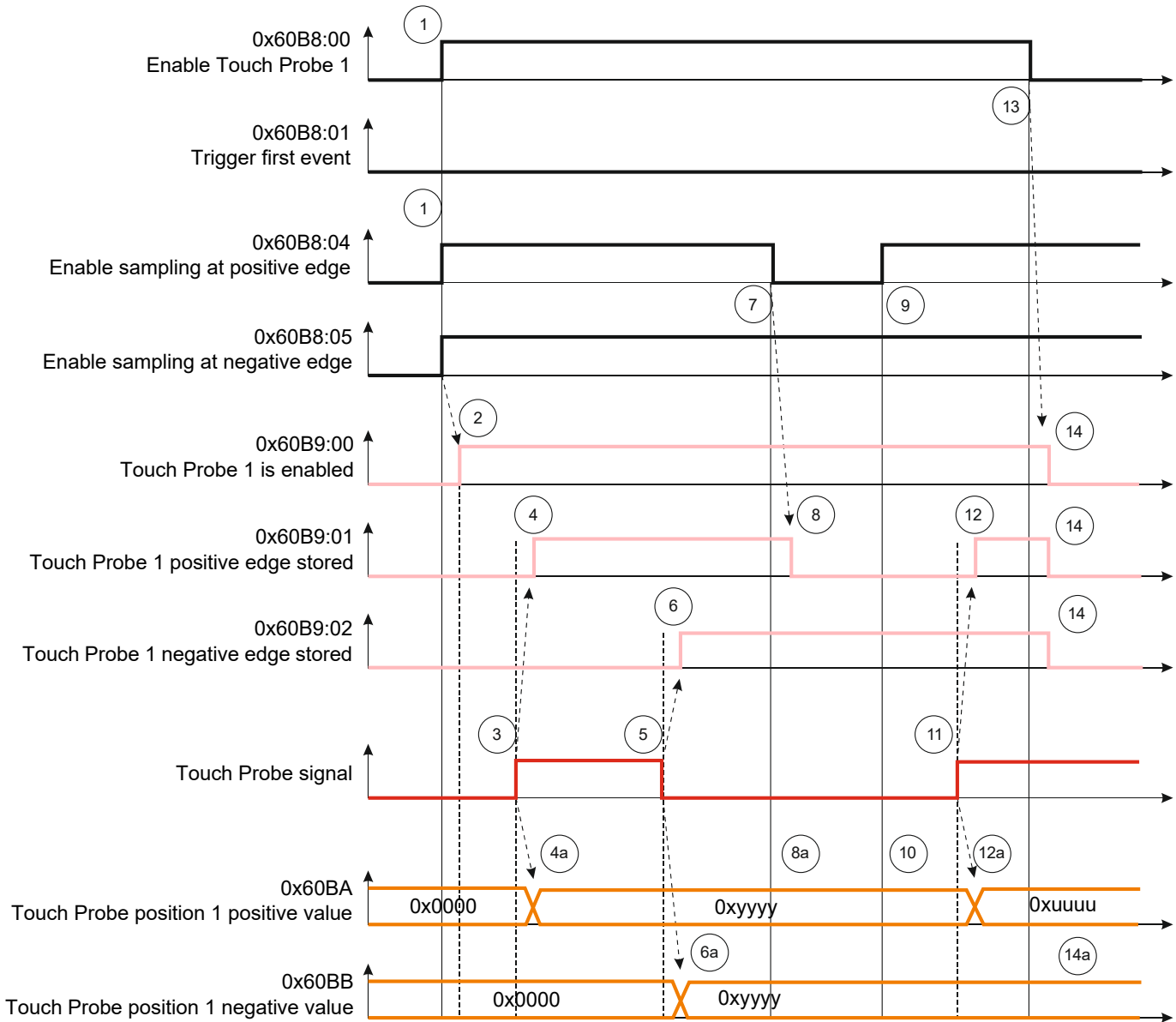
This object provides a continuous counter that is incremented with each positive edge at Touch Probe 2. The counter is only valid if the Touch Probe input is enabled (60B8h, bit 8 = 1b). For single event measuring only the value of bit 0 shall be evaluated. For continuous measuring the value is an unsigned 16-bit value with overflow.

4.9.1.12 Object 60D8h: Touch Probe 2 negative edge counter

Similar to previous object 60D7h, but sensitive to the negative edge.

4.9.2 Example

The following figure shows a timing diagram for a Touch Probe configuration and its corresponding behavior.



Number	Touch Probe behavior	
(1)	0x60B8:00	Enable Touch Probe1
	0x60B8:01: 04.:05	Configure and Enable Touch Probe 1 positive and negative edge
(2)	0x60B9:00	Status "Touch Probe 1 enabled"
(3)	External touch prove signal has positive edge	
(4)	0x60B9:01	Status "Touch Probe 1 positive edge stored" is set
(4a)	0x60BA	Touch Probe position 1 positive value is stored
(5)	External Touch Probe signal has negative edge	
(6)	0x60B9:02	Status "Touch Probe 1 negative edge stored" is set
(6a)	Touch Probe position 1 negative value is stored	
(7)	0x60B8:04	Sample positive edge is disabled
(8)	0x60B9:00	Status "Touch Probe 1 positive edge stored" is reset
(8a)	0x60BA	Touch Probe position 1 positive is not changed

Number	Touch Probe behavior	
(9)	0x60B8:04	Sample positive edge is enabled
(10)	0x60BA	Touch Probe position 1 positive is not changed
(11)	External Touch Probe signal has negative edge	
(12)	0x60B9:01	Status "Touch Probe 1 positive edge stored" is set
(12a)	0x60BA	Touch Probe position 1 positive value is stored
(13)	0x60B8:00	Touch Probe 1 is disabled
(14)	0x60B9:00,; 01 :12	Status "Touch Probe 1 are reset
(14a)	0x60BA, 0x60BA	Touch Probe position 1 positive/negative value are not changed

4.10 Controller register access

4.10.1 General description

AccurET controller provides several registers to configure or command it. All of these registers can be accessed through SDO object 4008h.

4.10.2 Related objects

Object 4008h is 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Unit
4008h	0	Register Access	UINT8	RO	
	1	Control Word	UINT16	RW	
	2	Status Word	UINT16	RO	
	3	Type	UINT16	RW	
	4	Data Type	UINT16	RW	
	5	Number	UINT16	RW	
	6	Depth	UINT16	RW	
	7	Data (low-part)	UINT32	RW	
	8	Data (high-part)	UINT32	RW	

4.10.2.1 Object 4008h: Register access

This object has 8 sub-indexes:

4.10.2.1.1 Sub-object 1: Control word

This object indicates the command controlling the register access.

15								1	0
								write	read
MSB								LSB	

- Bit 0 enable reading request command
 Bit 1 enable writing request command

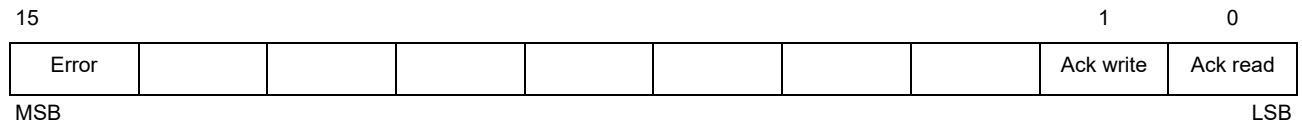
In any case of reading or writing, the objects *Type* (sub-object 3), *Data Type* (sub-object 4), *Number* (sub-object 5) and *Depth* (sub-object 6) MUST have been set before setting bits 0 or 1 to 1.

When writing, the object *Data (low-part)* (sub-object 7) or *Data (high-part)* (sub-object 8) MUST equally have been set before setting bit 1 to 1.

The application must wait *Status Word (sub-object 2)* corresponding *Ack* bit to be 1. Only raising edge of request command is considered. So before issuing another write/read command, the application sets the command back to 0 in *Control Word (sub-object 1)* and wait again for *Status Word (sub-object 2)* *Ack* bit to be back to 0. Only then another write/read command can be requested.

4.10.2.1.2 Sub-object 2: Status word

This object indicates the command controlling the register access.



Bit 0 acknowledge reading request
 Bit 1 acknowledge writing request
 Bit 15 indicates an error occurred

When an acknowledge bit is set to 1, this means that either *Data sub-objects* are read and set by the controller (when reading) or the register has been written (when writing) to the controller.

Bit 15 is set to 1 when the request failed. This bit is only valid when the command bit in *Control Word (sub-object 1)* and corresponding *Ack* bit in *Status Word (sub-object 2)* are set to 1. In any other case, this bit has no meaning.

4.10.2.1.3 Sub-object 3: Type

This object indicates which type of register needs to be accessed.

Type number	AccurET register type
0	M register
1	K register
2	C register
3	X register

4.10.2.1.4 Sub-object 4: Data type

This object indicates which data type of register needs to be accessed.

Type number	AccurET register data type
0	Integer
1	Long
2	Float
3	Double

4.10.2.1.5 Sub-object 5: Number

Number of register to be accessed.

4.10.2.1.6 Sub-object 6: Depth

Depth of register to be accessed.

4.10.2.1.7 Sub-object 7: Data (low-part)

Contains the data to be read or written.

In case of data type Long or Double this register indicates the low part of the data.

4.10.2.1.8 Sub-object 8: Data (high-part)

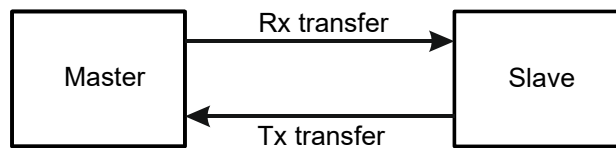
Contains the data to be read or written.

In case of data type Long or Double this register indicates the high part of the data.

In case of data type Integer or Float this register is ignored.

4.11 RTV equivalent**4.11.1 General description**

TransnET protocol allows the user to map AccurET registers to slots, to get real-time communication between devices. A similar method has been implemented in EtherCAT. In EtherCAT, the user set a selection of AccurET registers in PDO. Up to 16 registers can be selected for Rx transfer and up to 16 Tx transfer too.



For Rx transfer, the master values selected using the user-defined PDO 160Fh, will be copied synchronously to M450-M465, ML450-ML457 or MF450-MF465. The user is free to choose.

For Tx transfer, the user can choose any AccurET registers. They will be transferred synchronously to the master using the mapping defined in the user TxPDO 1A0Fh.

4.11.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

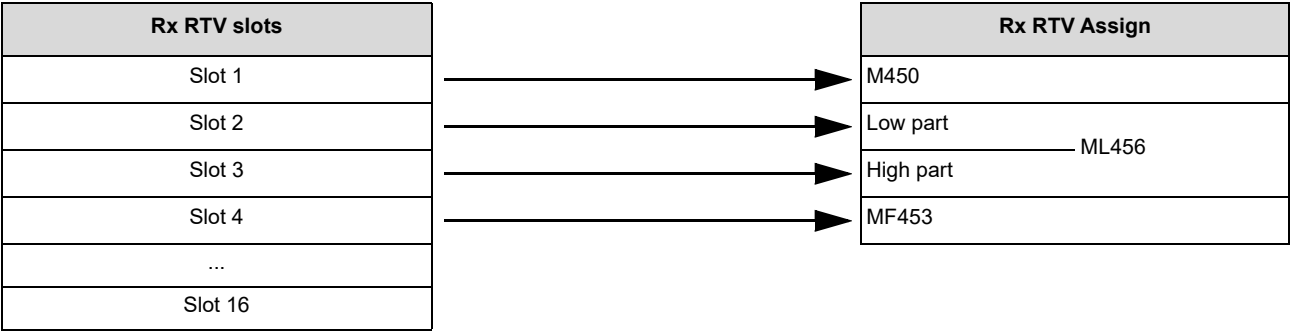
Index	Sub-index	Name	Data type	Access	Value
400Ah	0	Rx RTV Assign	UINT8	RW	
	1	Assignment 1	UINT32	RW	
	...		UINT32	RW	
	16	Assignment 1	UINT32	RW	
400Bh	0	Tx RTV Assign	UINT8	RW	
	1	Assignment 1	UINT32	RW	
	...		UINT32	RW	
	16	Assignment 1	UINT32	RW	
400Ch	0	Rx RTV Slots	UINT8	RO	
	1	Slot 1	UINT32	RW	
	...		UINT32	RW	
	16	Slot 16	UINT32	RW	
400Dh	0	Tx RTV Slots	UINT8	RO	
	1	Slot 1	UINT32	RO	
	...		UINT32	RO	
	16	Slot 16	UINT32	RO	

4.11.2.1 Object 400Ah: Rx RTV Assign

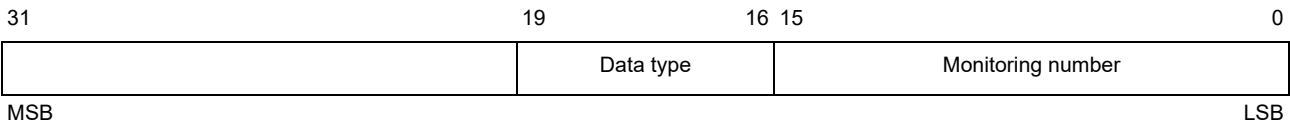
This object indicates the list of monitoring registers associated to RTV slots.

Authorized monitorings are M450-M465, ML450-ML457 and MF450-MF465. When an MLxxx monitoring is set, the data of this object is mapped on two slots.

The following figure shows how the monitorings are mapped on Rx RTV slots object.



The format of the value to be set:



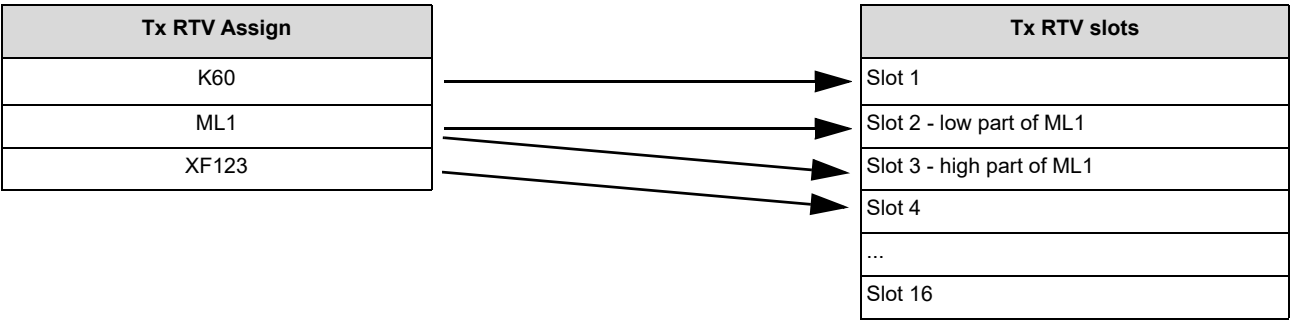
With Data type coded as follow:

Type number	AccurET register data type
0	Integer
1	Long
2	Float

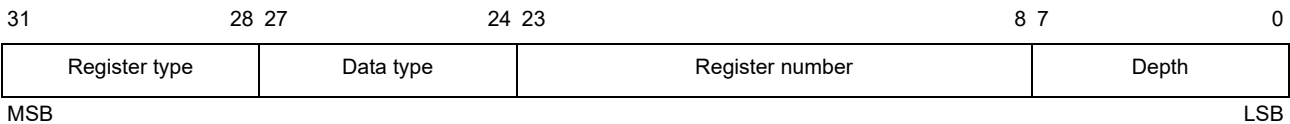
Caution: The drive must be in PREOP state to be able to change the values of this object.

4.11.2.2 Object 400Bh: Tx RTV Assign

This object indicates the list of registers associated to Tx RTV slots. When a register of 8 bytes long is set, the data of this object is mapped on two slots. The following figure shows how the monitoring registers are mapped on Tx RTV Slots object.



The format of the value to be set:



With register type:

Register type	AccurET register type
0	M register
1	K register

Register type	AccurET register type
2	C register
3	X register

With data type:

Data type	AccurET register type
0	Integer
1	Long
2	Float
3 (only for X registers)	Double

Caution: This object must be set in PREOP state

4.11.2.3 Object 400Ch: Rx RTV Slots

This object indicates the RTV values set with Rx RTV Assign object. This object is an array of 16 x INT32. This means maximum 512 bytes are allocated for transferring data. When assigning the monitorings, the user must consider this limitation. Data in this object are only relevant in OP state. For real-time registers sharing between master and AccurET, objects in this array can be mapped into PDO object 160Fh.

4.11.2.4 Object 400Dh: Tx RTV Slots

This object indicates the RTV values set with Tx RTV Assign object. This object is an array of 16 x INT32. This means maximum 512 bytes are allocated for transferring data. When assigning the registers the user must consider this limitation. Data in this object are only relevant in OP state. For real-time registers sharing between Master and AccurET, objects in this array can be mapped into PDO object 1A0Fh.

4.12 Information and diagnosis data

4.12.1 General description

AccurET controller provides information and diagnosis data to inform the master about its state.

4.12.2 Related objects

Object 10F3h is 'controller specific', therefore there is NO object at [object + 800h].

Index	Sub-Index	Name	Data Type	Access	Unit
10F3h	0	Diagnosis History	UINT8	RO	
	1	Maximum Messages	UINT8	RO	
	2	Newest Message	UINT8	RO	
	3	Newest Acknowledged Message	UINT8	RW	
	4	New Messages Available	BOOLEAN	RO	
	5	Flags	UINT16	RO	
	6	Message 1	OCTET STRING[28]	RO	
	OCTET STRING[28]	RO	
	25	Message 20	OCTET STRING[28]	RO	

4.12.2.1 Object 10F3h: Diagnosis history

This object is a record with the following sub-object meaning:

Index	Name	Meaning	
10F3:01	Maximum Messages	Maximum number of stored messages. A maximum of 20 messages can be stored	
10F3:02	Newest Message	Subindex of the latest message	
10F3:03	Newest Acknowledged Message	Subindex of the last confirmed message	
10F3:04	New Messages Available	Indicates that a new message is available	
10F3:05	Flags	Bit 0	Send as emergency
		Bit 1	Disable Info messages
		Bit 2	Disable Warning messages
		Bit 3	Disable Error messages
		Bit 4	Indicates the diagnosis history mode (0 "overwrite" Mode; 1 "acknowledge" Mode)
		Bit 5	Indicates if messages were overwritten ("overwrite" mode) or new messages were discard ("acknowledge" mode)
10F3:06	Message 1	Message 1	
..	...		
10F3:25	Message 20	Message 20	

4.13 Other objects

4.13.1 General description

This section describes some useful objects which are available in all modes.

4.13.2 Related objects

These objects are 'module specific', so it exists an equivalent object at [object + 800h] to access module 1 (axis 1).

Index	Sub-Index	Name	Data Type	Access	Unit
6007h	0	Abort Connection Option Code	INT16	RW	N/A
6075h	0	Motor Rated Current	UINT32	RW	mA
6076h	0	Motor Rated Torque	UINT32	RW	[mN/A] or [mNm/A]
6077h	0	Torque Actual Value	INT16	RO	%
6078h	0	Current Actual Value	UINT16	RO	%
607Bh	0	Position Range Limit	UINT8	RO	inc
	1	Min Position Limit	INT32	RW	
	2	Max Position Limit	INT32	RW	
607Dh	0	Software Position Limit	UINT8	RO	inc
	1	Min Position Limit	INT32	RW	
	2	Max Position Limit	INT32	RW	

4.13.2.1 Object 6007h: Abort connection option code

This object indicates what action to be performed in case of a communication error (Slave ESM leaves OP state).

0 No action 1 Fault signal 2 Disable voltage command 3 Quick stop command

4.13.2.2 Object 6075h: Motor rated current

This object indicates the max current the motor can support in [mA]. This object represents the AccurET register MF82.

4.13.2.3 Object 6076h: Motor rated torque

This object indicates the max torque the motor can provide in [mN/A] or [mNm/A] depending on the type of motor.

Since the controller is only able to measure the current provided to the motor, the torque is computed with Kt constant.

The Kt constant has a linear behavior at any speed when the controller works with an ironless motor. Though in case of an ironcore motor, the Kt is not linear anymore when the speed is higher than the nominal speed. In that case, the torque does not have a real meaning in order that no compensation is implemented in the AccurET.

4.13.2.4 Object 6077h: Torque actual value

This object indicates the actual value of the torque. The value is given per thousand of *Motor Rated Torque* (object 6076h).

4.13.2.5 Object 6078h: Current actual value

This object indicates the actual value of the current. The value is given per thousand of *Motor Rated Current* (object 6075h).

4.13.2.6 Object 607Bh: Position range limit

This object indicates the configured maximal and minimal position range limits in [inc]. On reaching or exceeding these limits, the input value wraps automatically to the other end of range.

This is same behavior as register MMD (K202) set to 17 and register K209 set to 2.

To disable this functionality, set 0 to both parameters.

For more information, refer to the '**AccurET Operation & Software Manual**' (section Linear or rotary movement).

4.13.2.7 Object 607Dh: Software position limit

This object indicates the configured maximal and minimal software position limits in [inc]. Every new target position are checked against these limits. If a required target position is beyond these limits, the axis moves to the limit and no error is raised.

This is same behavior as register MODESL (K36) set to 1.

To disable this functionality, set 0 to both parameters.

For more information, refer to the '**AccurET Operation & Software Manual**' (section Movements limits).

5 Object dictionary

The following table shows all objects of module 0 (Axis 0). If the Object is “module specific”, the module offset needs to be added to the index to get the objects of Module1. If no “module offset” is specified, the object is “controller specific” and is the same for axis 1.

Index	Sub-Index	Name	Data Type	Access	Module offset	AccurET register
1000h	00h	Device Type	UINT32	RO		
1001h	00h	Error Register	UINT8	RO		
1008h	00h	Device Name	STRING[18]	RO		
1009h	00h	Hardware Version	STRING[18]	RO		
100Ah	00h	Software Version	STRING[18]	RO		
1018h	00h	Identity	UINT8	RO		
	01h	Vendor ID	UINT32	RO		
	02h	Product Code	UINT32	RO		
	03h	Revision	UINT32	RO		
	04h	Serial Number	UINT32	RO		M73
10F1h	00h	Error Settings	UINT8	RO		
	01h	Local Error Reaction	UINT32	RW		
	02h	Sync Error Counter Limit	UINT16	RW		
10F3h	00h	Diagnosis History	UINT8	RO		
	01h	Maximum Messages	UINT8	RO		
	02h	Newest Message	UINT8	RO		
	03h	Newest Acknowledged Message	UINT8	RW		
	04h	New Messages Available	BOOL	RO		
	05h	Flags	UINT16	RW		
	06h	Diagnosis Message 1	OCTET STRING[28]	RO		
	OCTET STRING[28]	RO		
	19h	Diagnosis Message 20	OCTET STRING[28]	RO		
10F8h	00h	Timestamp Object	UINT64	RO		
1600h	00h	RxPDO PP	UINT8	RO	10h	
	01h	Control Word	UINT32	RO	10h	
	02h	Target Position	UINT32	RO	10h	
1601h	00h	RxPDO CSP	UINT8	RO	10h	
	01h	Control Word	UINT32	RO	10h	
	02h	Target Position	UINT32	RO	10h	
1602h	00h	RxPDO Sequence	UINT8	RO	10h	
	01h	Sequence Control Word	UINT32	RO	10h	
1603h	00h	RxPDO Digital IO	UINT8	RO	10h	
	01h	Demanded Digital Outputs	UINT32	RO	10h	
1604h	00h	RxPDO Fast Digital IO	UINT8	RO		
	01h	Demanded Fast Digital Outputs	UINT32	RO		
1607h	00h	RxPDO Touch Probe	UINT8	RO	10h	
	01h	Touch Probe Function	UINT32	RO	10h	
1608h	00h	RxPDO Mode Of Operation	UINT8	RO	10h	
	01h	Mode Of Operation	UINT32	RO	10h	

Index	Sub-Index	Name	Data Type	Access	Module offset	AccurET register
160Fh	00h	RxPDO Custom	UINT8	RW	800h	
	01h	Assignment 1	UINT32	RW	800h	
	UINT32	RW	800h	
	20h	Assignment 32	UINT32	RW	800h	
1706h	00h	RxPDO CSP with Latch	UINT8	RO	10h	
	01h	Control Word	UINT32	RO	10h	
	02h	Target Position	UINT32	RO	10h	
	03h	Latch Control Word	UINT32	RO	10h	
1A00h	00h	TxPDO PP	UINT8	RO	10h	
	01h	Status Word	UINT32	RO	10h	
	02h	Actual Position	UINT32	RO	10h	
1A01h	00h	TxPDO CSP	UINT8	RO	10h	
	01h	Status Word	UINT32	RO	10h	
	02h	Actual Position	UINT32	RO	10h	
1A02h	00h	TxPDO Sequence	UINT8	RO	10h	
	01h	Sequence Status Word	UINT32	RO	10h	
1A03h	00h	TxPDO Digital IO	UINT8	RO	10h	
	01h	Digital Inputs	UINT32	RO	10h	
	02h	Actual Digital Outputs	UINT32	RO	10h	
1A04h	00h	TxPDO Fast Digital IO	UINT8	RO		
	01h	Fast Digital Inputs	UINT32	RO		
	02h	Actual Fast Digital Outputs	UINT32	RO	10h	
1A05h	00h	TxPDO Motion Information	UINT8	RO	10h	
	01h	Velocity Actual Value	UINT32	RO	10h	
	02h	Current Actual Value	UINT32	RO	10h	
	03h	Torque Actual Value	INT16	RO	10h	
	04h	Following Error Actual Value	UINT32	RO	10h	
	05h	Actual Demand Internal Value	UINT32	RO	10h	
1A07h	00h	TxPDO Touch Probe	UINT8	RO	10h	
	01h	Touch Probe Status	UINT32	RO	10h	
	02h	Touch Probe 1 Positive Edge	UINT32	RO	10h	
	03h	Touch Probe 1 Negative Edge	UINT32	RO	10h	
	04h	Touch Probe 2 Positive Edge	UINT32	RO	10h	
	05h	Touch Probe 2 Negative Edge	UINT32	RO	10h	
	06h	Touch Probe 1 Positive Edge Counter	UINT32	RO	10h	
	07h	Touch Probe 1 Negative Edge Counter	UINT32	RO	10h	
	08h	Touch Probe 2 Positive Edge Counter	UINT32	RO	10h	
	09h	Touch Probe 2 Negative Edge Counter	UINT32	RO	10h	
1A08h	00h	TxPDO Mode Of Operation	UINT8	RO	10h	
	01h	Mode Of Operation Display	UINT32	RO	10h	
	02h	Supported Drive Modes	UINT32	RO	10h	
1A0Fh	00h	TxPDO Custom	UINT8	RW	800h	
	01h	Assignment 1	UINT32	RW	800h	
	UINT32	RW	800h	
	20h	Assignment 32	UINT32	RW	800h	

Index	Sub-Index	Name	Data Type	Access	Module offset	AccurET register
1B06h	00h	TxPDO CSP with Latch	UINT8	RO	10h	
	01h	Status Word	UINT32	RO	10h	
	02h	Actual Position	UINT32	RO	10h	
	03h	Latch Status Word	UINT32	RO	10h	
	04h	Latch Position	UINT32	RO	10h	
1C00h	00h	Sync Manager Type	UINT8	RO		
	01h	SubIndex 001	UINT8	RO		
	02h	SubIndex 002	UINT8	RO		
	03h	SubIndex 003	UINT8	RO		
	04h	SubIndex 004	UINT8	RO		
1C12h	00h	RxPDO Assign	UINT8	RW		
	01h	SubIndex 1	UINT16	RW		
	02h	SubIndex 2	UINT16	RW		
	UINT16	RW		
	10h	SubIndex 016	UINT16	RW		
1C13h	00h	TxPDO Assign	UINT8	RO		
	01h	SubIndex 1	UINT16	RW		
	02h	SubIndex 2	UINT16	RW		
	...		UINT16	RW		
	10h	SubIndex 016	UINT16	RW		
1C32h	00h	SM Output Parameter	UINT8	RO		
	01h	Synchronization Type	UINT16	RO		
	02h	Cycle Time	UINT32	RO		
	04h	Synchronization Types Supported	UINT16	RO		
	05h	Minimum Cycle Time	UINT32	RO		
	06h	Calc and Copy Time	UINT32	RO		
	08h	Get Cycle Time	UINT16	RW		
	09h	Delay Time	UINT32	RO		
	0Ah	Synch0 Cycle Time	UINT32	RO		
	0Bh	SM-Event Missed	UINT16	RO		
	0Ch	Cycle Time Too Small	UINT16	RO		
	20h	Sync Error	UINT8	RO		
1C33h	00h	SM Input Parameter	UINT8	RO		
	01h	Synchronization Type	UINT16	RO		
	02h	Cycle Time	UINT32	RO		
	04h	Synchronization Types Supported	UINT16	RO		
	05h	Minimum Cycle Time	UINT32	RO		
	06h	Calc and Copy Time	UINT32	RO		
	08h	Get Cycle Time	UINT16	RW		
	09h	Delay Time	UINT32	RO		
	0Ah	Synch0 Cycle Time	UINT32	RO		
	0Bh	SM-Event Missed	UINT16	RO		
	0Ch	Cycle Time Too Small	UINT16	RO		
	20h	Sync Error	UINT8	RO		
2802h	00h	Latch Control Word	UINT16	RW	800h	

Index	Sub-Index	Name	Data Type	Access	Module offset	AccurET register
2901h	00h	Latch Status Word	UINT16	RO	800h	
2902h	00h	Latch Position	INT32	RO	800h	
4000h	00h	Motor Information	UINT8	RO	800h	
	01h	Type	UINT16	RO	800h	
4001h	00h	Encoder Information	UINT8	RO	800h	
	01h	Period	UINT32	RO	800h	M239
	02h	Interpolation	UINT32	RO	800h	M241
4002h	00h	Controller Information	UINT8	RO	800h	
	01h	Controller Status 1	UINT32	RO	800h	M60
	02h	Controller Status 2	UINT32	RO	800h	M61
	03h	Controller Status	UINT32	RO	800h	M63
	04h	MLTI [10*ns]	UINT32	RO	800h	
	05h	Movement Limit Min (low-part)	INT32	RO	800h	KL34(1sl)
	06h	Movement Limit Min (high-part)	INT32	RO	800h	KL34(msl)
	07h	Movement Limit Max (low-part)	INT32	RO	800h	KL35(1sl)
	08h	Movement Limit Max (high-part)	INT32	RO	800h	KL35(msl)
	09h	Timeout Delayed Power Off (ms)	UINT32	RW	800h	
4003h	00h	Sequence Status Word	UINT16	RO	800h	
4004h	00h	Sequence Control Word	UINT16	RW	800h	
4005h	00h	Sequence Label Index	UINT8	RO	800h	
	01h	Label for Thread1	UINT32	RW	800h	
	02h	Label for Thread2	UINT32	RW	800h	
4006h	00h	Fast Digital IO	UINT8	RO		
	01h	Fast Digital Inputs	UINT32	RO		M52
	02h	Demanded Fast Digital Outputs	UINT32	RW		C5
	03h	Actual Fast Digital Outputs	UINT32	RO		M172
4007h	00h	Digital IO	UINT8	RO	800h	
	01h	Digital Inputs	UINT32	RO	800h	M50
	02h	Demanded Digital Outputs	UINT32	RW	800h	K171
	03h	Actual Digital Outputs	UINT32	RO	800h	M171
4008h	00h	Register Access	UINT8	RO	800h	
	01h	Control Word	UINT16	RW	800h	
	02h	Status Word	UINT16	RO	800h	
	03h	Type	UINT16	RW	800h	
	04h	Data Type	UINT16	RW	800h	
	05h	Number	UINT16	RW	800h	
	06h	Depth	UINT16	RW	800h	
	07h	Data (low-part)	UINT32	RW	800h	
	08h	Data (high-part)	UINT32	RW	800h	
4009h	00h	Source reservation	UINT8	RO	800h	
	01h	Touch Probe 1	UINT16	RW	800h	
	02h	Touch Probe 2	UINT16	RW	800h	

Index	Sub-Index	Name	Data Type	Access	Module offset	AccurET register
400Ah	00h	Rx RTV Assign	UINT8	RW	800h	
	01h	Assignment 1	UINT32	RW	800h	
	UINT32	RW	800h	
	10h	Assignment 16	UINT32	RW	800h	
400Bh	00h	Tx RTV Assign	UINT8	RW	800h	
	01h	Assignment 1	UINT32	RW	800h	
	UINT32	RW	800h	
	10h	Assignment 16	UINT32	RW	800h	
400Ch	00h	Rx RTV Slots	UINT8	RO	800h	
	01h	Slot 1	UINT32	RO	800h	
	UINT32	RO	800h	
	10h	Slot 16	UINT32	RO	800h	
400Dh	00h	Tx RTV Slots	UINT8	RO	800h	
	01h	Slot 1	UINT32	RO	800h	
	UINT32	RO	800h	
	10h	Slot 16	UINT32	RO	800h	
6007h	00h	Abort Connection Option Code	INT16	RW	800h	
603Fh	00h	Error Code	UINT16	RO	800h	
6040h	00h	Control Word	UINT16	RW	800h	
6041h	00h	Status Word	UINT16	RO	800h	
605Ah	00h	Quick Stop Option Code	UINT16	RW	800h	
605Dh	00h	Halt Option Code	INT16	RW	800h	
6060h	00h	Mode of Operation	INT8	RW	800h	
6061h	00h	Mode of Operation Display	INT8	RO	800h	
6064h	00h	Position Actual Value	INT32	RO	800h	M1
6065h	00h	Following Error Window	UINT32	RW	800h	K30
6066h	00h	Following Error Timeout	UINT16	RO	800h	
606Ch	00h	Velocity Actual Value	INT32	RO	800h	
6075h	00h	Motor Rated Current	UINT32	RW	800h	
6076h	00h	Motor Rated Torque	UINT32	RW	800h	
6077h	00h	Torque Actual Value	INT16	RO	800h	
6078h	00h	Current Actual Value	INT16	RO	800h	
607Ah	00h	Target Position	INT32	RW	800h	
607Bh	00h	Position Range Limit	UINT8	RO	800h	
	01h	Min Position Limit	INT32	RW	800h	
	02h	Max Position Limit	INT32	RW	800h	
607Ch	00h	Homing Offset	INT32	RW	800h	
607Dh	00h	Software Position Limit	UINT8	RO	800h	
	01h	Min Position Limit	INT32	RW	800h	
	02h	Max Position Limit	INT32	RW	800h	
607Fh	00h	Max profile velocity	UINT32	RW	800h	
6080h	00h	Max Motor Speed	UINT32	RW	800h	
6081h	00h	Profile Velocity	UINT32	RW	800h	
6083h	00h	Profile Acceleration / Deceleration	UINT32	RW	800h	
6085h	00h	Quick Stop Deceleration	UINT32	RW	800h	

Index	Sub-Index	Name	Data Type	Access	Module offset	AccurET register
6098h	00h	Homing Method	INT8	RW	800h	
6099h	00h	Homing Speeds	UINT8-	RO	800h	
	01h	SubIndex 001	UINT32	RW	800h	KL41
	02h	SubIndex 002	UINT32	RW	800h	KL41
609Ah	00h	Homing Acceleration	UINT32	RW	800h	KL42
60B2h	00h	Torque Offset	INT16	RW	800h	
60B8h	00h	Touch Probe Function	UINT16	RW	800h	
60B9h	00h	Touch Probe status	UINT16	RO	800h	
60BAh	00h	Touch Probe 1 positive edge	INT32	RO	800h	
60BBh	00h	Touch Probe 1 negative edge	INT32	RO	800h	
60BCh	00h	Touch Probe 2 positive edge	INT32	RO	800h	
60BDh	00h	Touch Probe 2 negative edge	INT32	RO	800h	
60C2h	00h	Interpolation Time	UINT8	RO	800h	
	01h	Interpolation Time Period	UINT8	RW	800h	
	02h	Interpolation Time Index	INT8	RW	800h	
60D0h	00h	Highest sub-index supported	UINT8	RO	800h	
	01h	Touch Probe 1 source	INT16	RO	800h	
	02h	Touch Probe 2 source	INT16	RO	800h	
60D5h	00h	Touch Probe 1 positive edge counter	UINT16	RO	800h	
60D6h	00h	Touch Probe 1 negative edge counter	UINT16	RO	800h	
60D7h	00h	Touch Probe 2 positive edge counter	UINT16	RO	800h	
60D8h	00h	Touch Probe 2 negative edge counter	UINT16	RO	800h	
60E3h	00h	Supported Homing Methods	UINT8	RO	800h	
	01h	SubIndex 001	INT8	RO	800h	
	02h	SubIndex 002	INT8	RO	800h	
	03h	SubIndex 003	INT8	RO	800h	
	04h	SubIndex 004	INT8	RO	800h	
	05h	SubIndex 005	INT8	RO	800h	
	06h	SubIndex 006	INT8	RO	800h	
60F2h	00h	Positioning Option Code	UINT16	RW	800h	
60F4h	00h	Following Error Actual Value	INT32	RO	800h	
60FCh	00h	Position Demand Internal Value	INT32	RO	800h	
6502h	00h	Supported Drive Modes	UINT32	RO	800h	
F000h	00h	Modular Device Profile	UINT8	RO		
	01h	Module Index Distance	UINT16	RO		
	02h	Maximum Number of Modules	UINT16	RO		
F010h	00h	Module Profile List	UINT8	RO		
	01h	SubIndex 001	UINT32	RO		
	02h	SubIndex 002	UINT32	RO		
F030h	00h	Configured Module Ident List	UINT8	RW		
	01h	SubIndex 001	UINT32	RW		
	02h	SubIndex 002	UINT32	RW		
F050h	00h	Detected Module Ident List	UINT8	RO		
	01h	SubIndex 001	UINT32	RO		
	02h	SubIndex 002	UINT32	RO		

6 Appendixes

6.1 Configuration

6.1.1 AccurET configuration for EtherCAT

EtherCAT and TransnET protocols cannot be used at the same time. By default AccurET comes with default parameters set to use the TransnET mode. To configure the AccurET to work with EtherCAT mode, do as follows:

1. Set the register C1 to 2 in ComET terminal, as shown in the following figure.
2. Save the controller parameter.
3. Reboot the controller to use EtherCAT mode.

```
# USB
# ISO linear units : mm, mm/s, mm/s², mV, mA, ms (used in case of interpolation)
# ISO rotary units : deg, deg/s, deg/s², V, A, s
♦ C1.0=2 ; set Manager mode
♦ SAV.0=4 ; Controller save
✕ RSD.0=255 ; Resets controller
▶ |
```

A similar procedure with C1 = 0 allows the user to switch back to TransnET mode.

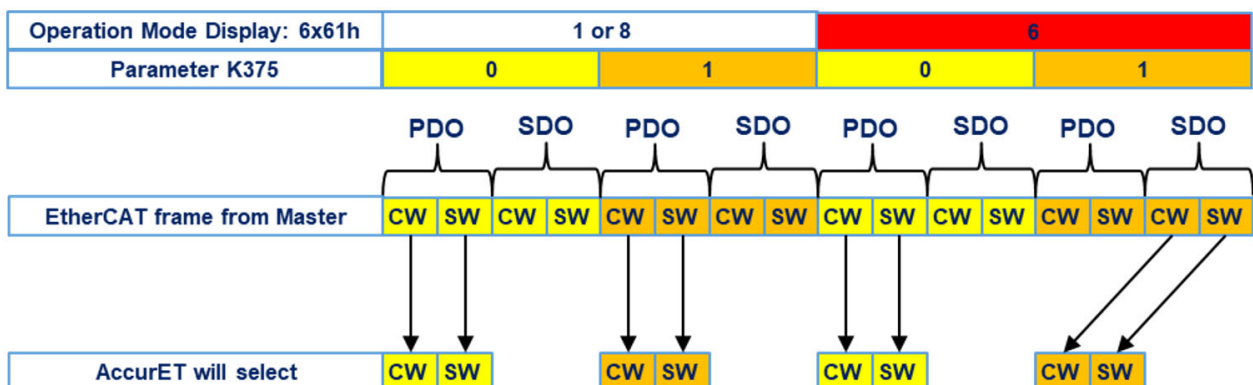
At this point the controller works with EtherCAT, no further parametrization is required regarding the communication.

The system commissioning (motor, encoder, control, etc.) must still be done with ComET as usual way. If EtherCAT is already enabled, communication between AccurET and ComET must be done through USB.

Once EtherCAT mode is started, the user can only communicate with AccurET by using EtherCAT or directly with a USB connection and ComET commissioning software. TCP/IP mode and TransnET communication through UltimET are disabled.

6.1.2 Controlword and statusword source

The register K375 selects access channel to *ControlWord* (CW) and *StatusWord* (SW) objects depending on set Mode Of Operation display (object 6061h).



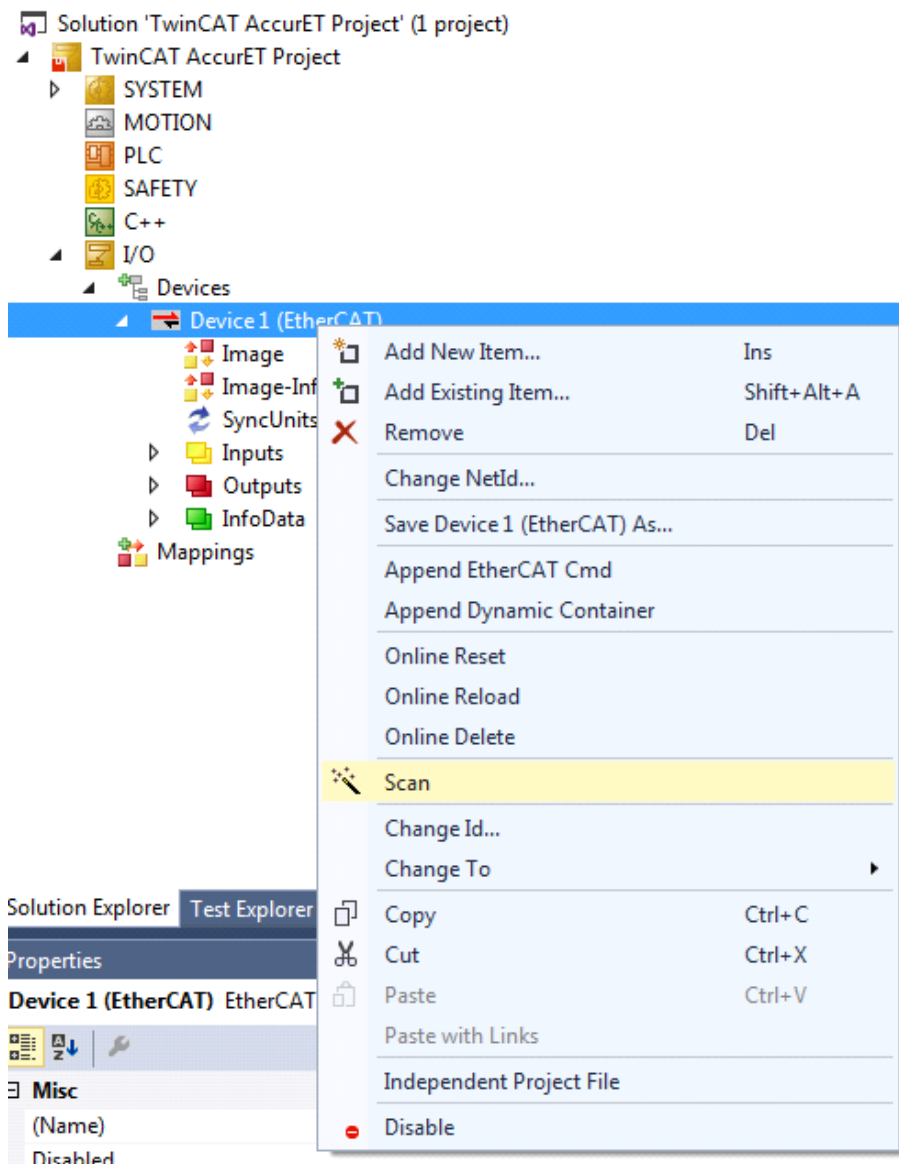
6.1.3 TwinCAT configuration

Once the controller is configured for EtherCAT, it is possible to integrate it in TwinCAT. To do this, the three following ESI (EtherCAT Slave Information) XML files...

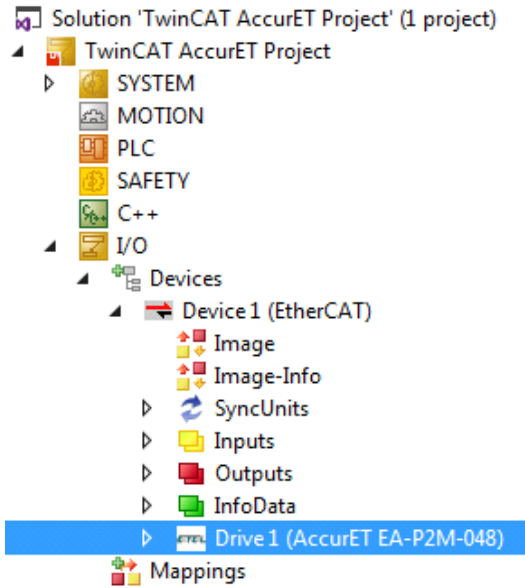
- ETEL_AccurET.xml
 - ETEL_AccurET_Disctionary.xml
 - ETEL_AccurET_Diagnosis.xml
- ... must be copied to the relevant following folder...
- For TwinCAT 3, use path "C:\TwinCAT\3.1\Config\Io\EtherCAT"
 - For TwinCAT 2, use path "C:\TwinCAT\Io\EtherCAT".

After copying the files, TwinCAT is able to scan for AccurET controllers. To do this, first step is to add and configure an EtherCAT Master in your TwinCAT configuration. If unsure please consult the TwinCAT documentation.

After configuring an EtherCAT Master, the user is able to scan for all slaves connected to this Master.



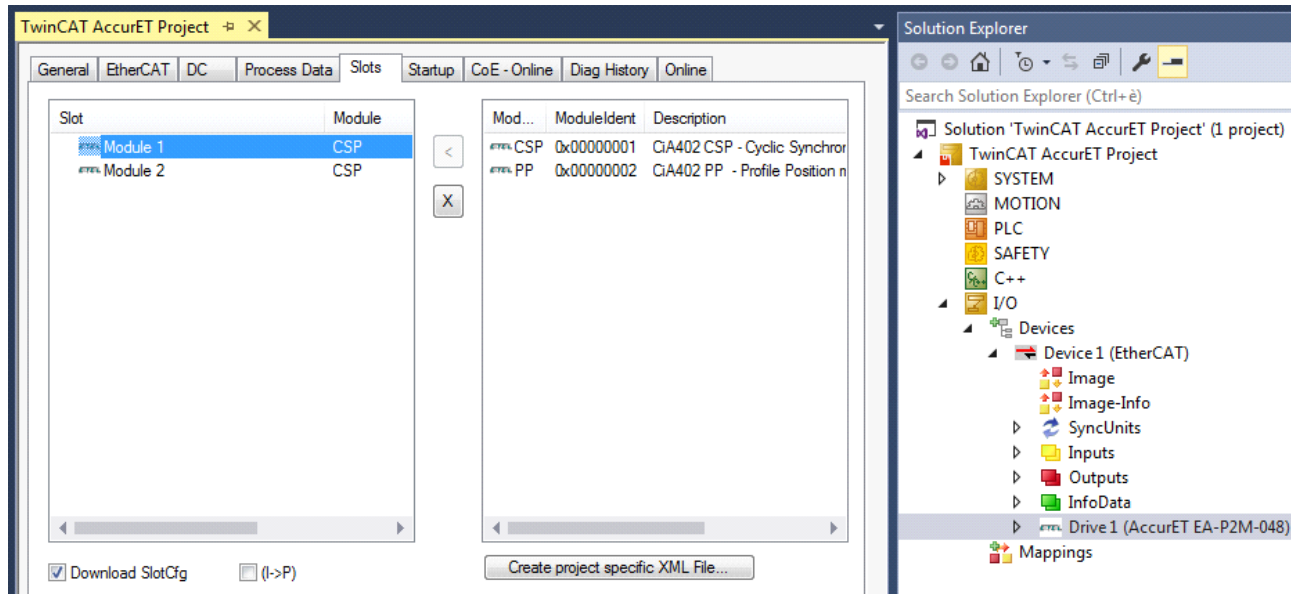
Then, all slaves connected to the Master are visible in TwinCAT configuration as well as AccurET drives.



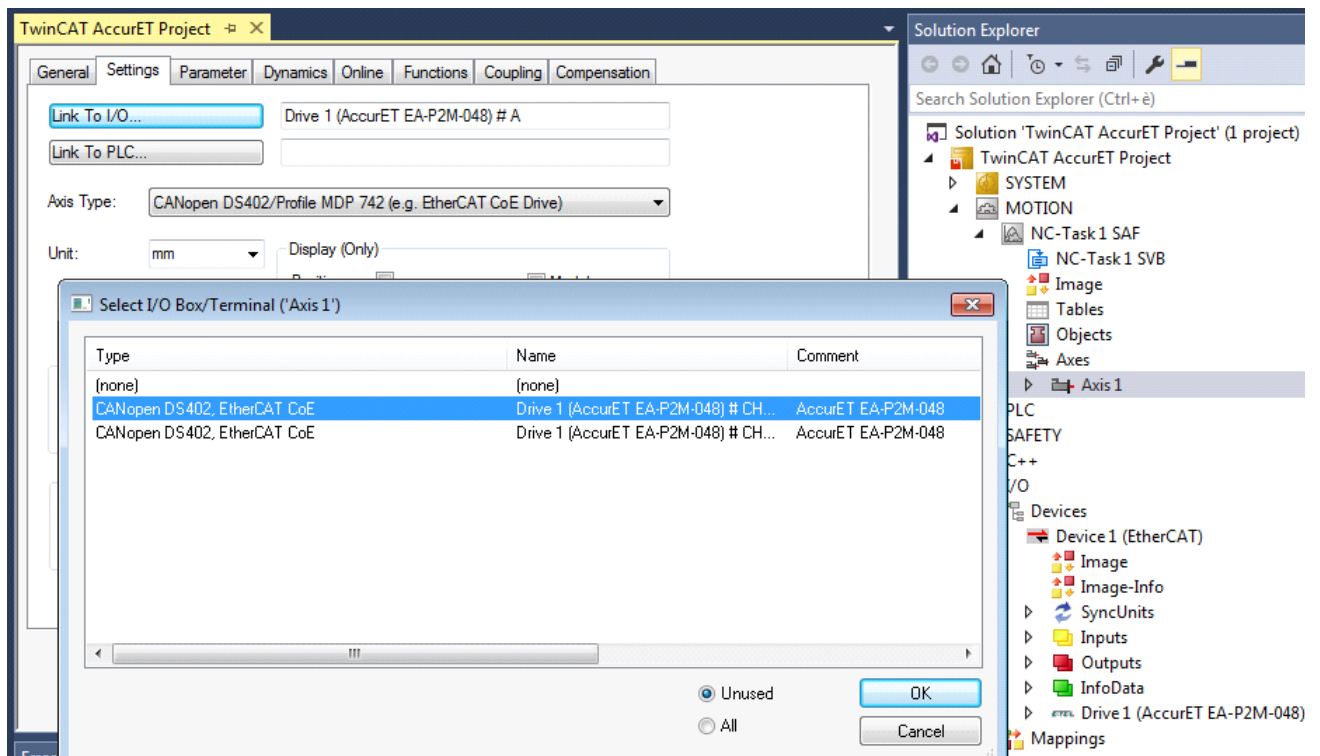
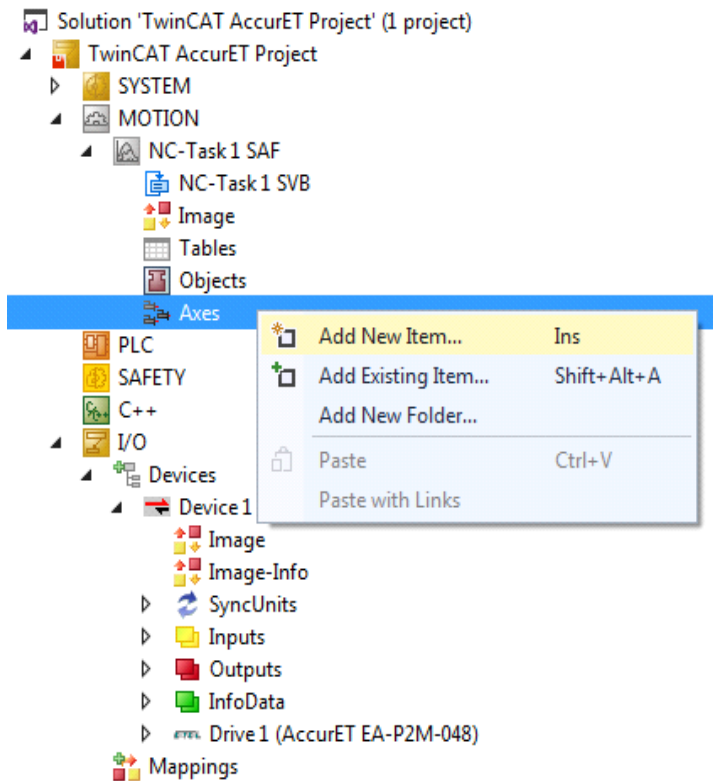
At this point, the master is able to communicate with the controller.

6.1.4 Cyclic synchronous position mode

To configure TwinCAT to work in CSP, select the CSP mode of operation in the drive properties.



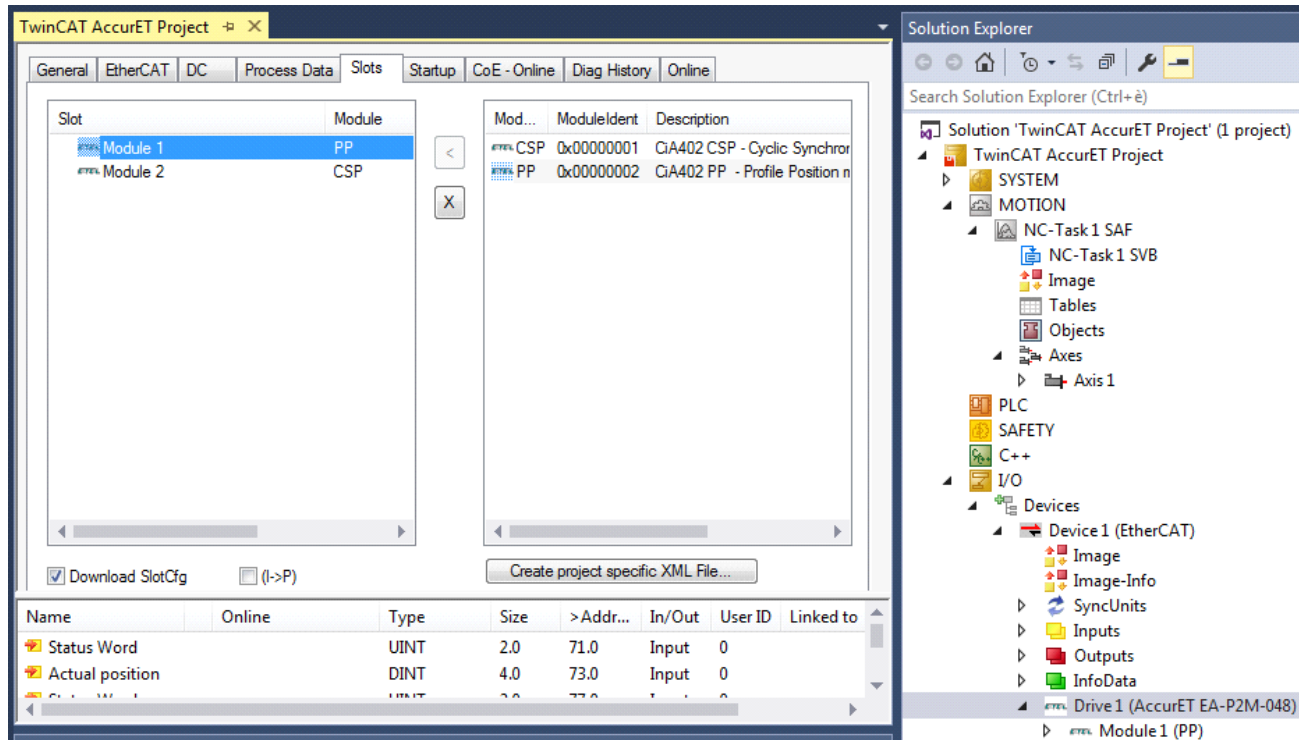
Then, create a new Axis in Motion tree.



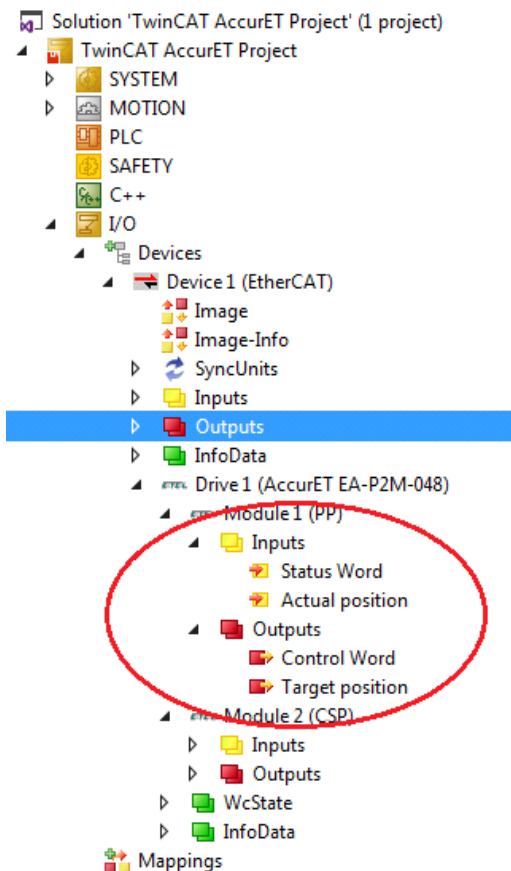
After linking Axis to the right drive, the TwinCAT is able to work with the drive the way Beckhoff recommends. Refer to TwinCAT documentation for more information.

6.1.5 Profile position mode

To configure TwinCAT to work in PP mode, select PP mode in drive properties.



Objects to control the drive are then visible as follows:



Refer to [§4](#) for information about these objects, and refer to TwinCAT documentation for information about the way to access them.

6.1.6 Create custom PDO

To create a custom PDO with TwinCAT, got to tab "Process Data", and select in "PDO List" group box the custom PDO object (16xFh or 1AxFh) to be edited.

Process Data

PDO List:

Flags	Index	Size	Name	Flags	SM	SU
	0x1A07	26.0	Inputs	F		0
	0x1A08	5.0	Inputs	F		0
	0x1B06	12.0	Inputs	F		0
	0x1A0F	6.0	Inputs	F		0
	0x1601	6.0	Outputs	F		0
	0x1602	2.0	Outputs	F		0
	0x1603	4.0	Outputs	F		0
	0x1605	2.0	Outputs	F		0

PDO Content (0x1A0F):

Index	Size	Offs	Name	Type	Default (hex)
0x6041:00	2.0	0.0	Status Word	UINT	
0x6064:00	4.0	2.0	Actual Position	DINT	
		6.0			

Predefined PDO Assignment: (none)

Load PDO info from device

Save Unit Assignment

Select Custom PDO 0x160F or 0x1A0F

Right-click to get the menu, then edit the content.

After selecting "Insert" or "Edit" command, the following dialog box is displayed for selecting object to be added in PDO.

Edit Pdo Entry

Name: Actual Position

Index (hex): 6064 24676

Sub Index: 0

Data Type: DINT

Bit Length: 32

☒ Show only objects from related module

From Dictionary:

6.1.7 RTV configuration

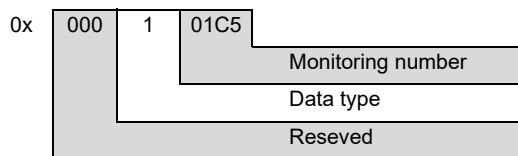
To use RTV equivalent feature, first the user must assign AccurET register with slot as follow:

Transition	Protocol	Index	Data	Comment
<PS>	CoE	0x1C13:01	0x1A0F (6671)	download pdo 0x1C13:01 i...
<PS>	CoE	0x1C13:02	0x1A11 (6673)	download pdo 0x1C13:02 i...
<PS>	CoE	0x1C13:03	0x1A1F (6687)	download pdo 0x1C13:03 i...
<PS>	CoE	0x1C13:00	0x03 (3)	download pdo 0x1C13 count
<PS>	CoE	0xF030:00	0x00 (0)	clear slot cfg 0xF030 entries
<PS>	CoE	0xF030:01	0x00000001 (1)	download slot cfg 0xF030 ...
<PS>	CoE	0xF030:02	0x00000001 (1)	download slot cfg 0xF030 ...
<PS>	CoE	0xF030:00	0x02 (2)	download slot cfg 0xF030 ...
SO	CoE	0x6060:00	8	
PS	CoE	0x6860:00	8	
PS	CoE	0x480B:00	0x03 (3)	SubIndex 000
PS	CoE	0x480B:01	0x01000100 (16777472)	
PS	CoE	0x480B:02	0x00003200 (12800)	
PS	CoE	0x480B:03	0x00003400 (13312)	
PS	CoE	0x480A:00	0x02 (2)	
PS	CoE	0x480A:01	0x000101C5 (65989)	
PS	CoE	0x480A:02	0x000201CA (131530)	

- Rx RTV Assignment**

Sub-index 0 is set to 3 to indicate the number of assignment to do be done.

Sub-index 1 is set to 0x01000100. The decomposition looks like:



This means the register ML453:0 is used. With the same decomposition, we can deduce the second assignment is MF458.

- **Rx RTV Slots**

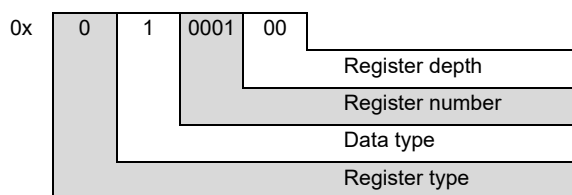
Rx RTV Slots object looks like as follow:

The screenshot shows the 'CoE - Online' tab in the EtherCAT configuration software. The 'Rx RTV Slots' object (480C:0) is expanded, showing sub-indexes 01 to 0E. Red arrows point to the values of sub-indexes 01, 02, and 03, which are labeled as 'Low-Part of ML453', 'High-Part of ML453', and 'MF458' respectively.

Index	Name	Flags	Value
4808:0	Register Access	RO	> 8 <
4809:0	Source Reservation	RO	> 2 <
480A:0	Rx RTV Assign	RW	> 2 <
480B:0	Tx RTV Assign	RW	> 3 <
480C:0	Rx RTV Slots	RO	> 3 <
480C:01	SubIndex 001	RO	0xFFFFFB2E (-1234)
480C:02	SubIndex 002	RO	0xFFFFFFFF (-1)
480C:03	SubIndex 003	RO	0x1A9FBE77 (446676599)
480C:04	SubIndex 004	RO	---
480C:05	SubIndex 005	RO	---
480C:06	SubIndex 006	RO	---
480C:07	SubIndex 007	RO	---
480C:08	SubIndex 008	RO	---
480C:09	SubIndex 009	RO	---
480C:0A	SubIndex 010	RO	---
480C:0B	SubIndex 011	RO	---
480C:0C	SubIndex 012	RO	---
480C:0D	SubIndex 013	RO	---
480C:0E	SubIndex 014	RO	---

- **Tx RTV Assignment:**

Sub-index 0 is set to 3 in order to indicate the number of assignment to do be done.
Sub-index 1 is set to 0x01000100. The decomposition looks like:



This means the register ML1 is used. With the same decomposition, we can deduce the second assignment is M50 and the third one is M52.

- Tx RTV Slots

Tx RTV Slots object looks like as follow:

General EtherCAT DC Process Data Plc Slots Startup **CoE - Online** Diag History

Update List ☐ Auto Update ☒ Single Update ☐ Show Offline Data

Advanced... All Objects

Add to Startup... Online Data Module OD (AoE Port): 0

Index	Name	Flags	Value
480C:0	Rx RTV Slots	RO	> 3 <
480D:0	Tx RTV Slots	RO	> 4 <
480D:01	SubIndex 001	RO	0x00000400 (1024)
480D:02	SubIndex 002	RO	0x00000000 (0)
480D:03	SubIndex 003	RO	0x00000000 (0)
480D:04	SubIndex 004	RO	0x00000007 (7)
480D:05	SubIndex 005	RO	---
480D:06	SubIndex 006	RO	---
480D:07	SubIndex 007	RO	---
480D:08	SubIndex 008	RO	---
480D:09	SubIndex 009	RO	---
480D:0A	SubIndex 010	RO	---
480D:0B	SubIndex 011	RO	---
480D:0C	SubIndex 012	RO	---
480D:0D	SubIndex 013	RO	---
480D:0E	SubIndex 014	RO	---
480D:0F	SubIndex 015	RO	---
480D:10	SubIndex 016	RO	---
603F	Error Code	RO	0x0000 (0)

Low-Part ML1

High-Part ML1

M50

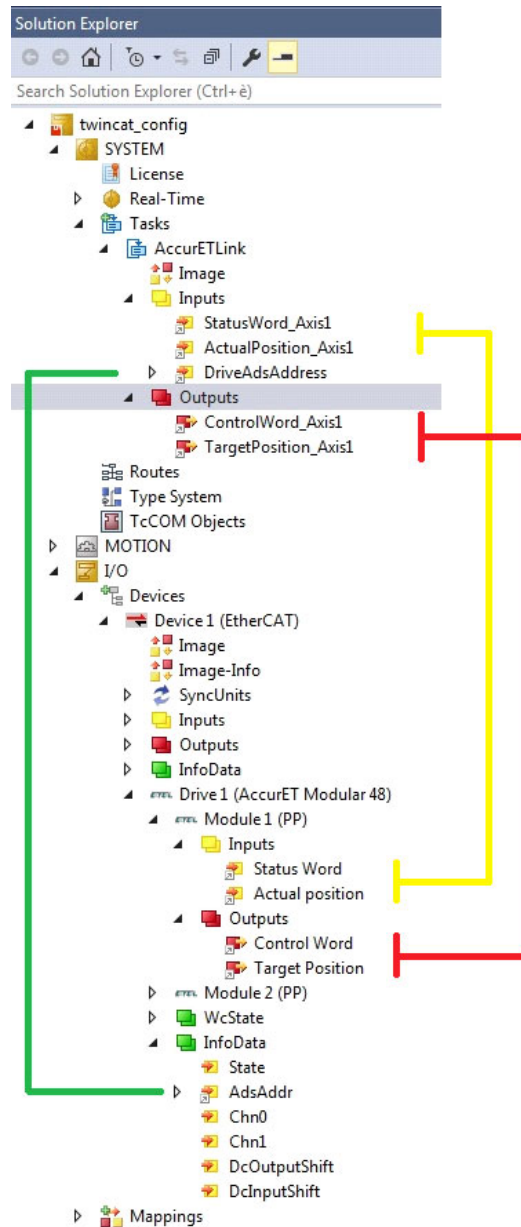
M52

6.1.8 SDO / PDO access with TwinCAT through a C application

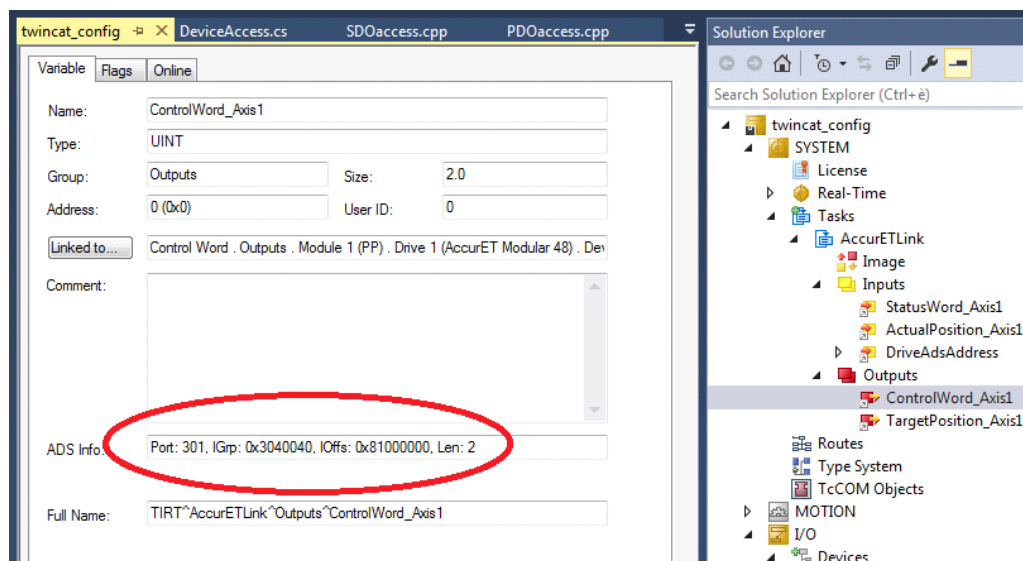
This section describes how a developer can deal with SDO and PDO objects with a C application. These samples assume the controller is working in *Profile Position* mode. The described method can be applied in *Cycle Synchronous Position* mode as well.

The first step is to create a new task in SYSTEM / Tasks. In our sample, this task is named AccurETLink. In the task, create necessary variables to work with PDO objects. In the figure, refer to yellow and red lines.

To work with SDO, another variable (DriveAdsAddress) is added in AccurETLink task to provide to the C application the ADS address of the controller the application wants to access. Refer to the green line in the figure.



To access AccurETLink variables from a C application the user must retrieve ADS information of each variable as follows:



ADS info provides all information required to communicate with the task.

6.1.8.1 Open / close ADS communication

To open and close ADS communication with TwinCAT, use the following sample source code:

```

long nErr, nPort;
AmsAddr addr;

nPort = AdsPortOpen();
nErr = AdsGetLocalAddress(&addr);
ADS_VERIFY(nErr);
// ADS Port opened.

...    // Client application running

// Closing ADS communication
nErr = AdsPortClose();
ADS_VERIFY(nErr);

```

6.1.8.2 Accessing AccurETLink task variable

To read a variable, use the following sample:

```

WORD GetStatusWordAxis1(AmsAddr* pAddr)
{
    //Port: 301, IGrp : 0x3040040, IOffs : 0x80000000, Len : 2
    WORD status;
    pAddr->port = 301;
    long code = AdsSyncReadReq(pAddr, 0x3040040, 0x80000000, sizeof(WORD),
                               &status);

    ADS_VERIFY(code);
    return status;
}

```

To write a variable, use the following sample:

```

void SetControlWordAxis1(AmsAddr* pAddr, WORD value)
{
    // Port: 301, IGrp: 0x3040040, IOffs: 0x81000000, Len: 2
    pAddr->port = 301;
    long code = AdsSyncWriteReq(pAddr, 0x3040040, 0x81000000, sizeof(WORD),
                                &value);

    ADS_VERIFY(code);
}

```

Remainder: If the user change the mode of operation (object: 6060h) for homing for example, the PDO is 1600h or 1A00h is not accessible anymore. To access objects in these PDO, the user must use SDO access procedure.

6.1.8.3 Accessing SDO object

To access an object on a controller, use the following sample source code:

The function `GetDeviceAdsAddress` retrieves the ADS of the controller which has been linked as AccurETLink task variable.


```
AmsAddr GetDeviceAdsAddress(AmsAddr* pAddr)
{
    AmsAddr addr;

    unsigned long cbReturn;

    pAddr->port = 301;

    // Read AccurETLink variable DriveAdsAddress
    long code = AdsSyncReadReqEx(pAddr, 0x3040040, 0x80000006,
                                sizeof(AmsAddr), &addr, &cbReturn);
    ADS_VERIFY(code);
    if (cbReturn != sizeof(AmsAddr))
    {
        printf("Bad reading - Some characters are missed\n");
        system("pause");
        exit(0);
    }

    return addr;
}
```

GetSDO and SetSDO functions read or write data into a specific object.

```
void GetSDO(AmsAddr* pAddr, BYTE ModuleId, WORD index, WORD subindex,
            unsigned long dataLen, void* data)
{
    AmsAddr devAddr = GetDeviceAdsAddress(pAddr);

    DWORD offset = index + ModuleId * 0x800;
    offset <= 16;
    offset |= subindex;

    long code = AdsSyncReadReq(&devAddr, 0xF302, offset, dataLen, data);
    ADS_VERIFY(code);
}

void SetSDO(AmsAddr* pAddr, BYTE ModuleId, WORD index, WORD subindex,
            unsigned long dataLen, void* data)
{
    AmsAddr devAddr = GetDeviceAdsAddress(pAddr);

    DWORD offset = index + ModuleId * 0x800;
    offset <= 16;
    offset |= subindex;

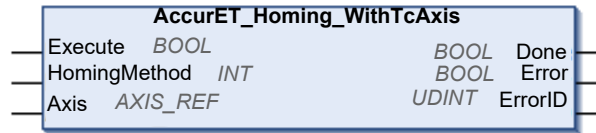
    long code = AdsSyncWriteReq(&devAddr, 0xF302, offset, dataLen, data);
    ADS_VERIFY(code);
}
```

For more information about working with ADS protocol, refer to TwinCAT documentation.

6.2 ETEL library

ETEL provides a PLC library (TcPLCEtel) to speed up working with EtherCAT AccurET especially in homing procedures. Of course, working with TwinCAT homing is still valid. This library aims to extend homing procedures to all those available in AccurET firmware.

The most important object in this library is the object block **AccurET_Homing_WithTcAxis**:



IN variables

Execute Start the procedure.
HomingMethod Choose the desired homing method. Refer to [§4.6.2.4](#).
Axis Select axis to be homed.

OUT variables

Done Indicates when the procedure is finished, with or without error.
Error Indicates if an error occurred during the homing procedure.
ErrorID Indicates an error code in case of error.

6.3 Errors reference list

The points to check in the **errors reference list** are indicated in the table below.

Remark: Only errors relevant to EtherCAT are listed below, for a complete list refer to the '**AccurET Operation & Software Manual**' (section Errors reference list).

M64	Displayed message	Description	Solutions	HW res	Brk	Protection type	Product
160	ETHERCAT FAULT	EtherCAT state machine in fault reaction	Check diagnosis message history		ON	1	All
161	ETHERCAT FAULT	ETCOM management fault	Internal error		ON	1	All
162	ETHERCAT FAULT	Record management fault	Internal error		ON	1	All
163	ETHERCAT FAULT	Invalid transition required in EtherCAT state machine	The master asks for an unauthorized transition. Update the master or modify it.		ON	1	All
164	ETHERCAT FAULT	Invalid EtherCAT configured station alias	Check DIP-Switch and Configuration Station Alias in EEPROM, at least one must be set to 0		ON	1	All
165	ETHERCAT FAULT	RTV bad configuration	Check RTV configuration (especially objects 400Ah and 400Bh)		ON	1	All

6.4 Units conversion

For more information, refer to the '**AccurET Operation & Software Manual**' (section Units conversion).

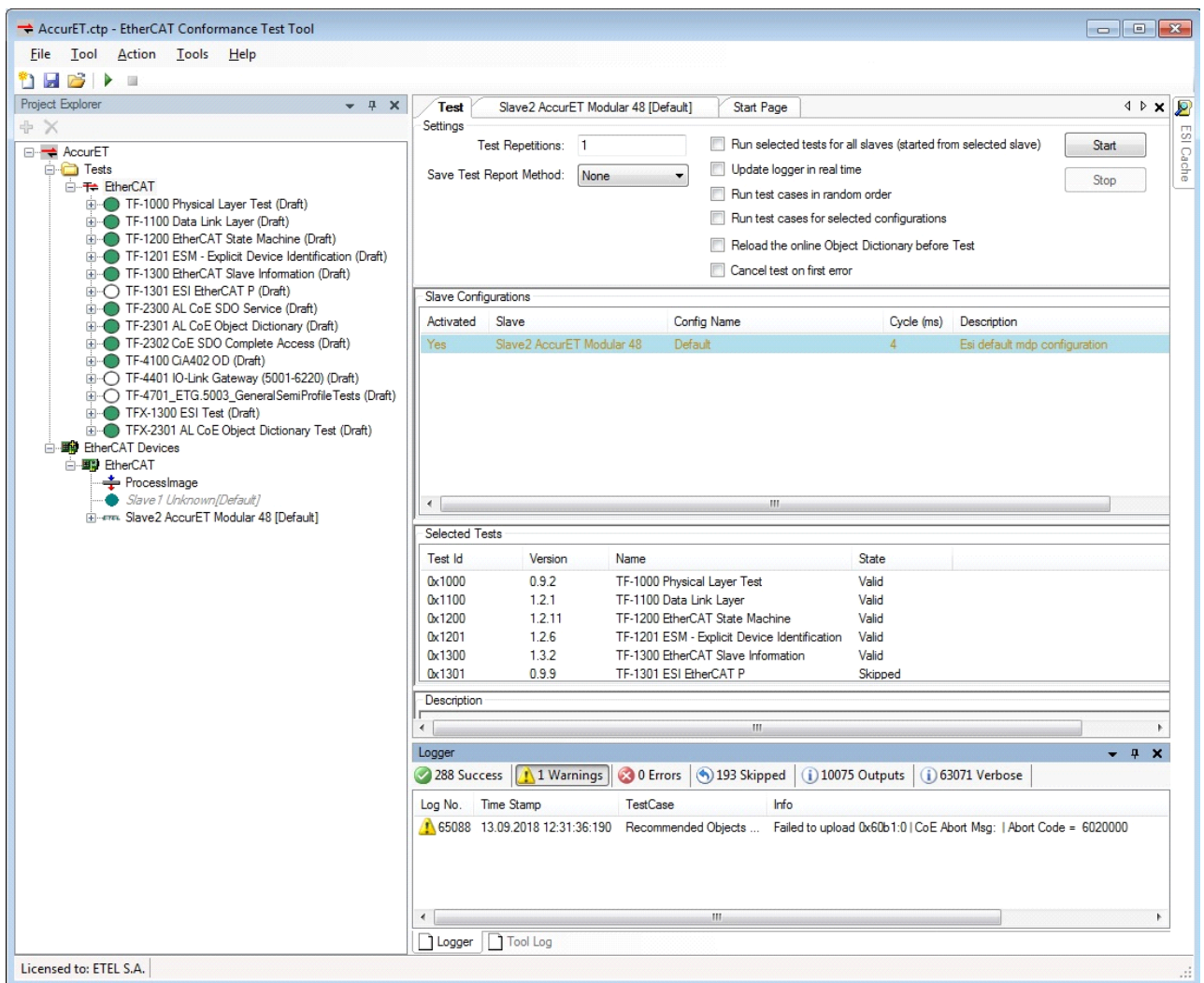
6.5 EtherCAT certification

AccurET is certified as an EtherCAT device.



To get certified as EtherCAT device, it is required that the AccurET passes with success the *EtherCAT Conformance Test Tool* provided by EtherCAT ETG.

AccurET passes this test with one remaining warning (see the following figure) due to the object 60B1h missing in AccurET. This object represents a feed-forward velocity offset for the complete control loop of the controller. Since AccurET has two cascaded control loops, one for position loop and one for current loop, but none for velocity, the object 60B1h has no meaning, and therefore was not implemented.



7 Service and support

For any inquiry regarding technical, commercial and service information relating to ETEL S.A. products, please contact your ETEL S.A. representative:

HEADQUARTER / SWITZERLAND	BELGIUM	CHINA
ETEL S.A. Zone industrielle CH-2112 Môtiers Phone: +41 (0)32 862 01 00 E-mail: etel@etel.ch http://www.etel.ch	HEIDENHAIN nv/sa Pamelse Klei 47 1760 Roosdaal Phone: +32 54 34 31 58 E-mail: sales@heidenhain.be	DR. JOHANNES HEIDENHAIN (CHINA) Co., Ltd No. 6, Tian Wei San Jie, Area A, Beijing Tianzhu Airport, Industrial Zone Shunyi District, Beijing 101312 Phone: +86 400 619 6060 E-mail: sales@heidenhain.com.cn
CZECH Republic	FRANCE	GERMANY
HEIDENHAIN s.r.o. Dolnomecholupská 12b 102 00 Praha 10 - Hostivar Phone: +420 272 658 131 E-mail: heidenhain@heidenhain.cz	HEIDENHAIN FRANCE SARL 2 avenue de la cristallerie 92310 Sèvres Phone: +33 (0)1 41 14 30 09 E-mail: sales@heidenhain.fr	DR. JOHANNES HEIDENHAIN GmbH Technisches Büro Südwest II Verkauf ETEL S.A. Schillgasse 14 78661 Dietingen Phone: +49 (0)741 17453-0 E-mail: tbsw.etel@heidenhain.de
GREAT-BRITAIN	ISRAEL (Representative)	ITALY
HEIDENHAIN (GB) Ltd. 200 London Road, Burgess Hill, West Sussex RH 15 9RD Phone: +44 (0)1444 247711 E-mail: sales@heidenhain.co.uk	MEDITAL COMOTECH Ltd. 36 Shacham St., P.O.B 7772, Petach Tikva Israel 4951729 Phone: +972 3 923 3323 E-mail: comotech@medital.co.il	ETEL S.A. Piazza della Repubblica 11 28050 Pombia Phone: +39 0321 958 965 E-mail: etel@etelsa.it
JAPAN	KOREA	SINGAPORE
HEIDENHAIN K.K. Hulic Kojimachi Bldg. 9F 3-2 Kojimachi, Chiyoda-ku Tokyo - 102-0083 Phone: +81 3 3234 7781 E-mail: sales@heidenhain.co.jp	HEIDENHAIN KOREA Ltd. 75, Jeonpa-ro 24beon-gil, Manan-gu, Anyang-si Gyeonggi-do, 14087, Korea Phone: + 82 31-380-5304 E-mail: etelsales@heidenhain.co.kr	HEIDENHAIN PACIFIC PTE. LTD 51 Ubi Crescent, Singapore 408593 Phone: +65 6749 3238 E-mail: info@heidenhain.com.sg
SPAIN (Representative)	SWEDEN	SWITZERLAND
Farresa Electronica, S.A. C/ Les Corts, 36 bajos ES-08028 Barcelona Phone: +34 93 409 24 91 E-mail: farresa@farresa.es	HEIDENHAIN Scandinavia AB Storsåtragränd 5 127 39 Skärholmen Phone: +468 531 93 350 E-mail: sales@heidenhain.se	HEIDENHAIN (SCHWEIZ) AG Vieristrasse 14 CH-8603 Schwerzenbach Phone: +41 (0)44 806 27 27 E-mail: verkauf@heidenhain.ch
TAIWAN	THE NETHERLANDS	UNITED STATES
HEIDENHAIN CO., LTD. No. 29, 33rd road, Taichung Industrial Park Taichung 40768, Taiwan, R.O.C. Phone: +886 4 2358 8977 E-mail: info@heidenhain.tw	HEIDENHAIN NEDERLAND B.V. Copernicuslaan 34 6716 BM Ede Phone: +31 (0)318 581800 E-mail: verkoop@heidenhain.nl	HEIDENHAIN CORPORATION 333 E. State Parkway Schaumburg, IL 60173 Phone: +1 847 490 1191 E-mail: info@heidenhain.com

The technical hotline, based in ETEL S.A.'s headquarters, can be reached by:

- Phone: +41 (0)32 862 01 12.
- Fax: +41 (0)32 862 01 01.
- E-mail: support@etel.ch.

Please refer to your corresponding ETEL S.A. representative for more information about the technical documentation. ETEL S.A. organizes training courses for customers on request, including theoretical presentations of our products and practical demonstrations at our facilities.