



AccurET Modular Position Controllers

Operation & Software Manual

Version Q

ETEL

THIS PAGE IS INTENTIONALLY LEFT BLANK

Table of contents

Chapter A: Internal functioning & generalities

1 Controller principle	18
1.1 AccurET modular controllers	18
1.2 AccurET VHP controllers	18
2 Commands & registers syntax	20
2.1 Commands	20
2.1.1 Sending commands	20
2.2 Registers	21
2.2.1 Registers group	21
2.2.2 Register value reading	22
2.2.3 Register value writing	23
2.3 Bits field or numerical values for registers and commands	24
2.3.1 Bits field	24
2.3.2 Numerical values	25
2.4 Communication with the controller	25
2.4.1 Communication protocols	25
2.4.2 Single-axis configuration	27
2.4.3 Multi-axis configuration	27
2.5 Commands manager	28
2.5.1 Urgent commands	29
2.5.2 Monitoring commands	29
2.5.3 Normal commands	29
2.5.4 TransnET communication	30
2.5.5 TCP/IP and USB communication	30
2.5.6 Compiled sequences	30

Chapter B: System setup and tuning

3 Initial system installation	32
3.1 Controller connection	32
3.1.1 Single-axis configuration	32
3.1.2 Multi-axis configuration	32
3.2 ComET software installation	33
3.2.1 Establishment of the communication	33
3.2.2 Firmware download	34
4 Simplified regulator's principle	36
4.1 General diagram	36

4.2 Parameters description	37
4.2.1 Current regulator	37
4.2.2 Position regulator	38
5 Controller regulators tuning principle	40
 Chapter C: System functions	
6 Saving of the settings	44
7 Basic functions and settings	47
7.1 Axes number selection	47
7.1.1 Selection with DIP switch	47
7.1.2 Selection with command AXI	47
7.2 Motor selection	48
7.3 Position encoder selection	49
7.3.1 1 Vpp analog encoders (K79=0)	51
7.3.2 TTL encoders (K79=2)	53
7.3.3 EnDat 2.2 encoders without sine and cosine signals (K79=3)	54
7.3.4 EnDat 2.1 encoders or EnDat 2.2 with sine and cosine signals (K79=4)	54
7.3.5 1 Vpp HSEI analog encoders (K79=5)	55
7.3.6 Absolute position from any register (K79=20)	57
7.3.7 Dual Encoder Feedback	57
7.3.8 No encoder and no possibility to make a power on (K79=98)	65
7.3.9 No encoder (K79=99)	65
7.3.10 Position factors for EDI package	65
7.3.11 Encoder monitorings	66
7.4 Protection parameters - IMPORTANT	66
7.4.1 Movements limits	66
7.4.2 Current limits	68
7.4.3 General protection	71
7.5 Errors and warnings handling	74
7.5.1 Errors reset: RST and RSD	74
7.5.2 Errors propagation on the TransnET	75
7.5.3 Dynamic braking	76
7.5.4 Warning management	78
7.6 Normal reference mode (K61=1)	79
7.7 Initialization	79
7.7.1 Phasing basics	80
7.7.2 Phasing processes (parameter K90)	82
7.8 Autosetting	88
7.8.1 AUT command	88

7.8.2 PWR command	90
7.9 Homing	91
7.9.1 Homing purpose	91
7.9.2 Homing modes	94
7.10 Wait commands	105
7.10.1 WTT command	105
7.10.2 WTM and WTP commands	105
7.10.3 Wait for Window: WTW command	106
7.10.4 WTS command	107
7.10.5 Wait on bits: WBS and WBC commands	108
7.10.6 Wait on values: WPL, WSL, WPG and WSG commands	109
7.10.7 Clear wait commands	110
7.11 Emergency stop: HLT, HLB, HLO	110
7.12 Basic movements	111
7.12.1 Zero machine	111
7.12.2 Linear or rotary movement	112
7.12.3 Trajectory parameters and start movement	112
7.12.4 Rotary S-curve movement	115
7.13 Controller software characteristics	117
7.14 Stepper in open loop	118
7.15 Power Sag Detection	121
8 Advanced functions (only for advanced users)	123
8.1 Regulators in details - advanced tuning	123
8.1.1 Regulators diagram	123
8.1.2 Regulators parameters	124
8.1.3 Monitorings diagram	130
8.1.4 Cogging compensation	132
8.2 Advanced reference modes (K61≠1 or K63=1)	133
8.2.1 External reference modes (K61=3, 4 or 36 and K63=1)	133
8.2.2 Master/slave position compensation	138
8.2.3 Compensation on real position	139
8.3 Advanced movements	140
8.3.1 Movements types	140
8.3.2 Look-up table movements	141
8.3.3 Infinite rotary movement	144
8.3.4 Movement with predefined profile	147
8.3.5 Start movements: STA, STI and SMP commands	149
8.3.6 Concatenated movements: MMC command	152
8.3.7 CAM command	153
8.3.8 STE command	154
8.3.9 BRK and STP commands	155

8.4 Trajectory filter	156
8.4.1 Setting phases	156
8.5 Digital inputs / outputs	157
8.5.1 Digital inputs of the controller	157
8.5.2 Digital inputs of the optional board	158
8.5.3 Digital outputs of the controller	159
8.5.4 Digital outputs of the optional board	160
8.5.5 Digital outputs triggered by RTV	161
8.6 Analog inputs / outputs	161
8.6.1 On AccurET modular controllers	161
8.6.2 On AccurET VHP controllers	166
8.7 Position capture on digital inputs	168
8.7.1 Description of the position capture process on the digital inputs	169
8.8 In-window	170
8.9 Fast Triggers	171
8.9.1 Overview	171
8.9.2 General parameters	172
8.9.3 1D Fast Triggers	181
8.9.4 2D Fast Triggers	183
8.9.5 Fast Triggers status	188
8.9.6 Additional features	189
8.9.7 Error and warning	195
8.9.8 Limitation	196
8.9.9 How to	197
8.10 Controller status	229
8.10.1 IsMoving bit	229
8.11 Encoder Scale Mapping	230
8.11.1 Setting	230
8.11.2 Activation of the corrections	231
8.11.3 Use	232
8.12 Encoder Stretch	232
8.13 Set / reset bit(s) management	234
8.14 Set several registers	234
8.15 Serial Peripheral Interface bus (SPI)	235
8.15.1 SPI interface configuration	236
8.15.2 Message configuration	238
8.15.3 CRC computation	239
8.15.4 Monitoring	240
8.15.5 Use of SPI communication	241
8.16 Traces management	242

8.16.1 Traces configuration	242
8.16.2 Non-synchronized mode	245
8.16.3 Synchronized mode	245
8.16.4 Trigger condition test and edge detection	247
8.16.5 Traces upload	247
8.16.6 Continuous traces	247
8.17 External motor braking control	248
8.17.1 DOUT selection	248
8.17.2 Brake timing	248
8.17.3 Motor braking state	249
8.17.4 Brake management during the phasing process	249
9 Stage protection	250
9.1 Terminology	250
9.1.1 Type of braking	250
9.1.2 Error types	251
9.2 Restriction	251
9.3 Principle diagram	252
9.4 Registers	253
9.4.1 Stage protection setting	255
9.4.2 Error reset	255
9.5 Error cases	255
9.5.1 Example with speed braking due to an error type 1	255
9.5.2 Example with speed braking and dynamic braking due to an error type 1	256
9.5.3 Example with speed braking and dynamic braking due to an error type 1 and 2	257
9.5.4 Example with speed braking and dynamic braking due to an error type 1, 2 and 3	258
9.6 Error management in protection mode	259
10 Real-time channels on TransnET	261
10.1 Principle	261
10.2 Data sharing	261
10.2.1 Between controllers	261
10.2.2 Between controllers and the PC application (UltimET Light PCI only)	261
10.2.3 Disappearing of TransnET	262
10.3 Interface description	262
10.3.1 Slot management	263
10.3.2 UltimET – AccurET functions for sharing data	263
10.3.3 Real-time channels timing	266
11 Stage error mapping	267
11.1 Introduction	267
11.2 Applied correction	267
11.3 Stage Mapping configuration	268

11.3.1 Schematics of different configurations	268
11.4 Activation / deactivation of the stage error mapping	270
11.4.1 Introduction	270
11.4.2 Full activation / deactivation	272
11.4.3 Activation / deactivation correction	273
11.4.4 Mapping area	273
11.4.5 Error management	274
11.5 Activation example	274
12 Gantry control	277
12.1 Introduction	277
12.2 Gantry levels	277
12.2.1 Visibility aspect	277
12.2.2 Level configuration parameters	278
12.3 Gantry axes configuration	278
12.4 Gantry protections and errors	279
12.4.1 Error management between the axes on the gantry	279
12.4.2 Tracking error management between the axes on the gantry	279
12.4.3 'Power off' protection management between the axes on the gantry	279
12.4.4 Parameter check	279
12.5 Management in gantry level 2	279
12.5.1 Commands management	279
12.5.2 Parameters management	280
12.5.3 Monitorings management	280
12.5.4 Sequence management in gantry level 2	281
12.6 Homing mode in gantry level	281
12.6.1 Homing in gantry level 1	281
12.6.2 Homing in gantry level 2	282
12.7 How to make the setting of a gantry	282
12.7.1 Setting of axes X1 and X2	282
12.7.2 Setting of the beam (axis Y)	282
12.8 Command for slot assignment (if advanced gantry feedforward needed)	282
12.9 Advanced gantry feedforward	283
12.9.1 Management of the status beam axis (Y) on gantry side	284
12.9.2 Setting of the advanced gantry feedforward	284
13 Force control mode	285
13.1 Single axis force control	285
13.1.1 Introduction	285
13.1.2 Contact force definition	285
13.1.3 Commands	285
13.1.4 Registers	290

13.1.5 Touchdown detection	292
13.1.6 Restrictions	293
13.1.7 Hardware protection	293
13.1.8 Examples	293
14 ZxT combined module functions	295
14.1 Operation principle	295
14.1.1 Z axis	296
14.1.2 Tip-tilt (Rx, Ry)	296
14.1.3 Transfer matrices	296
14.2 Tip-tilt	296
14.2.1 Trajectory generator	297
14.2.2 Stage feedforward	297
14.2.3 Stage Mapping	297
14.2.4 Software limits	297
14.3 Homing	299
14.3.1 Calibration procedure	299
14.3.2 Homing procedure	300
14.4 Commands	301
14.4.1 MVETILT and RMVETILT	301
14.4.2 SETTILT	301
14.4.3 STETILT_ADD, STETILT_SUB and STETILT_ABS	302
14.4.4 IND	302
14.4.5 SLS	302
14.4.6 WTW	302
14.4.7 WTWZXT	302
14.4.8 SAFRX	303
14.4.9 SAFRY	303
14.5 Parameters	304
14.6 Monitorings	305
14.7 Errors	305
14.8 Unit conversion	306
14.8.1 Tip-tilt positions (dpi) for Rx and Ry	306
14.8.2 Current (cur/curPhysical)	306

Chapter D: Programming

15 Programming of the controller	308
15.1 Introduction	308
15.2 Basic concepts	309
15.2.1 Source file	309
15.2.2 Comments	309

15.2.3 Header	309
15.2.4 Register and command	309
15.2.5 Instruction delimiter	309
15.2.6 Immediate values	309
15.2.7 C preprocessor define	310
15.2.8 Thread	310
15.2.9 User variables X	311
15.3 Structured code	311
15.3.1 Identifiers	311
15.3.2 Types	312
15.3.3 Global variables	312
15.3.4 Registers	313
15.3.5 Functions	313
15.3.6 Operators	315
15.3.7 Expressions	318
15.3.8 Commands behavior	320
15.3.9 Statements	320
15.3.10 Units conversion	324
15.3.11 Predefined functions	325
15.4 How to start a sequence	327
15.4.1 Start a sequence thread from ComET	327
15.4.2 Start a sequence thread from an application running on the PC	327
15.4.3 Start a sequence on the AccurET from a sequence present in the UltimET	327
15.4.4 Start a sequence thread from another thread	327
15.4.5 Start a sequence thread automatically	328
15.5 How to stop a sequence thread	328
15.6 Error management	328
15.6.1 Simplified error routine	329
15.6.2 ERR command	330
15.7 Performance aspects	330
15.7.1 Allocation of time to the compiled sequence	330
15.7.2 Compiled sequence stack management	330

Chapter E: Appendixes

16 Commands reference list	334
17 Parameters K	350
18 Common parameters C	368
19 Monitorings M	372
20 Warnings reference list	385

21 Errors reference list	387
22 Units conversion	400
22.1 Cinematic quantities units	400
22.1.1 Linear motors	401
22.1.2 Rotary motors	403
22.1.3 Resolution	404
22.2 Current units	405
22.3 Time quantities unit	406
22.4 Volt vs. increment	406
23 Service and support	407

Chapter 1: Index

THIS PAGE IS INTENTIONALLY LEFT BLANK

Record of revisions:

Document revisions			
Version	Date	Main modifications	
Ver A	15.05.09	First version	
Ver B	26.08.09	Updated version: - EnDat 2.2 compatibility - AccurET Modular 48 added	- New alias implemented
Ver C	03.01.11	Updated version: - Compatibility break (refer to page 14) - New regulator diagram (refer to §8.1.1) - Friction feedforward (refer to §8.1.2.3) - New monitoring diagram (refer to §8.1.3) - Cogging compensation (refer to §8.1.4) - Look-up table movement (refer to §8.3.2)	- New encoder Scale Mapping (refer to §8.11) - Stage protection (refer to §9.) - Stage error mapping (refer to §11.) - Real-time channels on TransnET (refer to §10.) - Gantry control (refer to §12.)
Ver D	22.02.12	Updated version: - Movement with predefined profile (refer to §8.3.4) - Trigger initialization and activation - Encoder Stretch (refer to §8.12)	- WTW and WTS commands (refer to §7.10.3 and §7.10.4) - Homing mode in gantry level (refer to §12.6) - Predefined function (refer to §15.3.11)
Ver E	12.11.12	Updated version (with firmware from version 2.06A): - Warning management (refer to §8.13)	- External motor braking control through DOUT (refer to §8.17)
Ver F	17.04.13	Updated version (with firmware from version 2.07A): - Commands synchronization parameter C19 (refer to §2.5.4) - Firmware download procedure updated (refer to §3.2.2)	- Advanced Gain Scheduling (refer to §8.1.2.2) - Mask for active thread parameter K297 (refer to §15.1) - ERR command updated (refer to §15.6.2)
Ver G	09.05.14	Updated version (with firmware from version 3A): - New commands: CLRWAIT (refer to §7.10.7), CH_BIT_REG32 (refer to §8.13) and SAF (refer to §8.1.2.4)	- Force control function (refer to §13.) - Trigonometric float functions (refer to §15.3.11.6) - New parameter K144 (refer to §15.5)
Ver H	17.12.14	Updated version (with firmware from version 3.01A): - Dual Encoder Feedback function (refer to §7.3.7) - Position factors for EDI package (refer to §7.3.10) - Safety signals on DOUT / FDOUT (refer to §7.4.3.3)	- K32 parameters updated (refer to §7.9.1.1) - IsMoving bit (refer to §8.10.1) - Classic code removed because not used any more.
Ver I	12.05.15	Updated version (with firmware from version 3.02A): - New command: SSR (refer to §8.14)	- Master / slave mode (refer to §8.2.1 and §8.2.2)
Ver J	14.07.16	Updated version (with firmware from version 3.10A): - New command: TRS (refer to §8.9.5.2) - 2D trigger function (refer to §8.9.4)	- Force control improved (refer to §13.)
Ver K	19.12.16	Updated version (with firmware from version 3.12A): - AccurET VHP controller added - Absolute position from any register (refer to §7.3.6)	- Trigger improvement - ZxT module function (refer to §14.)
Ver L	27.11.17	Updated version (with firmware from version 3.14A): - Dual Encoder Feedback configuration (refer to §7.3) - Trigger chapter revised (refer to §8.9) - New parameter (K460) for RTV (refer to §10.3)	- New dual axis force control feature - Unit conversion for ZxT updated (refer to §14.8)
Ver M	29.11.18	Updated version (with firmware from version 3.16A): - Dual Encoder Feedback improved (refer to §7.3.7) - Zero machine position improved /refer to §7.12.1) - Compensation on real position (refer to §8.2.3) - Position capture on digital inputs improved (refer to §8.7)	- Brake management during the phasing process (refer to §8.17.4) - Force control function improved (refer to §13.) - ZxT combined module functions improved (refer to §14.) - Performance aspects (refer to §15.7)
Ver N	12.06.19	Updated version (with firmware from version 3.17A): - New registers: MF112 (refer to §7.3.7), MF95 (refer to §7.7.2.4), KF318 & KF319 (refer to §13.1.4)	- Touchdown detection (refer to §13.1.5)
Ver O	11.12.19	Updated version (with firmware from version 3.18A): - New registers: M163 (refer to §7.12.3), M87, M88 and M89 (refer to §7.13), K249 (refer to §8.1.2.3), ML3 (refer to §8.3.9), M443 and M447 (refer to §10.3.2), ML396 (refer to §13.1.7)	- Digital outputs over RTV (refer to §8.5.5) - Trigger updated (refer to §8.9.2.3.2 , §8.9.6.3.5) - Minor changes (refer to modification strokes)
Ver P	14.09.20	Updated version (with firmware from version 3.19A): - Dual axis force control feature removed - New common parameter C3 (refer to §2.4.1)	- Power Sag Detection (refer to §7.15) - Motor braking updated (refer to §8.17.2)
Ver Q	22.03.21	Updated version (with firmware from version 3.20A): - Minor improvements	

Documentation concerning the AccurET family:

- | | |
|---|--|
| • AccurET Modular Operation &Software Manual | AccurET setup, use & programming manual |
| • AccurET Modular 48 Hardware Manual | Specifications & electrical interfaces |
| • AccurET Modular 48 Service Manual | Maintenance of the fuse |
| • AccurET Modular 300 Hardware Manual | Specifications & electrical interfaces |
| • AccurET Modular 300 Service Manual | Maintenance of the fuse |
| • AccurET Modular 400-600 Hardware Manual | Specifications & electrical interfaces |
| • AccurET Modular 400-600 Service Manual | Maintenance of the fuse |
| • AccurET VHP 48 Hardware Manual | Specifications & electrical interfaces |
| • AccurET VHP 100 Hardware Manual | Specifications & electrical interfaces |
| • AccurET VHP 100 Service Manual | Maintenance of the fuse |
| • ComET Communication Manual | Setting & monitoring |

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 regarding the system's functioning, installation, tuning, functions and programming possibilities. For electrical specifications, interfaces and hardware items, please refer to the corresponding '**AccurET Hardware Manual**'.



The information given in this manual is valid for a firmware from version 3.18A. If an UltimET motion controller is used, its firmware must be from version 3.18A. If EDI package is used, it must be from EDI4.22A for EDI4.xx. ComET must be from version 4.22A (Comet is a 32bits application). The AccurET firmware version 3.xx and above are only usable with an AccurET hardware revision 2 (refer to the corresponding 'Hardware Manual' for more information).

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

Safety

A circular icon showing a white silhouette of a person sitting and reading a book, set against a blue background.	<p>The user must have read and understood this documentation as well as those listed in page 14 before carrying out any operation on an AccurET position controller. Please contact ETEL S.A. or authorized distributors in case of missing information or question regarding the installation procedures, safety or any other issue.</p>
A yellow triangular warning sign with a black exclamation mark in the center.	<p>ETEL S.A. disclaims all responsibility for accidents and damages if the safety instructions, the procedures and the usage described in the present "Operation & Software Manual" are not followed (including the ones given in the manuals listed page 14).</p>

- 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 DC bus capacitors to discharge.
- In the controller, the **leakage current** through the protective conductor to the GND is greater than a.c. 3.5 mA.
- The safety symbols placed on the controller or written in the manuals ([page 14](#)) 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 a device sensitive to Electrostatic Discharges (ESD). All necessary precautions against ESD must be taken by the user.

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.

How to use this manual

If you are not an experienced user, read first the **Chapter A** to catch the **basis** about the controller's internal functioning and commands' syntax.

Then, follow the recommendation of the **Chapter B** to successfully realize the **first installation and setup** of the controller.

In **Chapter C**, [§6](#) and, [§7](#), all the **basic functions** necessary to operate the controller are described in details. **Chapter C**, [§8](#), is reserved for experienced users and describes **advanced functions** only used in specific applications.

Chapter D describes how to **program** the controller.

The appendixes in **Chapter E** include the **registers and commands references lists**, as well as **error and warning messages lists** and **units conversion formulas**.

Remark: ETEL can provide its customers with training courses including theoretical presentation and practice in real conditions, at our facilities in Môtiers (Switzerland).



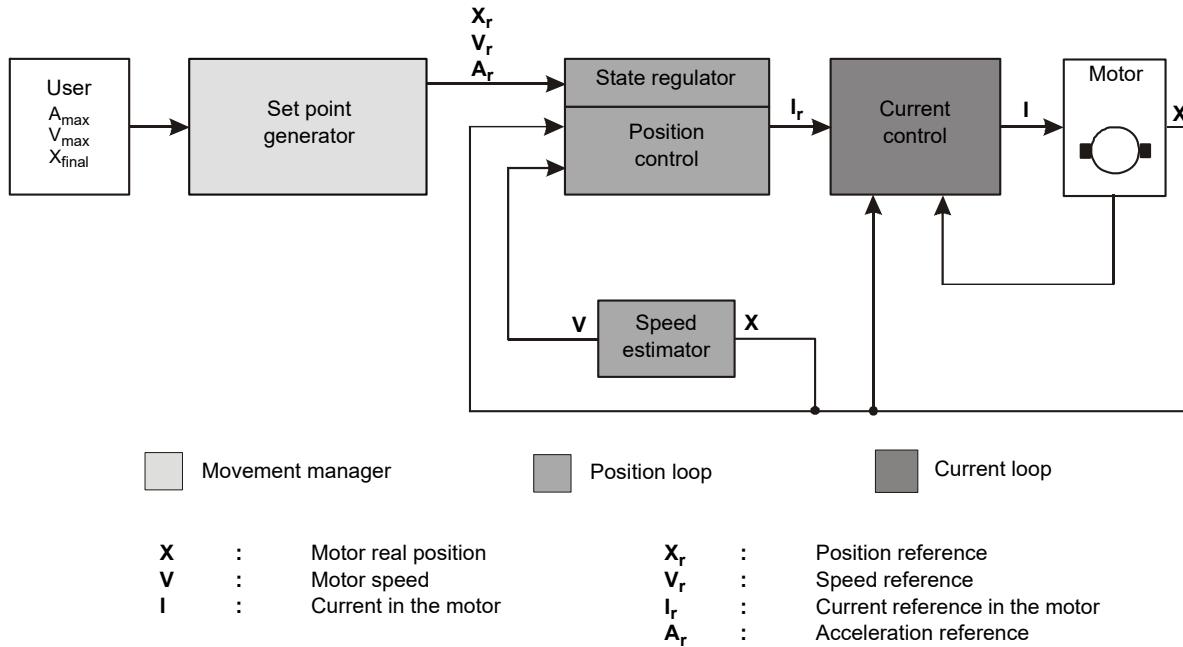
The AccurET Modular position controllers as well their corresponding power supply (if existing) have been successfully tested and evaluated to meet the UL 508C for US market.

This standard describes the fulfillment by design of minimum requirements for electrically operated power conversion equipment which is intended to eliminate the risk of fire, electrical shock, or injury to persons being caused by such equipment.

Chapter A: Internal functioning & generalities

1. Controller principle

1.1 AccurET modular controllers



A processor (a Sharc from Analog Devices) manages the controller made up of a **set point generator** and two regulation loops: the **position loop** and the **current loop**.

- The **set point generator** calculates the motor position, speed and acceleration references which are introduced in the position regulator. This calculation is made according to the type of requested movement, as well as the final position to reach, maximum authorized speed and acceleration given by the user. The set point generator carries out one of the most important functions of the controller: the **movement** calculation.

The cycle time of the set point generator, called **MLTI** (Manager Loop Time Interrupt), is 400µs. The MLTI value is also given by the monitoring **M245**.

- The **position loop** calculates the current reference I_r according to the reference calculated in the set point generator. This current reference is afterwards sent to the current loop.
The cycle time of the position loop, called **PLTI** (Position Loop Time Interrupt), is 50µs. The PLTI value is also given by the monitoring **M244**.
- In the **current control loop**, the current is calculated to be sent to the phases of the motor. The cycle time of the current loop, called **CLTI** (Current Loop Time Interrupt), is 50µs.
The CLTI value is also given by the monitoring **M243**.

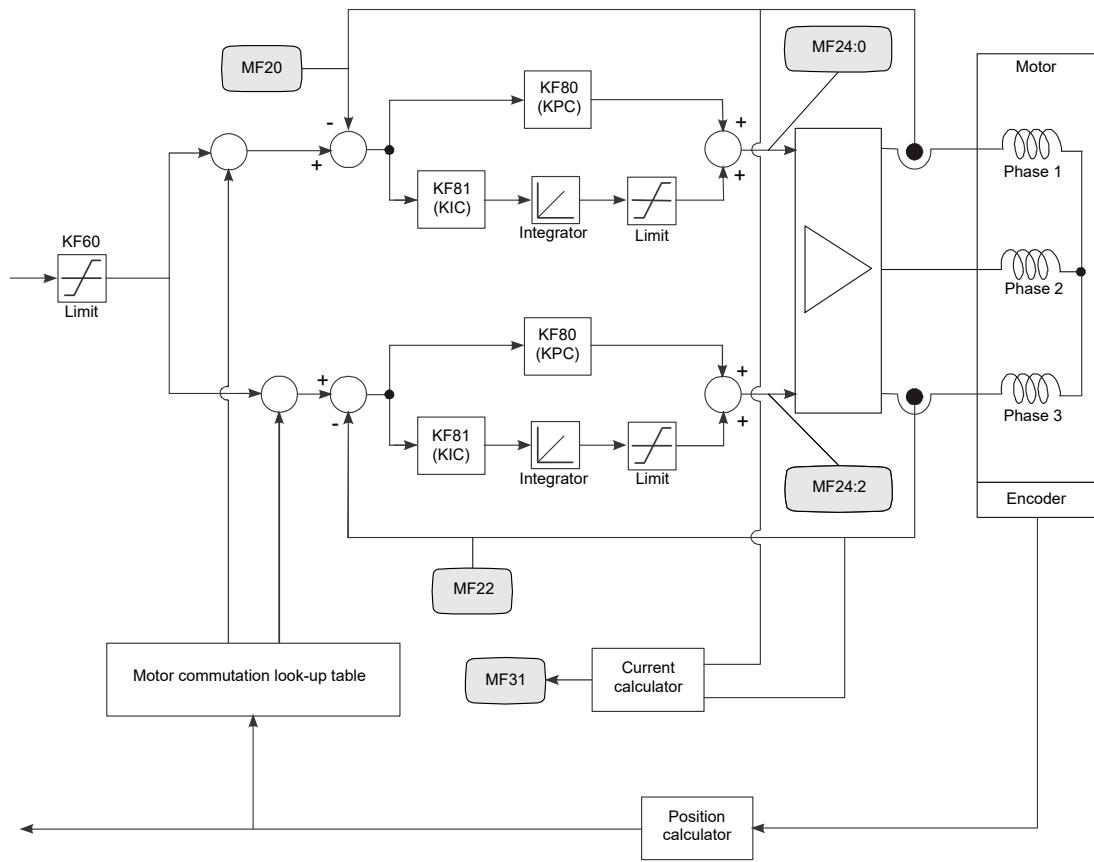
The elements of this regulation general diagram are detailed in [§4](#). (for beginners) and in [§8.2](#) (for advanced users).

Remark: If all features are enabled at the same time, a processor overload could occur which means that interruption times (MLTI/PLTI) could exceed their limit values. It is recommended to evaluate the risk with the tool 'Processor Load' available in ComET.

1.2 AccurET VHP controllers

The current outputs of the VHP controllers is driven by linear amplifiers instead of the switching amplifier (PWM) like on other AccurET controllers. The cycle time of the current loop (called **CLTI**), is 12.5 µs (80kHz). The position loop of the VHP controllers remains identical to the one of the other controller.

On a VHP controllers, the setting of the proportional and integral gain (parameters KF80 and KF81) of the current loops is independent of the Vpower DC Bus voltage. Therefore, KF80 and KF81 ISO gains in EDI / ComET are in [V/A] and [V/A/s] respectively and not in [1/A/s] as on the other AccurET controllers.



Remark: MF24 in EDI/ComET is in [V].
MF26 and MF29 do not exist in the regulator of the AccurET VHP.

2. Commands & registers syntax

2.1 Commands

2.1.1 Sending commands

Syntax:

<cmd_name>.<axis>[=<P1>[,<P2>]...]

Fields put in 'square brackets' (like: [=<P1>]) are **optional**. All commands do not use them.

<cmd_name> Command name. All controller's commands names have three letters or more.

.<axis> Axis or group of axis number which have to execute the command.

Possible values:

- **Integer from 0 to 62** according to the number of axis if the command refers to one single axis.
- Symbol ! if the command refers to all linked axes.
- Some selected axes numbers between commas.

[=<P1>[,<P2>]...] The command can have zero, one, two or more parameters (note: the = link sign is needed only if at least one [<Px>] field is present).

Possible values:

- Immediate value in integer 32 or 64-bits, or in float 32-bits
- Value contained in any register, at any depth. [Px] syntax is similar to a command syntax: **<register>[:<depth>].<axis>**. Refer to [§2.2.1.1](#) for more information.

Example:

```
WTM.1;                                // Command without parameter, sent to the axis 1.
SAV.(1,2,8)=1;                          // Command with one parameter (P1=1), sent to the axes 1, 2 and 8.
STA.1=1,7;                             // Command with two parameters (P1=1 and P2=7), sent to the axis 1.
```

Remark: Some commands calculate parameter values. For example, the AUT command calculates KF80, K56 and K53 parameters value. Others use the value contained in the parameters when they are executed. For example, theINI command has a different action according to the value contained in parameter K90.

If at least one parameter (P1,...) of a command is wrong (bad format for example), the **BAD CMD PARAM** error (M64=70) occurs.

Exception: The STE command (ex: STE_ADD.1=X4.1) uses + or - operators, with the following syntax:
<cmd_name>.<axis>[<operator>][=<P1>][,<P2>]
Refer to [§8.3.8](#) for more information.

2.2 Registers

2.2.1 Registers group

The **registers** are accessible to the user, they store all the controller's internal values. Each register has an identification number preceded by one or two letters corresponding to its group. There are 8 main types of registers, 4 are basic (often or always used) and 4 are advanced (for specific applications only).

2.2.1.1 Basic registers

K, for parameters	They define the configuration of one axis of the controller and the setting of its features. Refer to §17 for the complete lists of K parameters.
M, for monitorings	They are exclusively used to monitor the controller's internal values such as motor speed, acceleration, motor current, etc. They can only be read, and no value can be assigned. Refer to §19 for the complete lists of M monitorings.
X, for user variables	They are variables that the user may freely use for programming. Each user variable function may be defined in a program, according to the user's needs. Values can be stored in variables and read at any time.
C, for common	They allow the user to store configurations common to both axes on the same dual-parameter axes controller (the IP address for example). Reading or writing C registers from or to the first or second AccurET axis provides the same result. Refer to §18 for the complete lists of C common parameters.

Depending on the type of the associated value, a letter must be added to the one of the corresponding register: 'L' for 'Long' (integer 64 bits), 'F' for 'Float' (float 32 bits) and 'D' for 'Double' (float 64 bits). This rule can be applied to all registers. The registers name becomes as follows (with in bracket the type value used for indirect addressing in some commands or registers):

Register	Integer	Long	Float	Double
K	K (=2)	KL (=66)	KF (=34)	KD (=98)
M	M (=3)	ML (=67)	MF (=35)	MD (=99)
X	X (=1)	XL (=65)	XF (=33)	XD (=97)
C	C (=13)	CL (=77)	CF (=45)	CD (=109)

Registers with a same number but of different type are not linked. They are independent and have their own memory address.

When double values are stored on the flash with a SAV command, they are stored in 48-bit where the mantissa is coded on 32-bit. After the reboot of the controller, if the user reads the double afterwards, only 32-bit on mantissa are present. This means when the user compares the files generated in double (with 52-bit mantissa) with the value after a reboot (with 32-bit mantissa), the user will have a difference due to the truncate from 52-bit to 32-bit. The maximum number of the different registers varies according to the type of associated value:

Register	Integer	Long	Float	Double
K	$0 \leq K \leq 511$	$0 \leq KL \leq 511$	$0 \leq KF \leq 511$	$0 \leq KD \leq 511$
M	$0 \leq M \leq 511$	$0 \leq ML \leq 511$	$0 \leq MF \leq 511$	$0 \leq MD \leq 511$
X	$0 \leq X \leq 511$	$0 \leq XL \leq 255$	$0 \leq XF \leq 511$	$0 \leq XD \leq 255$
C	$0 \leq C \leq 511$	$0 \leq CL \leq 255$	$0 \leq CF \leq 511$	$0 \leq CD \leq 255$

The format of the corresponding values will be for example (as written in the 'Terminal' of ComET):

ISO	Integer	Long	Float	Double
123.0 1.2	123 1	123L 1L	123F 1.2F	123D 1.2D

Remark: An **alias** is a more user-friendly term, like SPD (for speed), ACC (for acceleration),..., representing a parameter K. Their syntax is identical to the one of the corresponding register. Refer to [§17](#), [§18](#), and [§19](#), to know the list of the alias.

2.2.1.2 Advanced registers

E , for Fast Triggers	They are used if the user's system has to react specifically when the motor reaches some defined positions. As these registers can only be associated to a 'Long' value type, it will always be written EL registers ($0 \leq EL \leq 255$). Refer to §8.10 for more information.
L , for look-up table	They allow the execution of movements with a user-defined trajectory and the storage of errors compensation for mappings. As these registers can only be associated to a 'Double' value type, it will always be written LD register ($0 \leq LD \leq 8191$).
T , for trace	They allow the acquisition of registers X, K, C, M and L of the controller versus time. They are used by the 'Scope' of ComET ($0 \leq T \leq 16383$). According to the format of the register, the trace will correspond to T, TF, TL or TD.
P , for mapping	They allow the user to store the correction points of the Stage Mapping.

2.2.2 Register value reading

Here is the syntax allowing the user to know the value in a register.

Syntax:

<register>[:<depth>].<axis>
--

Fields put in 'square brackets' (like: **[<depth>]**) are **optional**. They are not always used.

<register> Defines the register used. It must include the **type** of register as well as its corresponding **number**:

Possible types:

- **K, M, X, C, E, T, L, P**

Possible numbers:

- Integer (refer to [§2.2.1.1](#) and [§2.2.1.2](#))

[<depth>] Some registers may contain different values simultaneously. Each value is stored at a different **depth**. If no depth is defined, depth 0 is automatically used by default.

Possible values:

- Integer from **0** to **7** if the register type is **K** or **C**
- Integer from **0** to **3** if the register type is **X**
- Integer from **0** to **5** if the register type is **E**
- Integer from **0** to **3** if the register type is **T**
- Integer from **0** to **15** if the register type is **M**
- Integer from **0** to **9** if the register type is **L**
- Integer from **0** to **7** if the register type is **P**

Remark: The depth usage for the X registers is particular: from an application's point of view (ComET terminal or EDI application) X registers depths 0 and 1 point to the same value. Either depth can be set and the value is reflected on the other. Depths 2 and 3 are independent. When used in Compiled Sequences, when no depth is specified or if 0 is specified, it is the thread number in which the register is accessed that is used as the register depth.

<axis> Number of the axis whose register value must be read.

Possible values:

- **Integer from 0 to 62** according to the number of axis if the command refers to one single axis.

Example:

KF2 : 1 . 2 ; // Read the value of parameter KF2 at the depth 1 of the axis 2.

2.2.3 Register value writing

Here is the syntax allowing the user to change the value in a register.

Syntax:

<register>[:<depth>].<axis>[<operator>]=<P1>

Fields put in 'square brackets' (like: [<operator>]) are **optional**. They are not always used.

<register> Defines the register used. It must include the **type** of register as well as its corresponding **number**:

Possible types:

- K, X, C, E, L, P

Possible numbers:

- Integer (refer to [§2.2.1.1](#) and [§2.2.1.2](#))
- Y (indirect parameterization). Takes the value of the parameter K198.

[:<depth>] Some registers may contain different values simultaneously. Each value is stored at a different **depth**. If no depth is defined, depth 0 is automatically used by default.

Possible values:

- Integer from 0 to 7 if the register type is K or C
- Integer from 0 to 3 if the register type is X
- Integer from 0 to 5 if the register type is E
- Integer from 0 to 9 if the register type is L
- Integer from 0 to 7 if the register type is P

Remark:

The depth usage for the X registers is particular: from an application's point of view (ComET terminal or EDI application) X registers depths 0 and 1 point to the same value. Either depth can be set and the value is reflected on the other. Depths 2 and 3 are independent. When used in Compiled Sequences, when no depth is specified or if 0 is specified, it is the thread number in which the register is accessed that is used as the register depth.

.<axis>

Axis or group of axis number whose registers need to be modified.

Possible values:

- **Integer from 0 to 62** according to the number of axis if the command refers to one single axis.
- Symbol ! if the command refers to all linked axes.
- Some selected axes numbers between commas.

[<operator>]

Mathematical sign for arithmetic and logical operations. It is available only for K, X and C registers.

Possible values:

- + addition.
- - subtraction.
- * multiplication.

- / division.
- % modulo (not available for Float 32 and 64 bits).
- ~ bitwise NOT (not available for Float 32 and 64 bits).
- & bitwise AND (not available for Float 32 and 64 bits).
- | bitwise OR (not available for Float 32 and 64 bits).
- ^ bitwise XOR (not available for Float 32 and 64 bits).
- &~ bitwise NOT AND (not available for Float 32 and 64 bits).
- |~ bitwise NOT OR (not available for Float 32 and 64 bits).
- ^~ bitwise NOT XOR (not available for Float 32 and 64 bits).
- >> arithmetic shift to the right (not available for Float 32 and 64 bits).
- << arithmetic shift to the left (not available for Float 32 and 64 bits).

Refer to [§15.3.6](#) for more information.

= obligatory link sign.

<P1> Value of the register.

Possible values:

- Immediate value in Integer, Long, Float or Double.
- Value contained in another register at any depth (indirect value). It can be taken from any axis. <P1> syntax: <register_name>[:<depth>].<axis>

Example:

```
KL210:2.10=10000L; // The value 10000 is attributed to the depth 2 of parameter KL210 of the axis 10.
KL211:3.2=XL21.2; // The value of the user variable XL21 is attributed to the depth 3 of parameter KL211,
// both from axis 2.
X0.0=1050;
X0.0%=100;           // Modulo 100 => X0=50
```

The indirect parameterization (Y) allows the user to give the value of the parameter **K198** to a register when it is equal to Y or y.

K	Name	Comment
K198	Indirect register index	The register number is given by K198 when it is equal to Y or y (ex: Ky.1=...)

Example:

When the user writes XY.0=0 (or Xy.0=0) and K198=10, then it corresponds to X10.0=0.

It is possible to execute a command by using a register belonging to the second axis of the same controller. For example, if there is the axes number 1 and 2 on the same controller, it is then possible to have X1.1=M7.2.

Remark: For all registers K and C, a **maximum value** and a **minimum value** are defined. If a higher value than the maximum value or a smaller value than the minimum value is given to a register, the value will automatically be restricted by those two limits at any depth. A **default value** is also defined for each register.

2.3 Bits field or numerical values for registers and commands

The registers and commands' values can be of two types: **bits field** or **numerical values**.

2.3.1 Bits field

Some registers and commands values are defined with a bits field. Thanks to this feature, their corresponding functions may be combined by using a **binary mask**.

How to recognize them? If the numbers (0, 1, 2,...) are present in the table under the **Bit#** header, the register or the command value is defined as a **bits field**.

Example:

AUT is a command with 3 functions, defined with a bits field. Each function corresponds to a bit. Thanks to the binary mask, these 3 functions may be combined: $2^3=8$ possibilities.

Command	<P1>	bit#	Comments
AUT.<axis>=<P1>	1	0	Tunes the proportional and integrator gain of the current loop parameters KF80, KF81, and DC power voltage rate parameter K98.
	2	1	Searches motor phases and sets parameter K56.
	8	3	Sets parameter K53 (fine phase adjustment) if K52=1.

Binary mask example:

The user can tune the current loop parameters (bit# 0, **AUT=1**) and also search for the motor phases (bit# 1, **AUT=2**). Simply add the two bits, **AUT=3**, and both functions are combined.

Remark: When all bits are equal to 0, their functions are not active, that is why AUT=0 is not described in the above-mentioned table. This is the case in this document for all registers and commands values defined as a bits fields.

2.3.2 Numerical values

Most registers and commands are defined with a numerical value. Their corresponding functions cannot be combined. **How to recognize them?** If no number is present in the table under the **Bit#** header, the register or the command value is defined as a **numerical value**.

Example:

K90 is a parameter with 8 functions, each function corresponds to a different numerical value.

K	Name	Value	Comment
K90	Phasing mode and commutation	0	No phasing (with 1-ph. motor or absolute EnDat encoder). The value stored in K53 is used if K52=1
		1	Phasing with current pulses (3-phase ironcore motors only). The value stored in K53 is used if K52=1
		2	Phasing by constant current in the motor phases (ironcore and ironless motor). The value stored in K53 is used if K52=1
		3	Phasing with digital Hall sensors (mode 1) until the index is found then commutation by position encoder. The value stored in K53 is used if K52=1
		4	Phasing with digital Hall sensors (mode 2) until the first edge of signal then commutation by position encoder. When index is found, the value stored in K53 is used if K52=1
		5	Phasing and commutation with digital Hall effect sensor only. The value stored in K53 is used if K52=1
		6	Small movements phasing. The value stored in K53 is used if K52=1
		7	Small movements phasing. Repeated automatically 3 times when phasing process fails. The value stored in K53 is used if K52=1

2.4 Communication with the controller

2.4.1 Communication protocols

Four different communication protocols (TransnET, USB2.0, Ethernet and EtherCAT) are available to communicate with the AccurET controller. The common parameter **C1** allows the user to choose between the TransnET, TCP/IP Ethernet and EtherCAT as these protocols share the same connector:

C	Name	Value	Comment
C1	Manager mode	0	TransnET mode

Remark:

This value is taken into account only at the power on of the controller.

For the communication with external devices (camera, sensors, PLC, etc.) a Serial Peripheral Interface (SPI) is available. Refer to [§8.15](#) for more informations.

The command NEW configures the Manager mode back to default factory configuration, i.e. for AccurET EtherCAT C1=2 and for other AccurET C1=0.

For a simplified identification of the AccurET controller in ComET, it is possible to name it. This name is usually set with the ComET setting tool, but it can be manually introduced with the common parameter **C3** as well.

C	Name	Comment
C3	Controller name	Allows to enter a controller name

2.4.1.1 TransnET

TransnET is the ETEL proprietary fieldbus managing the real-time commands between the UltimET motion controller and the AccurET position controllers. It guarantees highly deterministic synchronization and extremely low jitter. TransnET allows master to slave, slave to master, or slave to slave communication within the same cycle.

When the TransnET protocol is selected and enabled, the **TIMEOUT TRANSNET** error (M64=53) will occur if the communication is broken and the controller does not receive TransnET messages.

2.4.1.2 EtherCAT

The AccurET modular 48, 300 and 400/600 supports the standard EtherCAT fieldbus. Please refer to the corresponding “User’s Manual” for more information.

2.4.1.3 TCP/IP Ethernet

For the TCP/IP mode, the common parameter **C2** allows the user to configure the IP address. The alias of this parameter is **IPADD (IP ADDress)**. The IP address must be first set with a USB connection (or eventually TransnET) prior to changing the communication mode to TCP/IP (C1 = 1).

C	Alias	Name	Comment
C2	IPADD	IP address	If C2=0 => a DHCP server is used If C2≠0 => the value of the parameter corresponds to the IP address. The address must have the following format: "xxx.xxx.xxx.xxx" with xxx going from 0 to 255

The monitoring **M204** shows the current IP address of the controller (alias **MIPADD: Monitoring IP ADDress**) and the monitoring **M205** gives the Ethernet status, which can be cleared with the **CEC** command.

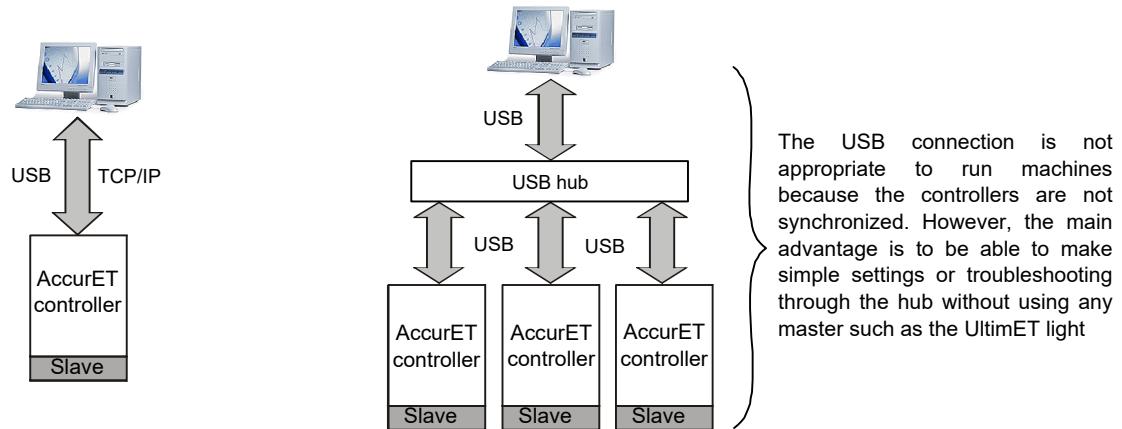
M	Alias	Name	Comment
M204	MIPADD	IP address	Gives the IP address
M205	-	Ethernet status	Gives the Ethernet status. Refer to the description of M205 in §19. for more information

Command	<P1>	Comment
CEC.<axis>=<P1>	1 2 4	Clears Ethernet CRC counter (bit#8 to bit#11 of Ethernet status M205) Clears Ethernet communication flags (bit#13 and bit#14 of Ethernet status M205) Clears Ethernet warning (bit#20 to bit#29 of Ethernet status M205)

2.4.2 Single-axis configuration

In this configuration, the PC and the controller can be linked through two different communication ways:

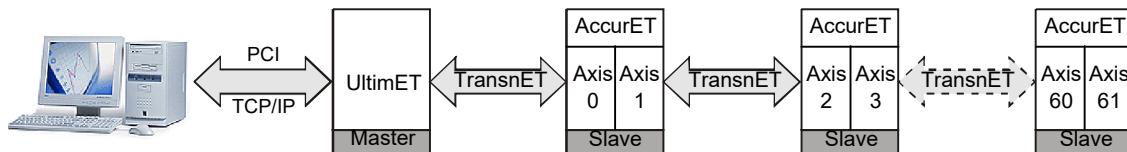
- USB2.0 full speed protocol
- Ethernet protocol



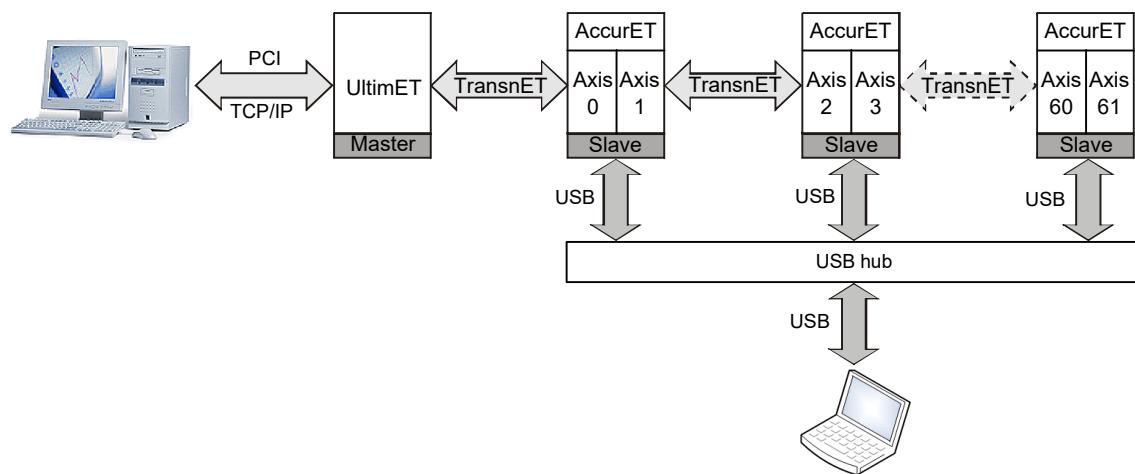
Remark: The USB protocol can be used for the debug and the setting of the controller whereas Ethernet must be used for the application management.

2.4.3 Multi-axis configuration

In this configuration, the controllers are linked in daisy-chain to a master (typically the UltimET light motion controller) communicating with the PC via PCI/PCIe or TCP/IP connector. All axes are linked together by the communication bus TransnET or EtherCAT. One of the roles of the master is to dispatch the orders received from the PC or (sent by itself) to the controllers. Each axis has a personal number, and if several axes are chained, their number must be different from each others (from 0 to 62). The master will always have the number 63 on TransnET. It is then possible to link up to 63 axes.

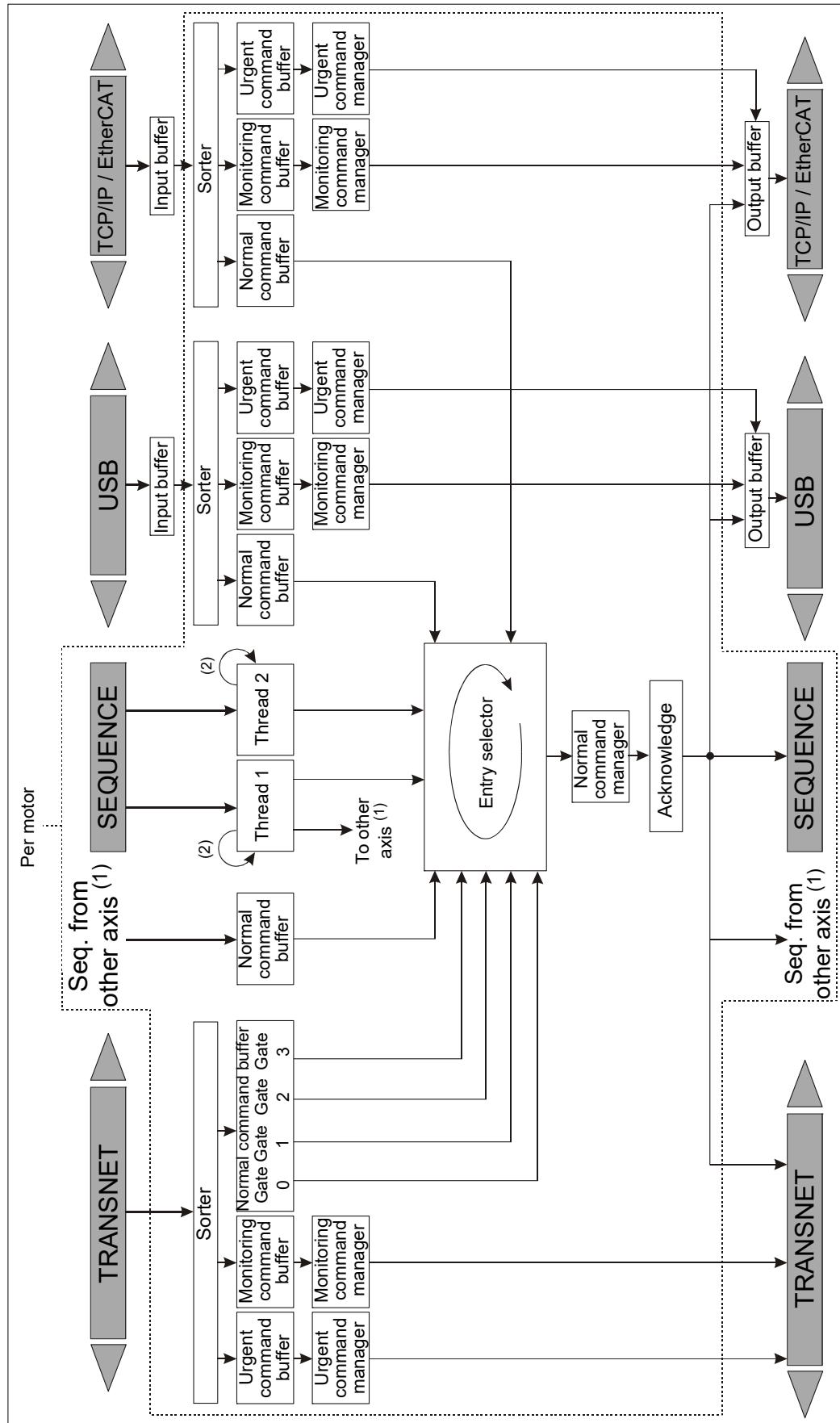


An additional USB communication can be used for synchronous monitoring on several controllers.



2.5 Commands manager

The command manager works at the manager loop time increment (MLTI, refer to [§1.](#)) interrupt frequency. There are several communication ports to send commands to the AccurET: TransnET, EtherCAT or TCP/IP (selection with C1 parameter, refer to [§2.4](#)), USB as well as Compiled Sequences (two threads per axis).



(1): the other axis is the one belonging to the same controller

(2): refer to the 'Compiled Sequence' paragraph

There are 3 main types of commands:

2.5.1 Urgent commands

There are some specific commands that must be executed immediately: immediate stop, stop with programmed deceleration and urgent power off (BRK, STP, HLT, HLB, HLO,...). It is possible to send these urgent commands through an EDI application or with the 'Terminal' of ComET. They are not available from the Compiled Sequences. In the 'Terminal', it is required to put an exclamation mark before the command (e.g. !HLB.0), otherwise the command is sent as a normal command (e.g. HLB.0). Each MLTI interrupt, the AccurET looks into each urgent command buffers (TransnET or TCP/IP and USB) if there are urgent commands to execute. The controller is able to execute one urgent command from each port in the same MLTI cycle which means one urgent command coming from USB and one from TCP/IP, TransnET or EtherCAT.

2.5.2 Monitoring commands

This command is used to read registers. It is possible to read up to two of any register types (not available for Compiled Sequences) in one monitoring command. Each MLTI interrupt, the AccurET looks into each monitoring command buffers (TransnET or TCP/IP and USB) if there are monitoring commands to execute. The controller is able to execute one monitoring command from each port in the same MLTI cycle which means one monitoring command coming from the USB and one from TCP/IP, TransnET or EtherCAT.

2.5.3 Normal commands

They are general commands that for example, put the power on the motor (PWR), start a movement (MVE), wait for the end of movement (WTM), etc (refer to [§16](#) for the complete list). Contrary to the urgent and monitoring commands, the controller cannot execute all commands coming from all the different communication ports during the same MLTI cycle. It can only execute maximum one per MLTI interrupt.

Before the normal command manager, there is the 'Entry selector'. An entry is the way from where a normal command can arrive. There are more entries than communication ports:

- 4 entries for the TransnET: gate 0, gate 1, gate 2 and gate 3. For example, we can imagine having gate 0 for the application on the PC, gate 1 for thread1 of UltimET (refer to the '**UltimET User's Manual**' for more information).
- 3 entries for the sequences: thread 1, thread 2 and thread 1 coming from the other axis belonging to the same controller.
- 1 entry for USB.
- 1 entry for TCP/IP.
- 1 entry fro EtherCAT

2.5.3.1 Entry selector

Each MLTI cycle, the 'Entry selector' scans each entry to check if there is a normal command. As soon as a command is present in an entry, the command is taken into account by the normal command manager. If the AccurET has just executed a command, the 'Entry selector' will start to scan the next MLTI cycle from the following entry. By default, each MLTI cycle, the 'Entry selector' will try to scan all the different entries. If an entry has a command and if the controller can execute it, the 'Entry selector' stops scanning. The wait commands type (WTT, WTM, WBC,...) are blocking only the entry from where this command has arrived, and not the other entries (refer to [§7.10](#)).

2.5.3.2 Normal command manager

When a command has been selected by the 'Entry selector', the normal command manager checks if it can start to execute or not this command during the same MLTI cycle. In fact, some normal commands have to wait for some conditions to be correct before executing it. For example, some normal commands cannot be executed if the motor is moving (SET, IND, AUT,...). In this case, the 'Entry selector' continues to scan the other entries. If another command is present on another entry and if the condition is correct to execute it, the controller will do it. When the condition becomes correct (for example the motor is not moving any more), the command manager can execute the command.

2.5.4 TransnET communication

The TransnET is used in a multi-axis configuration. The master of the TransnET is the UltimET light (motion controller), and the AccurET are the slaves. By this way, all commands from a host PC go through the UltimET. The 'Sorter' will put the command in the right buffer according to the type of command (urgent, monitoring or normal) and axis number. Urgent and monitoring commands are executed in the same MLTI cycle. The normal commands are managed as described here above by the normal command manager. Once the commands have been executed, the controller acknowledges them to indicate it can receive other commands.

The common parameter **C19** allows the user to enable the synchronous execution of commands sent over TransnET on several controllers. This parameter must be set to 1 to be activated (default value is 0). Depending on the timing, this option may lengthen the execution time of the commands by one MLTI.

C	Name	Comment
C19	Command synchronization	Enables the synchronization of commands between different controllers

2.5.5 TCP/IP and USB communication

The TCP/IP and USB communication ports have both an input buffer for both axes. Then they have a 'Sorter' to put the command in the right buffer according to the type of command type (urgent, monitoring or normal) and axis number. Urgent and monitoring commands are executed in the same MLTI cycle. The normal commands are managed as described here above by the normal command manager. Once the commands have been executed, the controller acknowledges them by putting them in the output buffer to indicate that it can receive other commands.

It is possible to send one command to both axes belonging to the same controller ($MVE.(0,1)=99282829L$), the sorter splits this command into two commands (one for both axes) and put them in the right axis buffer. Then each command manager will execute the command and acknowledge it. This means that the controller receives one command and acknowledges it twice (one for each axis). These two acknowledgements can occur at different moment.

It is also possible to execute a normal command with a parameter coming from the other axis belonging to the same controller. For example it is possible to execute $MVE.0=XL1.1$; the axis 0 first read XL1.1 and then executes MVE.0 with the read value.

2.5.6 Compiled sequences

Two threads can run simultaneously on each axis (refer to [§15](#)). Each thread has a time slot to execute commands in the MLTI cycle. The Compiled Sequences are optimized to execute more than one command per MLTI cycle and can be executed in parallel to the other normal command coming from different entries. They are optimized for fast execution on calculation, tests, loop... (mentioned by (2) on the command manager diagram [§2.5](#)). Nevertheless, some special commands cannot be executed directly by the thread (PWR, IND, ...) and must be given by the thread to the normal command manager. By this way, the command will be put to the entry selector, and will be managed by the other entries. The thread will stop executing command until this command has been executed by the command manager.

It is possible in a thread to execute a command with a parameter coming from the other axis belonging to the same controller. For example, it is possible to execute $XL1.0=ML7.1$. The axis 0 first reads ML7.1 and puts this value into XL1.0.

It is also possible from thread 1 **ONLY** to send a command to the other axis belonging to the same controller. For example in thread 1 of axis 0:

....
 $X0.0=10002002;$
 $X0.1=998272;$

In this case, the command $X0.1=998272$; is put into the entry 'Seq. from other axis' of axis 1, and will be managed by the normal command manager of axis 1. However, it is not possible in a thread to execute a command for the two axes belonging to the same controller: it is not possible to do $X0.(0,1)=10200299$; It is also not possible to have commands for an axis which does not belong to the controller. For example, it is not possible to have in a controller with axis 0 and 1, a command for axis 2.

Chapter B: System setup and tuning

3. Initial system installation

This chapter helps a new user to install a controller for the first time (in its minimal configuration). Refer to the corresponding '**Hardware Manual**' for more information about the electrical interfaces.

3.1 Controller connection

	First of all, always connect the protective earth (PE) before any other connection! Do not switch on the controller before all the connections are wired! Before connecting or disconnecting a cable on one of these connectors or touching the controller, turn off all the power supplies and wait 10 minutes to allow the internal DC bus capacitors to discharge.
---	--

Remark: Use the cables delivered with the controller(s). If you manufacture your own cables, refer to the corresponding '**Hardware Manual**' for more information about the pin assignment and the shielding.

3.1.1 Single-axis configuration

For a single-axis configuration, plug the connectors as listed below:

Interfaces to be connected	Required	Optional
1) Protective earth (refer to the corresponding ' Hardware Manual ' to know the connector location and condition)	X	
2) USB or TCP/IP communication (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
3) Position encoder (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
4) Motor connection (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
5) Power supply connection (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
6) Motor protection (refer to the corresponding ' Hardware Manual ' to know the connector location)		X
7) Customer I/O (refer to the corresponding ' Hardware Manual ' to know the connector location)		X

- When the connections are realized, turn on the power supply.
- Launch the ComET software to open the communication (refer to [§3.2](#)).
- A message (ACCURET READY) appears on the ComET software (the controller is not in the 'Power On' mode yet).

Remark: The status, the error and the warning messages of the controller are displayed on the status bar (at the bottom of the window) of the ComET software, however, the user can display them by using M95 (the conversion is automatically done by DLLs).

- The system is ready to be configured.

3.1.2 Multi-axis configuration

For a multi-axis configuration, plug the connectors as listed below:

Interfaces to be connected	Required	Optional
1) Protective earth (refer to the corresponding ' Hardware Manual ' to know the connector location and condition)	X	
2) PCI or TCP/IP communication between the PC and the master	X	
3) TransnET communication (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
4) Position encoder (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
5) Motor connection (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
6) Power supply connection (refer to the corresponding ' Hardware Manual ' to know the connector location)	X	
7) Motor protection (refer to the corresponding ' Hardware Manual ' to know the connector location)		X
8) Customer I/O (refer to the corresponding ' Hardware Manual ' to know the connector location)		X

- When the connections are realized, turn on the power supply.
- Launch the ComET software to open the communication (refer to [§3.2](#)).
- A message (ACCURET READY) appears on the ComET software (the controller is not in the 'Power On' mode yet).

Remark: The status, the error and the warning messages of the controller are displayed on the status bar (at the bottom of the window) of the ComET software, however, the user can display them by using M95 (the conversion is automatically done by DLLs).

- The system is ready to be configured.

3.2 ComET software installation

The **ComET** software is a user-friendly interface developed by ETEL to set up and monitor the operation of the controller. The users who are not accustomed to work with it should read first the '**ComET communication Manual**'.

Remark: The folder of installation as well as the backup of the files support **ONLY** standard characters (0 to 255) and **NOT** the extended characters.

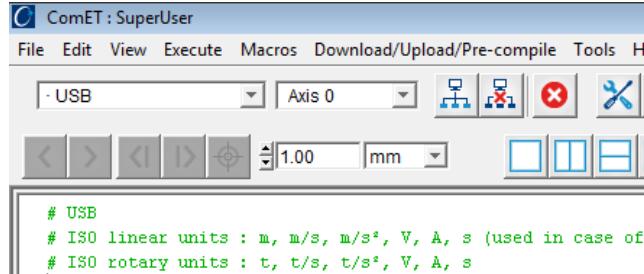
3.2.1 Establishment of the communication

The communication must be established prior to any operation otherwise the tools cannot be used (except for the firmware download). The connection is the first 'next suggested action' proposed by ComET. This suggestion is highlighted in dark blue on the main screen.

- **with a USB communication**

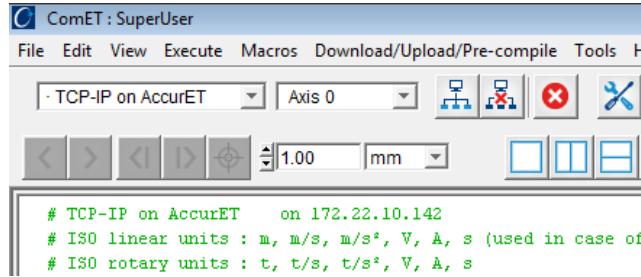
First of all, it is necessary to install the USB driver for the controller. To do so, connect the controller and when the operating system asks for a driver, select the one present in the ComET cd-rom under the «driver / usb» folder.

To do so, 'USB' must be selected in the top left corner menu:



- **with an Ethernet communication**

To do so, the controller must have the common parameter C1=1 (set and save by USB).



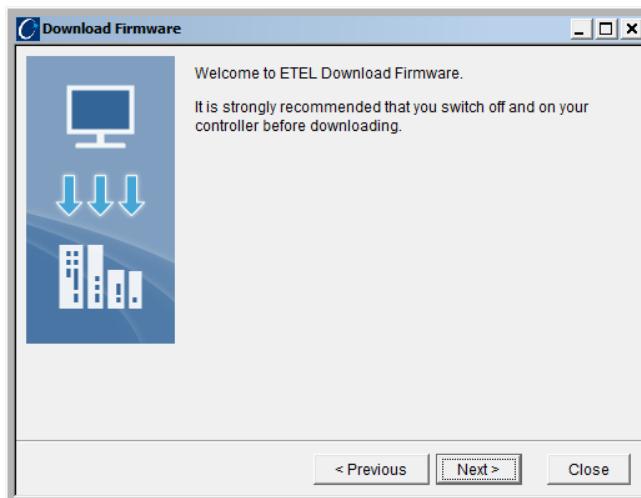
If the user wants to establish a new connection (with another hardware for example), the communication bus must be closed by clicking on this icon and the new hardware can then be selected.

Remark: Refer to the '**ComET Communication Manual**' for more information.

3.2.2 Firmware download

To download a new firmware (with USB, TransnET or TCP/IP), click on '**Download/Upload**' in the menu bar and then on '**Download Firmware**'. From now on, follow step by step the windows given by the '**Download Wizard**' tool.

- Up to firmware 2.06B, when a new firmware is downloaded, the links with the sequence are lost and it is then not possible to execute it. It is then necessary to have a backup of the sequence in a file to download it again once the firmware download is done. If this backup has not been done before the firmware download and if the original sequence file is not found, it is necessary to download the previous firmware version to be able to upload the sequence in a file from the controller. However it will work only if the sequence source was successfully downloaded.
- When upgrading from a firmware \leq 2.06B to a greater version, it is necessary to re-download the sequence (as mentioned above) otherwise the **FORMAT SEQ ERR** error (M64=460) will occur.
- For any upgrade with firmwares greater than 2.06B, it is not needed to re-download the sequence.



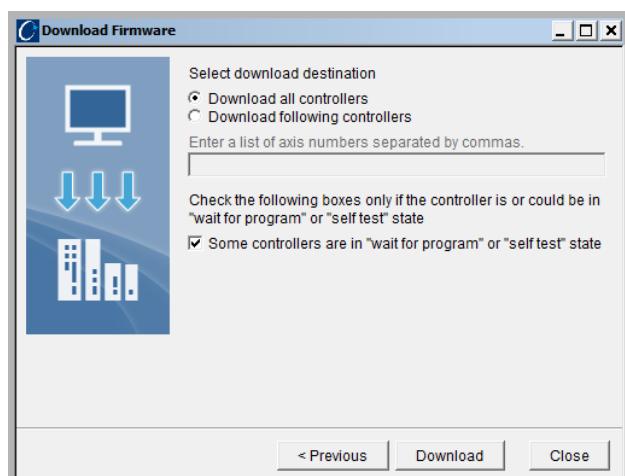
Remark: The name of the firmware file must have the **.far** extension.

The download through TransnET is possible from the following version: EDI 3.13A, AccurET firmware 2.04A and ComET 2.08A.

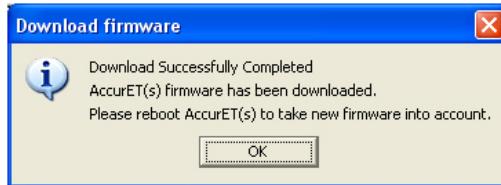
The download through TCP-IP is possible from the following version: EDI 3.15A, AccurET firmware 2.06A and ComET 2.10A.

If there is a problem during the download of the firmware, follows one of these 2 procedures:

- Stop the communication with the controller
- Switch off and on the power (during the starting phase of the controller, the red LED 'ERROR' and the green one 'POWER ON' are both switched ON)
- Restart the download procedure and select ONLY the USB communication when the port type is asked (in that case, the download is only possible with the USB cable). When the following window will appear, tick the box ('Wait for program') mentioned in the following picture.



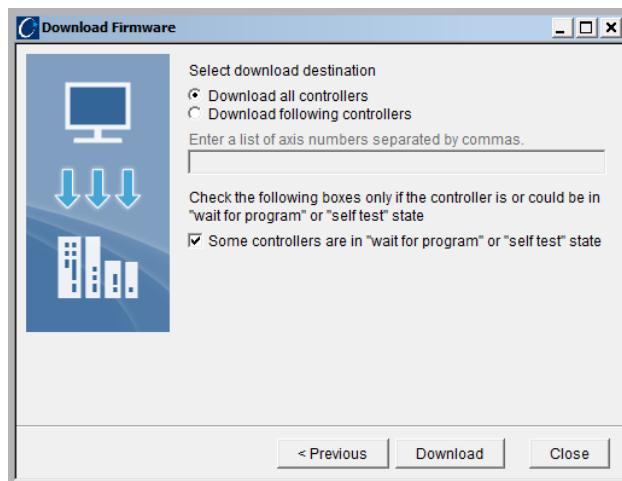
- Follow the information mentioned in the different windows up to the end of the download.



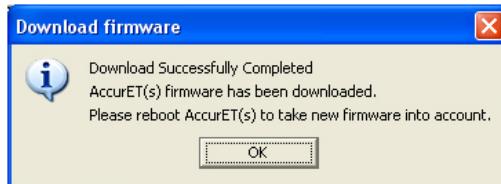
Remark: During the firmware download, the red LED 'ERROR' and the green one 'POWER ON' are both switched ON.

If after that, there is still a problem with the download of the firmware, follows these steps:

- Stop the communication with the controller
- Switch off the power
- Plug the jumper of the download key connector** (refer to the corresponding '**Hardware Manual**' for more information)
- Switch on the power (during the starting phase of the controller, the red LED 'ERROR' and the green one 'POWER ON' are both switched ON)
- Restart the download procedure and select ONLY the USB communication when the port type is asked (in that case, the download is only possible with the USB cable). When the following window will appear, tick the box ('Wait for program') mentioned in the following picture.



- Follow the information mentioned in the different windows up to the end of the download.

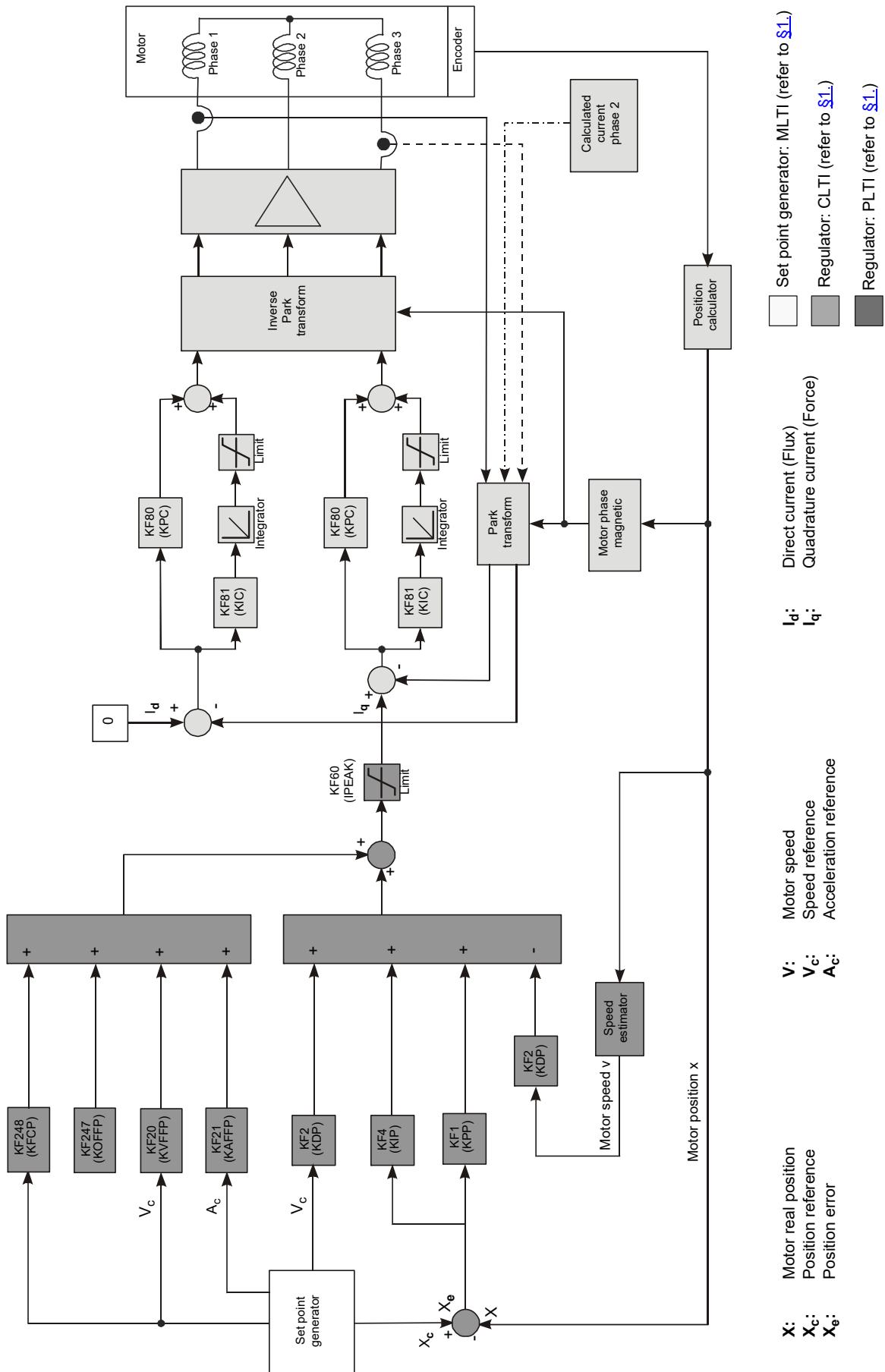


Remark: During the firmware download, the red LED 'ERROR' and the green one 'POWER ON' are both switched ON.

If after these 2 procedures, there is still a problem with the download of the firmware, please contact your ETEL's technical support.

4. Simplified regulator's principle

4.1 General diagram



4.2 Parameters description

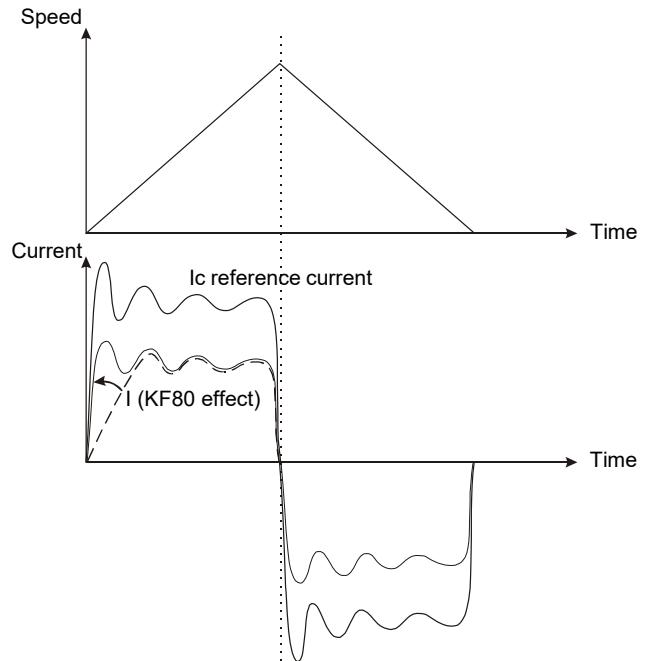
These explanations are more intuitive than theoretical. They should be used to 'feel' what happens and to help the reader to set up the regulator. The reader is supposed to have basic knowledge in regulation. This regulator's description is not complete, it has been simplified for a better understanding. However, the parameters mentioned in this paragraph are sufficient to set up the controller for simple applications. Nevertheless, the results will strongly depend on the type of application and system's working conditions.

4.2.1 Current regulator

In AccurET 48, 300 and 400/600, a current vector control is used to control the current output of the controller (except in stepper mode). In AccurET VHP, there is no vector control, but one current regulator on each phase 1 and phase 3. The parameters of the **current regulator** are:

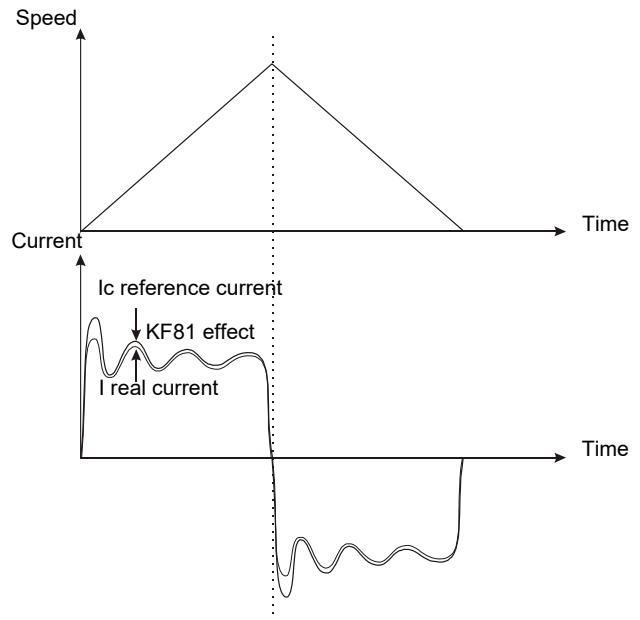
KF80: Proportional gain.

The main effect of the parameter KF80 is to make the current response faster.



KF81: Integral gain.

The main effect of the parameter KF81 is to suppress the current permanent error.



Remark: Refer to [§8.1.2.5](#) for the more information.

4.2.2 Position regulator

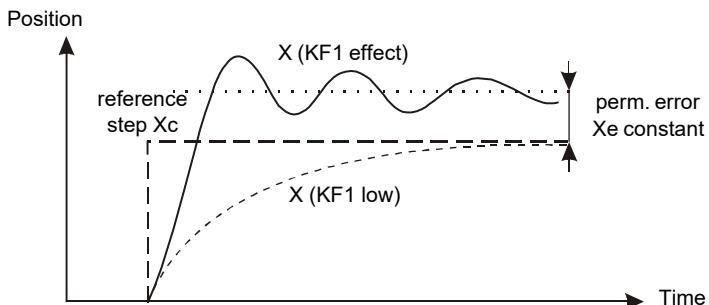
4.2.2.1 PID gains

The position is controlled by a state regulator which can be set by a proportional-integral-derivative (PID) regulator. The **position state regulator's** parameters are:

KF1: Proportional gain (P).

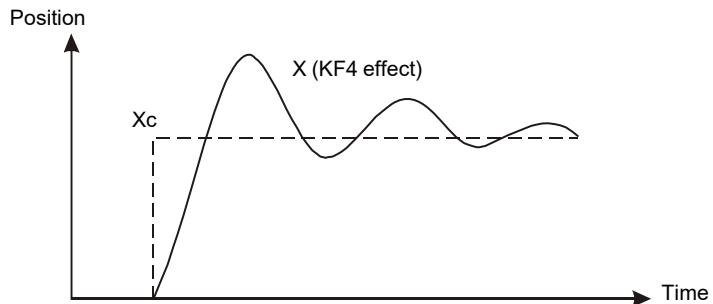
The main effect of the parameter KF1 is to make the position response faster, however, it may also create an overshoot, oscillations and a permanent error.

The position error is: $X_e = X_c - X$



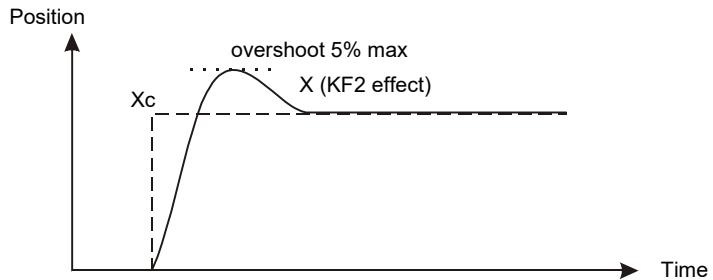
KF4: Integral gain (I).

The main effect of the parameter KF4 is to suppress the position permanent error (X_e constant).



KF2: Speed feedback gain, or pseudo-derivative gain (D). This is not a real position derivative gain because the input to KF2 does not directly come from the position error (X_e), but from the calculated speed (V).

The main effect of the parameter KF2 is to reduce or even suppress the overshoot and the low-frequency oscillations.



Remark: Refer to [§8.1.2.1](#) for more information.

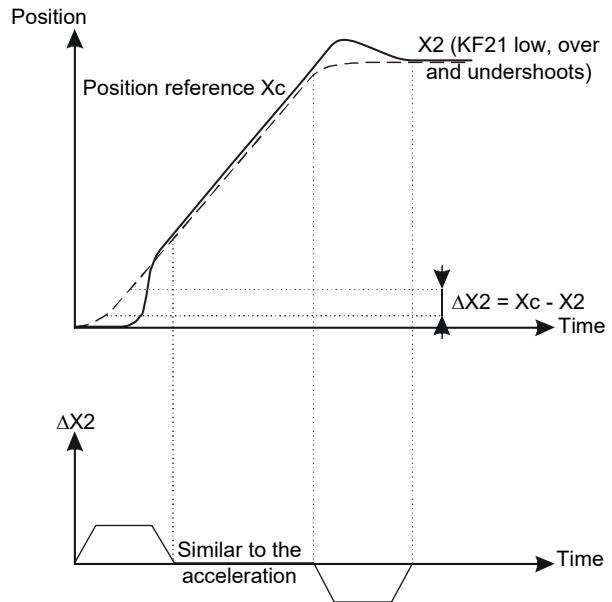
4.2.2.2 Feedforwards

During the acceleration and deceleration of a movement, a permanent error may appear between the position reference and the motor's real position. This drag effect can be due to mechanical frictions. It is possible to compensate it with the feedforward parameters. There are several feedforwards available in the AccurET firmware. The most important one is the acceleration feedforward described hereafter. This parameter will increase the acceleration command inputs to the state regulator.

KF21: Acceleration feedforward gain.

The difference between the position reference (X_c) and the real position (X_2) is similar to the acceleration's profile (ΔX_2).

The acceleration feedforward will compensate the undershoot and overshoot remaining after the speed feedforward compensation.



Remark: Refer to [§8.1.2.3](#) for more information.

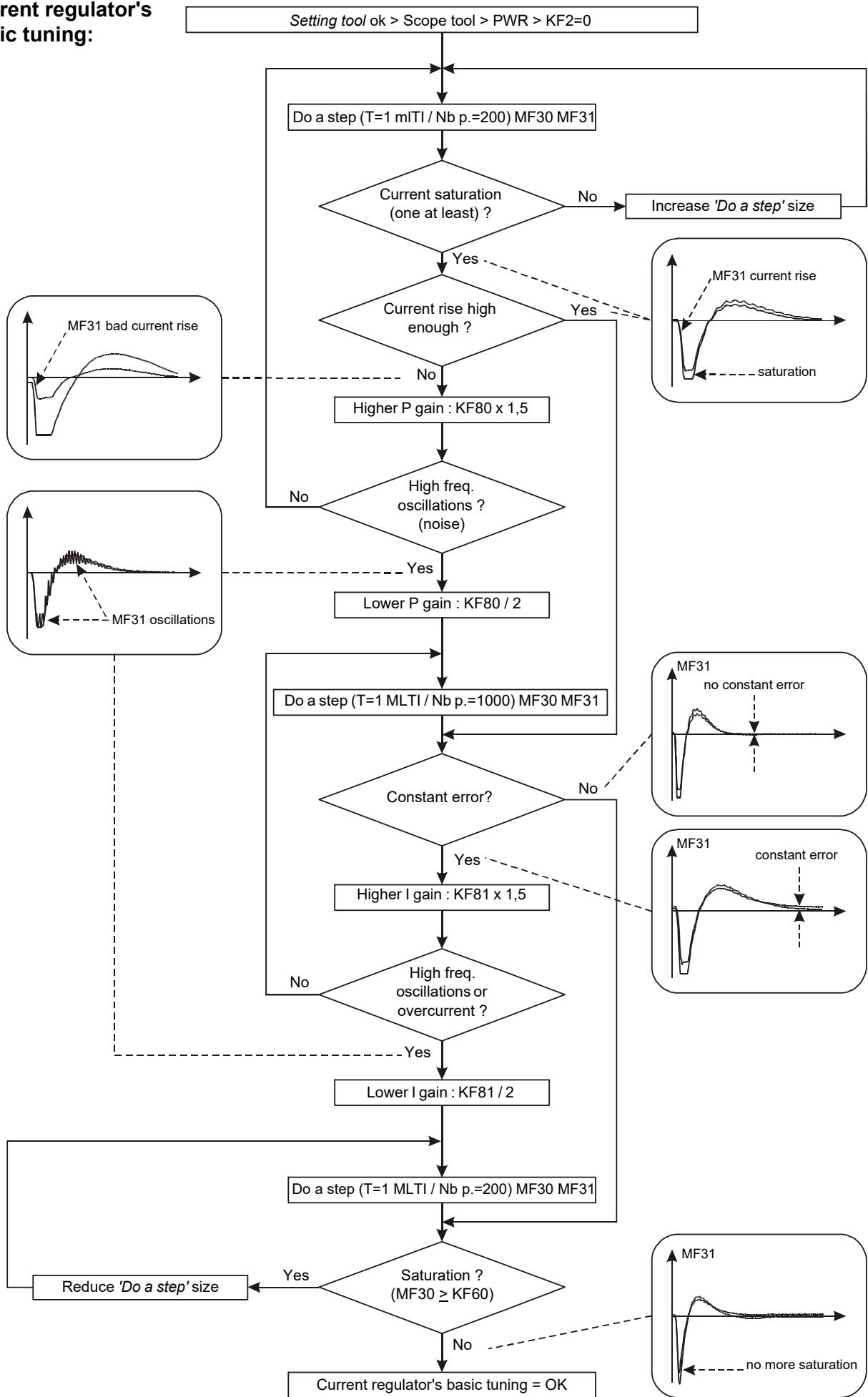
5. Controller regulators tuning principle

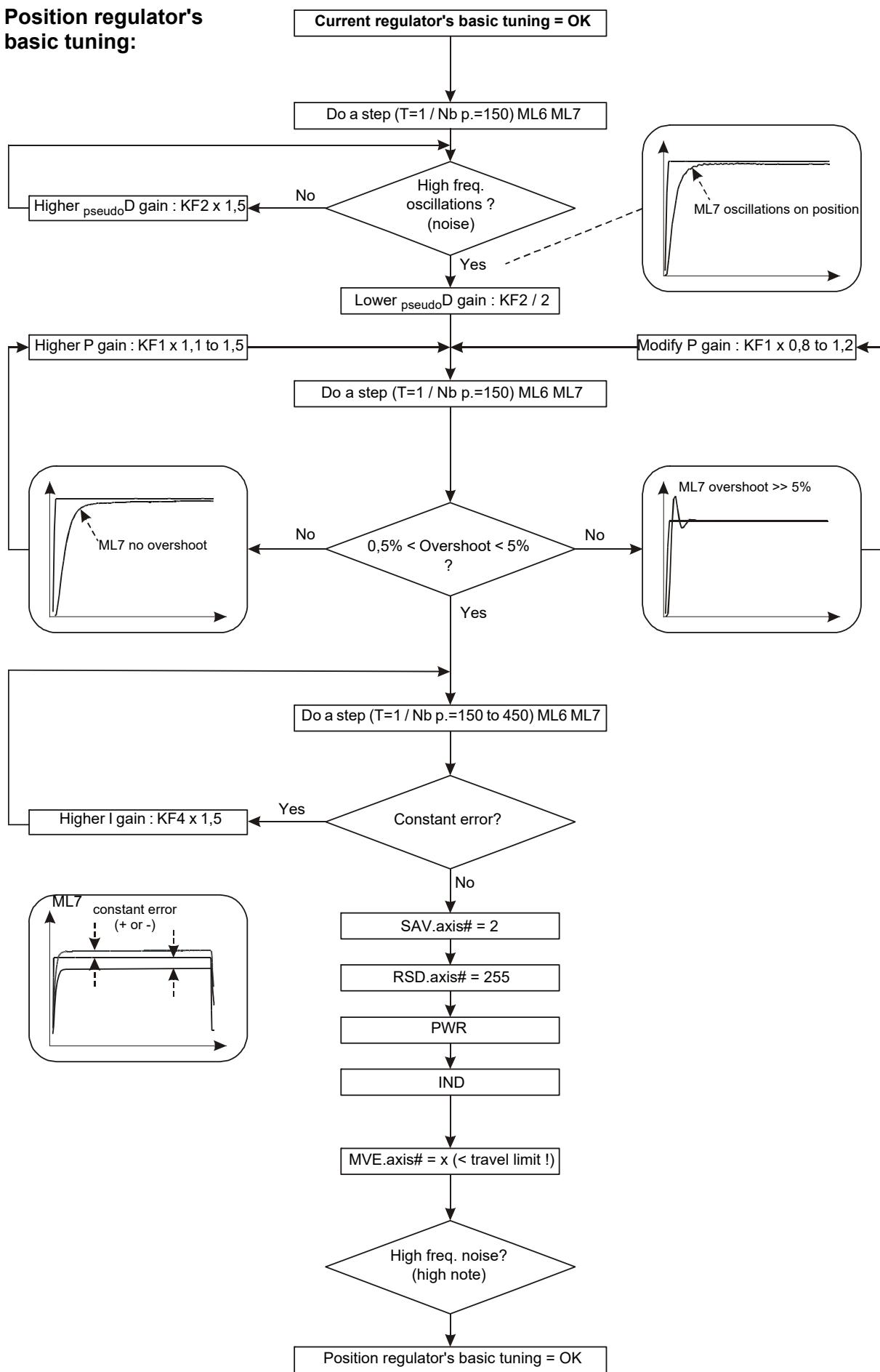
The tuning of the controller's regulators gains is necessary to adapt them to the characteristics of the motor and the associated load. ComET will be used to adapt and monitor them. Refer to the '**ComET Communication Manual**' for more information.

The following diagrams describe, in a basic way, the principle to tune the controller's current and position regulators. These procedures must be used as a guide line and cannot be considered as perfect for all applications. For a new user, a training can be given by an ETEL support engineer to be able to accurately tune a controller.

Basic rules:

- Start to tune first the loop to get the biggest bandwidth of the current loop;
- Tune the position loop;
- Tune the filters,
- Return to the position loop.

Current regulator's basic tuning:


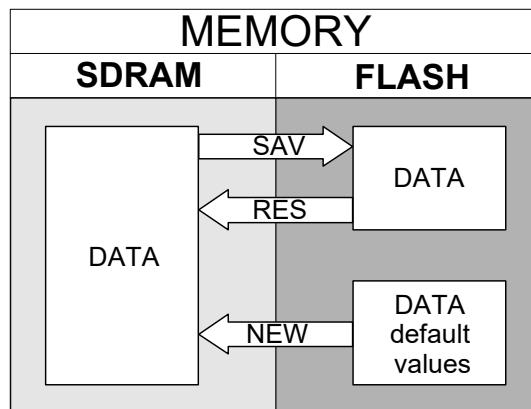
Position regulator's basic tuning:


Chapter C: System functions

6. Saving of the settings

The controller is set with **SDRAM** and **flash memory**. The SDRAM is a volatile memory which is erased each time the controller is switched off, whereas the flash is a non-volatile memory and the data stored inside are not lost when the controller is switched off. **All calculations and operations done by the controller are realized with the values present in the SDRAM.**

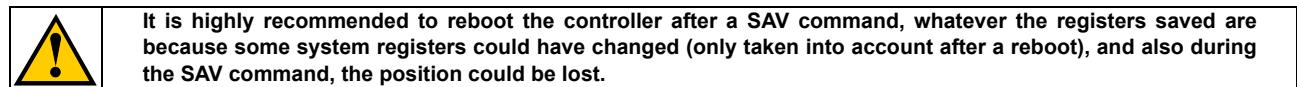
SAV, RES and NEW commands allow the user to transfer data from the SDRAM to the flash and vice-versa.



The **SAV** command (**SAVe**) saves into the 'flash' memory the controller's data (K, C, X, L, E and P registers) as well as the axis number and the sequence, so that they are not lost when the controller is switched off and on again. The type of saved data is defined by the parameter's value of the command.

Command	<P1>	Comment
SAV.<axis>=<P1>	0	Saves the sequence, LD registers (look-up table), K and C registers (parameter), EL registers (Fast Trigger), X registers (user variable), P stage error mapping variables and axis number in flash memory.
	1	Saves the sequence, LD registers (look-up table) in flash memory.
	2	Saves K and C registers (parameter), EL registers (Fast Trigger), X registers (user variable) and axis number in flash memory.
	3	Saves K registers (parameter), EL registers (Fast Trigger) in flash memory.
	4	Saves C registers (parameter) and axis number in flash memory.
	5	Saves X registers (user variable) in flash memory.
	6	Saves LD registers (look-up table) in flash memory.
	7	Saves the sequence in flash memory.
	8	Saves K registers (parameter) and EL registers (Fast Trigger) in flash memory.
	9	Saves P registers (Stage Mapping correction table) and stage error mapping variables in flash memory.

Command	<P1>	Sequence	LD	K	KF	KL	KD	C	CF	CL	CD	X	XF	XL	XD	EL	P	Axis number
SAV.<axis>=<P1>	0	x	x	x				x				x				x	x	x
	1	x	x															
	2				x			x				x						x
	3				x										x			
	4								x									x
	5										x							
	6		x															
	7	x																
	8				x										x			
	9															x		



The **RES** command (**REStore**) restores into the **SDRAM** memory the controller's data (K, C, X, L, E and P registers) as well as the axis number and the sequence previously saved with SAV command into the "flash" memory. The type of restored data is defined by the parameter's value of the command. **After a RES command, it is necessary to proceed to a new phasing procedure (INI command) and a new homing process (IND command).**

Command	<P1>	Comment									
RES.<axis>=<P1>	0	Restores the sequence, LD registers (Look-up table), K and C registers (parameter), EL registers (Fast Trigger), X registers (user variable), P stage error mapping variables and axis number from flash to sram memory.									
	1	Restores the sequence, LD registers (Look-up table) from flash to sram memory.									
	2	Restores K and C registers (parameter), EL registers (Fast Trigger), X registers (user variable) and axis number from flash to sram memory.									
	3	Restores K registers (parameter) from flash to sram memory.									
	4	Restores C registers (parameter) and axis number from flash to sram memory.									
	5	Restores X registers (user variable) from flash to sram memory.									
	6	Restores LD registers (look-up table) from flash to sram memory.									
	7	Restores the sequence from flash to sram memory.									
	8	Restores EL registers (Fast Trigger) from flash to sram memory.									
	9	Restores P registers (Stage Mapping correction table) and stage error mapping variables from flash to ram memory									

Command	<P1>	Sequence	LD	K	KF	KL	KD	C	CF	CL	CD	X	XF	XL	XD	EL	P	Axis number
RES.<axis>=<P1>	0	x	x		x			x		x		x				x	x	x
	1	x	x															
	2			x		x		x		x		x			x			x
	3			x														
	4						x											x
	5								x									
	6		x															
	7	x																
	8												x					
	9												x					

The **NEW** command **reloads** in the **SDRAM** the default values of K and C registers and **clears** the X, L, E and P registers stored in the SDRAM, depending on the parameter's value of the command.

Command	<P1>	Comment									
NEW.<axis>=<P1>	0	Clears the P Stage Mapping variables and sequence, sets default values for K parameter and C common parameters in ram memory									
	1	Clears the sequence in ram memory.									
	2	Sets default values for K and C registers (parameter) in ram memory.									
	3	Sets default values for K registers (parameter) in ram memory.									
	4	Sets default values for C registers (parameter) in ram memory.									
	5	Clears X registers (user variables) in sram memory.									
	6	Clears LD registers (look-up table) in ram memory.									
	7	Clears the sequence in ram memory.									
	8	Clears EL registers (Fast Trigger) in ram memory.									
	9	Clears P registers (Stage Mapping correction table) and stage error mapping variables in ram memory.									

Command	<P1>	Sequence	LD	K	KF	KL	KD	C	CF	CL	CD	X	XF	XL	XD	EL	P
NEW.<axis>=<P1>	0	x			x			x								x	
	1	x															
	2			x		x											
	3			x													
	4						x										
	5									x							
	6		x												x		
	7	x														x	
	8															x	
	9															x	

	After a NEW command that concerns K and C registers, these ones have default values that will not correspond to the ones of the dedicated motors. So it is necessary to download the K and C registers through a file (the registers must then be saved and the controller rebooted) or perform a 'drive setting' through ComET.
	After executing RES command, SDRAM ordinary values are replaced by the values read in the 'flash' memory and are definitively lost. Similarly, the SAV command crushes the values contained in the 'flash' with those contained in the SDRAM. SAV, RES and NEW commands perform a 'Power off' before being executed.

Remark: When the controller is **switched on**, data is automatically restored, like with a **RES.<axis>=0** command (so that it is not necessary to do it manually).

All registers' depths are saved and read when SAV, RES and NEW commands are executed. The command NEW configures the Manager mode back to default factory configuration, i.e. for AccurET EtherCAT C1=2 and for other AccurET C1=0.

Example:

The parameter KF1 is the position loop proportional gain. The default value of this parameter is 100F. The table below shows the state of the SDRAM and the 'flash' after the execution of different actions.

Actions	State of the SDRAM after action (the controller uses these values for all calculations)	State of the flash after action
Controller switched on	Parameter KF1 is equal to 100F (default value).	Parameter KF1 is equal to 100F (default value).
KF1.1=5000F	Parameter KF1 is equal to 5000F.	Parameter KF1 is equal to 100F (default value).
Controller switched off and on	Parameter KF1 is equal to 100F (default value).	Parameter KF1 is still equal to 100F (default value).
KF1.1=4000F	Parameter KF1 is now equal to 4000F.	Parameter KF1 is equal to 100F (default value).
SAV.1=2	Parameter KF1 is still equal to 4000F.	Parameter KF1 is now equal to 4000F.
NEW.1=2	Parameter KF1 is now equal to 100F (default value).	Parameter KF1 is still equal to 4000F.
RES.1=2	Parameter KF1 is equal to 4000F, value contained in the flash.	Parameter KF1 is still equal to 4000F.

This is possible to manage the registers and sequence versions used in each controller. The configuration of the version number is done in the ComET editor before the download of a sequence or registers to the controller. The file header contains the version of each register type present in the file.

Header example:

```
# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s
# ISO rotary units: t, t/s, t/s2, V, A, s
# ETEL Parameters Upload
# Written by "edi-tra" v2A
# Date: Mon Oct 18 09:42:20 2010
# User K,KL,KF,KD register version axis 0: 1.01A
# User X,XL,XF,XD register version axis 0: 2.01B
# User LD lookup table version axis 0: 1C
```

In order to set this version number, the user must either modify the current header (if present) or create a new one by using ComET editor function called 'Insert version header' (edit menu) and customize it.

The monitoring **M282** allows the user to know the registers and/or sequence version. The different depths (depths 0 to 2 are not used) of M282 correspond to the <P1> value of the SAV command. When the user uses the NEW command, the version of the registers and/or sequence corresponding to the <P1> value of the NEW command is set to 0. The registers and/or sequence version are saved in the same time as the registers and/or sequence with the SAV command.

Monitoring	Comment
M282	Register and/or sequence version (depth 0 to 2: not used, depth 3 to 9 as for SAV command)

The **CLX** command (**C**lear **X** variables) clears (resets) all X (X, XF, XL and XD) registers according to the <P1> value.

Command	Bit#	<P1>	Comment
CLX[.<axis>]=<P1>	-	0	Clears X, XF, XL, XD registers at all depths
	0	1	Clears X user variables at all depths
	1	2	Clears XF user variables at all depths
	2	4	Clears XL user variables at all depths
	3	8	Clears XD user variables at all depths

Example:

```
x2 .1=12 ; // The user variable X2.1 contains the value 12.
XF4 .1=20F ; // The user variable XF4.1 contains the value 20F.
CLX .1=0 ; // The variables X2 and XF4 (as well as all the others) contain the value 0.
```

7. Basic functions and settings

The functions defined in this chapter are necessary to operate the controller and are used for all applications. The advanced functions described in [§8.](#) are not used in basic applications and only advanced users may use them.

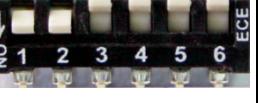
7.1 Axes number selection

The axis number of a position controller can be set either with a **DIP switch** or **by software** with the AXI command. Whatever the chosen solution, the user can display the axes number with the monitoring **ML83**. It is possible to link up to 63 axes.

M	Name	Comment
ML83	Controller axes mask	Give the mask of the axes number present on the controller

7.1.1 Selection with DIP switch

It is possible to assign or to change the axes number of the controller with a DIP switch. After each starting, the controller takes the axes number given by the DIP switch, except when all the switches are in the down position which means set to 1 (like in the picture below). In this case, the axes number is set to the value previously saved in the controller or to the default value equal to 0 for the first axis and 1 for the second (this default value is used when no AXI + SAV commands has been executed).

Default position		Example
	The value given on the DIP switch represents a binary value (64 possibilities). The axes are numbered from 0 to 62 because the node 63 is reserved for the UltimET. If the DIP switch is not used, all the bit# must be set to 1 (down position).	 The axes number given by this DIP switch is equal to: $2^0 + 2^1 = 3$. Then, the second axis of the controller will have the number 4.

Remark: All the axes numbers must be different. It is not possible to have twice the same axis number on the same communication bus.

7.1.2 Selection with command AXI

The **AXI (AXIs)** command is used to change an axis number. This command asks for the controller serial number and the new axis number. The serial number is asked by security to avoid an inappropriate change of the axis number. The AXI command can be used regardless of the position of the DIP switch. After having sent the AXI command, the user must save (with SAV=0, 2 or 4). The value given by the AXI command after a RSD will be taken into account only if all DIP switches are set to 1 (refer to [§7.1.1](#)).

Command	<P>	Comment
AXI.<axis>=<P1>,<P2>	<P1> <P2>	Changes the current axis number of a controller. Serial number New axis number for the first motor of the controller. The second motor of the controller will have the axis number <P2> + 1

The **SER** command (**SERial** number) which is an alias of the monitoring **M73** is used to know the serial number of the controller. It is requested by the AXI command before changing the axis number.

M	Alias	Name	Comment
M73	SER.<axis>	Serial number	Gives the serial number of the controller.

Example:

The user wants to change the axes number 2 and 3 into the axes number 6 and 7:
SER.2; // The controller gives its serial number (for example 4059).

```

AXI.2=4059,6; // The axis number is still the number 2.
SAV.2=2;      // The new axis number has been saved (the second axis number is calculated with regards
                // to the first one). The command SAV.2=4 works also.
RSD.2=255;    // Hardware reset of the controller.

```

The axis has now the number 6 (the second axis has now the number 7).

Remark: The first axis number can be an odd or even number.

7.2 Motor selection

The parameters described below are set only once and are automatically set with ComET during the auto-setting.

- **Movement type conversion (K240 parameter)**

K	Name	Value	Comment
K240	Movement type conversion	0	Linear movement (depth 0: main, depth 1: secondary, depth 2: auxiliary)
		1	Rotary movement (depth 0: main, depth 1: secondary, depth 2: auxiliary)
		2	Absolute EnDat linear movement (depth 0: main, depth 1: secondary, depth 2: auxiliary)
		3	Absolute EnDat rotary movement (depth 0: main, depth 1: secondary, depth 2: auxiliary)

This parameter is only used for calculation of the unit with ComET and the DLL libraries. If the values of this parameter is not correct, the motor will correctly work but the curves, the monitorings and the registers will not be displayed correctly in the units given by ComET.

The monitoring **M240** indicates the type of motor given either by the parameter K240 or the absolute EnDat encoder (depending on the encoder type selection (parameter K79)).

M	Name	Comments
M240	Movement type conversion	Gives the movement type conversion. 0: linear movement, 1: rotary movement (depth 0: main, depth 1: secondary, depth 2: auxiliary)

- **Number of phases and switching frequency (K89 parameter)**

K	Name	Value	Comment
K89	Motor phase number and PWM type selection	10	One-phase motor, PWM at 20kHz
		11	One-phase motor, PWM at 10kHz
		20	Two-phase motor, PWM at 20kHz
		21	Two-phase motor, PWM at 10kHz
		30	Three-phase motor, PWM at 20kHz not for Modular 600
		31	Three-phase motor, PWM at 10kHz
		34	Three-phase motor, PWM at 5kHz only for Modular 600

This parameter indicates the number of motor phases as well as the PWM frequency. There is no difference for the user if one, two or three phase motor is used as the meaning of each parameters is unchanged, commands are the same, etc. For especially high inductance motors, a specific switching frequency can be selected with parameter K89=11, 21, 31 or 34.

Remark: The parameter K89 is only read when the controller is **switched on**. This parameter must be saved with the SAV command when it is changed and then the controller must be switched off and on to integrate this new data. If this new value is incorrect, the **K89 BAD VALUE** error (M64=41) will appear.

- **Number of motor's pairs of poles (parameter K54)**

K	Alias	Name	Value	Comment
K54	PPOLE	Pairs of poles of the motor	-	Pairs of poles of the motor. For linear motor: K54=1

This parameter is used with rotary motors. It shows the pair number of motor's magnetic poles and is used by the motor commutation look-up table (LKT) which calculates the sinusoidal currents sent in the motor phases (refer also to [§7.7.1.4](#)). The **PPOLE** (Pairs of **POLE**) command is an alias of this parameter and uses the same syntax.

- **Motor phase and force inversion (parameter K56)**

K	Name	Value	Comment
K56	Motor phase and force inversion	0 1	Normal. Inverts phases and current signs.

This parameter enables the permutation by software of the motor phases connection as well as the sign of the motor current. If the phases have been inverted during the installation, any initialization will give a totally wrong value of K53 parameter (refer to [§7.7.1.4](#)). In this case, the AUT.<axis>=2 command must be sent to allow the automatic calculation of the adequate value for K56 parameter. **If you manually set K56 and K52=1, a AUT.<axis>=8 command must be performed to tune K53 parameter.**

7.3 Position encoder selection

The encoders (1Vpp, EnDat2.1, EnDat2.2 and TTL) can be used either as a main, secondary or auxiliary encoder. The difference between main, secondary and auxiliary is selected by the depth value of the corresponding parameter:

- The main encoder (depth 0) corresponds to the main encoder of the direct drive application.
- The secondary encoder (depth 1) is used when the application requires a second encoder like in a dual-encoder feedback application. This secondary encoder is taken into account in the regulation loop
- The auxiliary encoder (depth 2) is used to give a position feedback but it is not taken into account in the regulation loop

From version 1.00A, there were no amplitude error or loss of position error and no warning for a low amplitude on auxiliary analogue encoders.

From version 3.14A, with the **ERRORAUX** command, it is possible to decide whether the errors and warning should be generated or not.

From version 3.15A, the **ERRORAUX** command is replaced by K69 parameter. The alias for this register is **ERRORAUX**. The default value for K69 is 1 (errors and warning active).

K	Name	Value	Comment
K69	Allow error on auxiliary analog encoder	0 1	No error generated Error generated

The monitoring M102 allows the user to configure the Dual Encoder Feedback. The bit#4 M102 shows whether errors and warning are active or not.

M	Name	Value	bit#	Comment
M102	Dual encoder feedback configuration	1 2 16	0 1 4	Dual feedback configuration enabled Real dual feedback mode Error management on auxiliary analog encoder 1Vpp and HSEI enabled

- After K69=1, errors and warning are generated if necessary, bit#4 of M102=1
- After K69=0, no error are generated, bit#4 of M102=0.
- The **AUX ENCO AMPLITUD** error (M64=32) indicates an amplitude error and the **AUX ENCO POS LOST** error (M64=33) indicates a loss of position.
- The **AUX ENCO AMPLITUD** warning (M66=27) indicates an amplitude too low.

The parameters described below are set only once and are automatically set with ComET during the auto-setting.

K	Name	Value	bit#	Comment
K55 KL55	Encoder position increments factor	-	-	Number of [rdpi] per revolution for rotary motor (refer to S22.1.2.2 for more information) or number of [dpi] per magnetic period for linear motor (refer to S22.1.1.2 for more information). This parameter is used by the motor commutation look-up table (LKT). KL55 can be used instead of K55 when the value is bigger than $2^{31}-1$. For value lower than $2^{31}-1$, it is possible to use K55 or KL55 but it is not allowed to have a value different from 0 on both parameters (one of them must be at 0). If both parameters are different than 0, the NO PWR ON error (M64=72) will occur when the controller receives a PWR=1, INI or AUT=xx command.
K68	Encoder reading way inversion	1	0	Encoder reading way is inverted (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).
K70	Encoder sine offset correction	-	-	Correction of the sine signal offset (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).
K71	Encoder cosine offset correction	-	-	Correction of the cosine signal offset (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).
KF72	Encoder sine ampl. correction	-	-	Correction of the sine signal amplitude (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).
KF73	Encoder cos ampl. correction	-	-	Correction of the cosine signal amplitude (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).
K75	Distance between two indexes	-	-	Average distance between two indexes with a multi-reference marks encoder (the value is equal to 1 in case of mono-index encoder).
K79	Encoder type selection	0 2 3 4 5 20 98 99	-	1 Vpp analog encoder selection TTL encoder high frequency (10 MHz) selection EnDat 2.2 encoder without sine and cosine signals selection Absolute EnDat 2.1 encoder or EnDat 2.2 with sine and cosine signals selection HSEI Analog sine/cosine encoder 1Vpp (6MHz) (available only on VHP controllers) Absolute from any register No encoder and not possible to make a power on No encoder (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).
K241	Encoder period	-	-	Encoder period in [nm] for linear encoder or number of periods for rotary encoder (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).

Remark: If the parameter K68 is modified, the command AUT=10 must be executed to re-calculate parameters K56 and K53 if K52=1.

Parameter K75 does **not** exist with K79=3 and 4.

Parameter K79 is only read when the controller is **switched on** and must be saved with the SAV command when it is changed. Furthermore, the controller must be switched off and on to integrate this new data. If this new value is incorrect, the **K79 BAD VALUE** error (M64=40) will appear.

On the VHP controllers, it is possible to connect a 1Vpp incremental analog high speed encoder working up to 6MHz on the HSEI encoder connector (K79=5). To be able to use the index on the HSEI encoder, the homing speed must not exceed 500kHz frequency on the analog signal. If this is not the case, the **HOME SPEED** error will occur (M64=98).

For this encoder, a new '**W HSEI ENCODER**' warning has been implemented (M66=18) to indicate that the encoder's signals are not correct. It means that the encoder's head is probably not properly aligned with the encoder's scale.

For HSEI encoders, it is not possible to output the encoder signals to the fast digital outputs (K350, K351 and K352 parameters).

If the **SING IDX SEARCH** or **MULT IDX SEARCH** error (M64=62 and 61 respectively) occurs, the mechanical setting of the encoder's head must be done.

The monitoring M239 indicates the encoder period given either by the parameter K241 or the EnDat encoder (depending on the encoder type selection (parameter K79)).

M	Name	Comment
M239	Encoder period	Gives the encoder period in [nm] for linear encoder or number of period for rotary encoder (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder).

The currents sent to the motor phases are calculated by the current reference generator with a commutation look-up table. Here is the formula to calculate the number of increment (parameter K55 / KL55):

- For **rotary motor**:

$$K55 \text{ or } KL55 = M239 \cdot 1024 \cdot 2^{K77:0} \quad (\text{for principal analog and EnDat2.1 encoders})$$

$$K55 \text{ or } KL55 = M239 \cdot 64 \cdot 2^{K77:0} \quad (\text{for principal TTL encoder})$$

$$K55 \text{ or } KL55 = M239 \cdot 2^{K77:0} \quad (\text{for principal EnDat2.2 encoder})$$

- For **linear motor** (with K54=1 and Pmag=magnetic period [m] mentioned in the corresponding motor's data sheet):

$$K55 \text{ or } KL55 = \frac{Pmag}{M239 \cdot 1E-9} \cdot 1024 \cdot 2^{K77:0} \quad (\text{for principal analog and EnDat2.1 encoders})$$

$$K55 \text{ or } KL55 = \frac{Pmag}{M239 \cdot 1E-9} \cdot 64 \cdot 2^{K77:0} \quad (\text{for principal TTL encoder})$$

$$K55 \text{ or } KL55 = \frac{Pmag}{M239 \cdot 1E-9} \cdot 2^{K77:0} \quad (\text{for principal EnDat2.2 encoder})$$

The monitoring M241 indicates the encoder interpolation factor.

M	Name	Comments
M241	Encoder interpolation factor	Gives the interpolation factor of the encoder (depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder). In the above-mentioned formulas, monitoring M241 corresponds to $1024 * 2^{K77}$ (for an analog and EnDat2.1 encoder), $64 * 2^{K77}$ (for a TTL encoder) and 2^{K77} (for an EnDat2.2 encoder).

7.3.1 1 Vpp analog encoders (K79=0)

The **analog encoders** (1Vpp) can determine exactly the motor position thanks to two sinusoidal signals with a phase-shift of 90° (sine and cosine). The period of these signals varies according to the type of encoder used. The smaller the period is, the bigger the precision. These signals must be calibrated to have the same amplitude and no offset. Parameters K70, K71, KF72 and KF73 make such corrections. Some of the scales used with the analog encoders have multi-reference marks and in that case the average distance between reference marks must be given via the parameter K75. The formula is:

$$\text{For linear motors} \quad K75 = \frac{1024 \cdot \text{Distance between 2 indexes [m]}}{M239}$$

$$\text{For rotary motors} \quad K75 = \frac{1024 \cdot M239}{NRef} \quad \begin{aligned} NRef &= \text{Number of encoder's} \\ &\text{references marks per turn} \end{aligned}$$

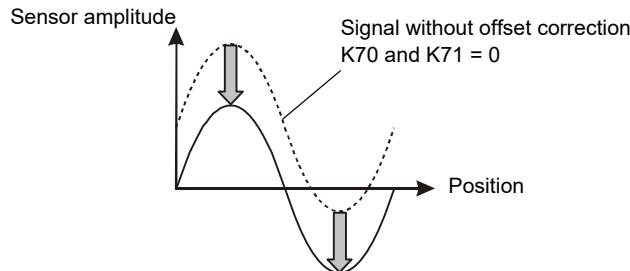
The encoder **resolution**, which is the smallest distance measured by the encoder, is given by the following formula:

$$\text{Analog encoder resolution [m]} = \frac{\text{Encoder period [m]}}{1024 \cdot \underbrace{2^{K77}}_{\text{Interpolation factor}}}$$

Remark: The parameter K77 has 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

- **Offset correction (K70, K71)**

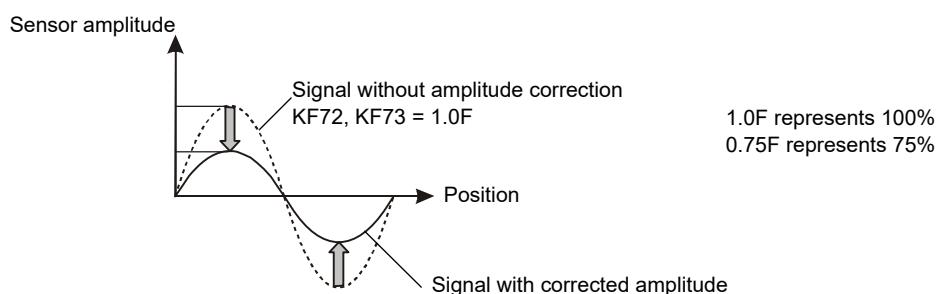
If parameters K70 or/and K71 are smaller than 0, a positive offset is added to the corresponding signal and a value higher than 0 adds a negative offset. If parameters K70 or/and K71 are equal to 0, no offset correction is done.



Remark: The parameters K70 and K71 have 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.
 Monitorings M40 and M41 allows the user to monitor the sine and cosine signals of the analog encoder (refer to [§7.3.11](#) for more information) after correction.

- **Amplitude correction (KF72, KF73)**

The amplitude correction allows the decrease of the signal amplitude but never the increase of it. That is why, **the tuning of the encoder's head** is important because it will not be possible to correct a too weak amplitude with the software. The default value (1.0F) does not correct any encoder signals amplitude.



Remark: The parameters KF72 and KF73 have 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

During a manual research of the offset and amplitude corrections, it is advised to start first disabling individually every signal offset, and then to correct the amplitudes in order to adjust the bigger on the smaller.

With ComET, the amplitude signal is used up to 1.3V without saturation.

At the main encoder level, the following message could occur: **ENCODER AMPLITUD** error (M64=20), **ENCODER POS LOST** error (M64=21) and **ENCODER AMPLITUD** warning (M66=5).

At the secondary encoder level, the following message could occur: **SEC ENCOAMPLITUD** error (M64=22), **SEC ENCOPOS LOST** error (M64=25) and **SEC ENCOAMPLITUD** warning (M66=26).

At the auxiliary encoder level, the following message could occur: **AUX ENCOAMPLITUD** error (M64=32), **AUX ENCOPOS LOST** error (M64=33) and **AUX ENCOAMPLITUD** warning (M66=27).

A period is divided into four equal parts and each of them is interpolated. The parameter **K77** determines this interpolation for analog encoders (1Vpp, EnDat2.1 and EnDat2.2 with sine and cosine signals). As the encoder signals are coded on 13 bits, it is possible to choose an interpolation included between 8 and 12 bits corresponding to an interpolation factor between 0 and 4.

Example:

We have an analog encoder with a period equal to 40 µm. If the maximum interpolation factor of the sine and the cosine (K77=4) is used, there are $1024 * 2^{K77} = 1024 * 2^4 = 16384$ increments on one period of encoder signals. The encoder resolution is:

$$\frac{40 \cdot 10^{-6}}{16384} = 2.44 \cdot 10^{-9} \text{ m}$$

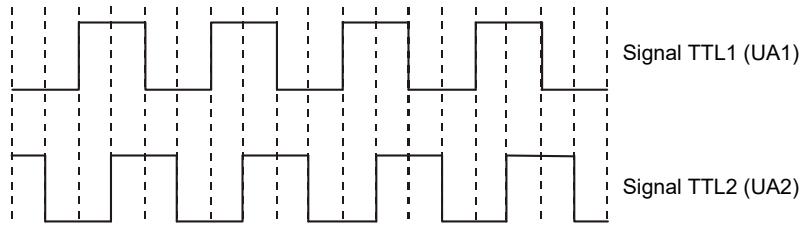
The parameter **K77** influences directly the motor position, speed, acceleration and jerk values (refer to [§7.12.3](#)).

K	Name	Value	Comment
K77	Encoder Interpolation shift value for encoder	-4	Interpolation of 64 dpi per encoder period
		-3	Interpolation of 128 dpi per encoder period
		-2	Interpolation of 256 dpi per encoder period
		-1	Interpolation of 512 dpi per encoder period
		0	Interpolation of 1024 dpi per encoder period
		1	Interpolation of 2048 dpi per encoder period
		2	Interpolation of 4096 dpi per encoder period
		3	Interpolation of 8192 dpi per encoder period
		4	Interpolation of 16384 dpi per encoder period
		5	Signals shift of 1 bit
		6	Signals shift of 2 bits
		7	Signals shift of 3 bits
		...	Signals shift of... bits
		12	Signals shift of 8 bits

Remark: The parameter K77 has 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.
With an analog encoder (1Vpp, EnDat2.1 and EnDat2.2 with sine and cosine signals), the minimum position reading resolution is 1 [dpi].

7.3.2 TTL encoders (K79=2)

TTL encoders measure the motor position with two phase-shifted TTL signals. Each change of state of one signal corresponds to a position increment. Parameters K70, K71, KF72 and KF73 are not used.



The **real position reading resolution** is given in the controller by parameter **K55** or **KL55** (refer to [§7.3](#)).

The encoder **resolution**, which is the smallest distance measured by the encoder, is given by the following formula:

$$\text{TTL encoder resolution [m]} = \frac{\text{Encoder period [m]} \cdot 16}{1024 \cdot 2^{K77}}$$

Remark: The parameter K77 has 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

With a TTL encoder, oscillations may happen when the motor stops and stays on position. When the motor moves, they disappear. They are due to the encoder weak resolution. A special **speed filter**, also named '**smooth filter**' has been created to reduce them. It can be set with the **parameter K11**. This parameter is taken into account only when the real speed is equal to 0. As soon as the real speed is different from 0, this parameter is deactivated.

K	Name	Comment
K11	TTL speed smooth filter	Speed smooth filter for TTL encoder

Remark: The **ENCODER LOST TTL** error (M64=240) is raised when a forbidden transition happens on TTL encoders. E.g. when the two signals simultaneously switch from high to low level.

The parameter **K77** determines the shift value. The parameter K77 influences directly the motor position, speed, acceleration and jerk values (refer to [§7.12.3](#)).

K	Name	Value	Comment
K77	Encoder Interpolation shift value for encoder	-4	Signals shift of 0 bits (=1 [dpi] min. resolution)
		-3	Signals shift of 1 bits (=2 [dpi] min. resolution)
		-2	Signals shift of 2 bits (=4 [dpi] min. resolution)
		-1	Signals shift of 3 bits (=8 [dpi] min. resolution)
		0	Signals shift of 4 bits (=16 [dpi] min. resolution)
		1	Signals shift of 5 bits (=32 [dpi] min. resolution)
		2	Signals shift of 6 bits (=64 [dpi] min. resolution)
		3	Signals shift of 7 bits (=128 [dpi] min. resolution)
		4	Signals shift of 8 bits (=256 [dpi] min. resolution)
		...	Value above 4 are normally not used

Remark: The parameter K77 has 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

7.3.3 EnDat 2.2 encoders without sine and cosine signals (K79=3)

The **EnDat 2.2** encoder is capable both of transmitting position values from incremental and absolute encoders as well as transmitting or updating information stored in the encoder. Thanks to the serial transmission method, only four signal lines are required. The period of these signals varies according to the type of encoder used.

Remark: The clock of the EnDat 2.2 is fixed at 6.25MHz.

Be careful of the length of the encoder cable (max. 40m). Refer to the encoder's manufacturer for more information about the EnDat 2.2 encoder's cable.

The Heidenhain designation of the EnDat 2.2 encoder is EnDat 22.

Limitations: The incremental EnDat encoders with batteries are not supported.

The "re-referencing" feature is not supported, i.e. it is not possible to repeat a homing process. Only the homing modes K40=8, 9, 10, 11, 16, 17, 18, 19, 24, 25, 26 and 27 are available for incremental EnDat 2.2.

The EnDat 2.2 incremental is not available in Dual Encoder Feedback mode.

The EnDat 2.2 incremental is not available in Gantry mode 1 and 2.

7.3.4 EnDat 2.1 encoders or EnDat 2.2 with sine and cosine signals (K79=4)

The absolute **EnDat 2.1 encoders** (analog) can determine the absolute position of the motor thanks to two sinusoidal signals with a phase-shift of 90° (sine and cosine) and the information given by its two serial lines (clock and data). The period of these signals varies according to the type of encoder used. The smaller the period, the higher the precision. These signals must be calibrated to have the same amplitude and no offset. Parameters K70, K71, KF72 and KF73 make such corrections.

The encoder **resolution**, which is the smallest distance measured by the encoder, is given by the following formula:

$$\text{Analog encoder resolution [m]} = \frac{\text{Encoder period [m]}}{1024 \cdot \underbrace{2^{K77}}_{\text{Interpolation factor}}}$$

Remark: The parameter K77 has 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder (refer to [§7.3.1](#) for more information).

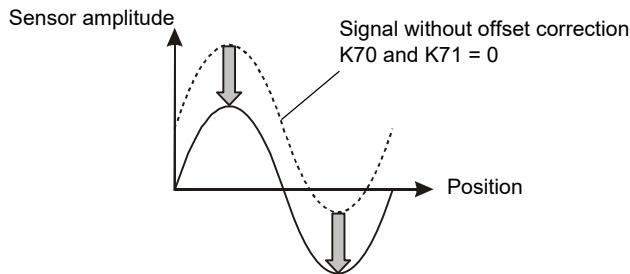
Be careful of the length of the encoder cable (max. 150 m with distributed capacitance 90 pF/m) because the clock frequency of the absolute EnDat 2.1 is equal to 500 kHz. Refer to the encoder's manufacturer for more information about the EnDat 2.1 encoder's cable.

The Heidenhain designation of the EnDat 2.1 and EnDat 2.2 encoders is EnDat 01 and EnDat 02 respectively.

- **Offset correction (K70, K71)**

If parameters K70 or/and K71 are smaller than 0, a positive offset is added to the corresponding signal and a

value higher than 0 adds a negative offset. If parameters K70 or/and K71 are equal to 0, no offset correction is done.

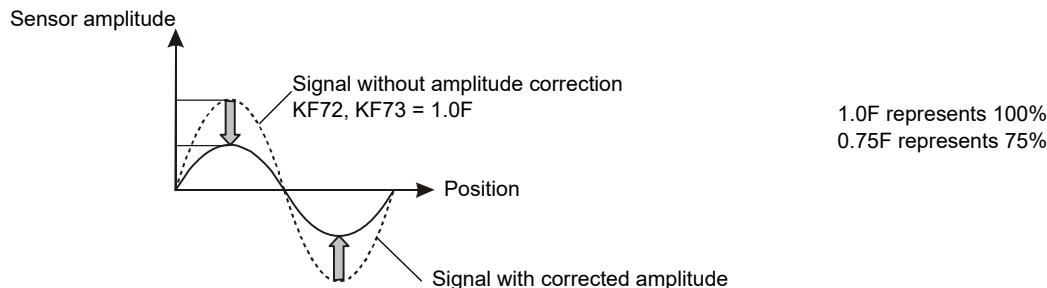


Remark: The parameters K70 and K71 have 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

Monitorings M40 and M41 allow the user to monitor the sine and cosine signals of the analog encoder (refer to [§7.3.11](#) for more information) after correction.

- **Amplitude correction (KF72, KF73)**

The amplitude correction allows the decrease of the signal amplitude but never the increase of it. That is why, **the mechanical tuning of the encoder's head** is important because it will not be possible to correct a too weak amplitude with the software. The default value (1.0F) does not correct any encoder signals amplitude.



Remark: The parameters KF72 and KF73 have 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

During a manual research of the offset and amplitude corrections, it is advised to start first disabling individually every signal offset, and then to correct the amplitudes in order to adjust the bigger on the smaller.

At the main encoder level, the following message could occur: **ENCODER AMPLITUD** error (M64=20), **ENCODER POS LOST** error (M64=21) and **ENCODER AMPLITUD** warning (M66=5).

At the secondary encoder level, the following message could occur: **SEC ENCOAMPLITUD** error (M64=22), **SEC ENCOPOS LOST** error (M64=25) and **SEC ENCOAMPLITUD** warning (M66=26).

At the auxiliary encoder level, the following message could occur: **AUX ENCOAMPLITUD** error (M64=32), **AUX ENCOPOS LOST** error (M64=33) and **AUX ENCOAMPLITUD** warning (M66=27).

7.3.5 1 Vpp HSEI analog encoders (K79=5)

The **HSEI analog encoders** (1Vpp) can determine exactly the motor position thanks to two sinusoidal signals with a phase-shift of 90° (sine and cosine). The period of these signals varies according to the type of encoder used. The smaller the period is, the bigger the precision. Some of the scales used with the HSEI analog encoders have multi-reference marks and in that case the average distance between reference marks must be given via the parameter K75. The formula is:

$$\text{For linear motors} \quad K75 = \frac{1024 \cdot \text{Distance between 2 indexes [m]}}{M239}$$

$$\text{For rotary motors} \quad K75 = \frac{1024 \cdot M239}{NRef}$$

NRef = Number of encoder's references marks per turn

The encoder **resolution**, which is the smallest distance measured by the encoder, is given by the following formula:

$$\text{Analog encoder resolution [m]} = \frac{\text{Encoder period [m]}}{1024 \cdot \underbrace{2^{K77}}_{\text{Interpolation factor}}}$$

Remark: The parameter K77 has 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

At the main encoder level, the following message could occur: **ENCODER AMPLITUD** error (M64=20), **ENCODER POS LOST** error (M64=21) and **ENCODER AMPLITUD** warning (M66=5).

At the secondary encoder level, the following message could occur: **SEC ENCOAMPLITUD** error (M64=22), **SEC ENCOPOS LOST** error (M64=25) and **SEC ENCOAMPLITUD** warning (M66=26).

At the auxiliary encoder level, the following message could occur: **AUX ENCOAMPLITUD** error (M64=32), **AUX ENCOPOS LOST** error (M64=33) and **AUX ENCOAMPLITUD** warning (M66=27).

On the VHP controllers only, it is possible to connect a 1Vpp incremental analog high speed encoder (1Vpp) working up to 6MHz on the HSEI encoder connector (K79=5). In that case, the parameter K77, influencing directly the motor position, speed, acceleration and jerk values, is as follows:

K	Name	Value	Comment
K77	Encoder Interpolation shift value for encoder	0	Interpolation of 1024 dpi per encoder period
		1	Interpolation of 2048 dpi per encoder period
		2	Interpolation of 4096 dpi per encoder period
		3	Interpolation of 8192 dpi per encoder period
		4	Interpolation of 16384 dpi per encoder period
		5	Interpolation of 32768 dpi per encoder period
		6	Interpolation of 65536 dpi per encoder period
		7	Interpolation of 131072 dpi per encoder period
		8 ... 12	Signals shift of 1 bit Signals shift of... bits Signals shift of 5 bits

Remark: The parameter K77 has 3 depths: depth 0 for the principal encoder, depth 1 for the secondary encoder and depth 2 for the auxiliary encoder.

7.3.6 Absolute position from any register (K79=20)

It is possible to use as an absolute position the value given by any register with the parameter K79=20. To do so, the parameters **K420**, **K421**, **K422** and **K423** must be defined.

K	Name	Value	Comment
K420	Type register for position coming via register	1	Indicates the register's type for position coming via register. (depth 0: main, depth 1: secondary, depth 2: auxiliary) Register type is a user variable X
		2	Register type is a parameter K
		3	Register type is a monitoring M
		13	Register type is a common parameter C
		33	Register type is a users variable XF
		34	Register type is a register KF
		35	Register type is a monitoring MF
		45	Register type is a common parameter CF
		65	Register type is a users variable XL
		66	Register type is a register KL
		67	Register type is a monitoring ML
		77	Register type is a common parameter CL
K421	Index register for position coming via register	-	Indicates the register's Index for position coming via register (depth 0: main, depth 1: secondary, depth 2: auxiliary)
K422	Depth register for position coming via register	-	Indicates the register's depth for position coming via register (depth 0: main, depth 1: secondary, depth 2: auxiliary)
K423	Use other axis register for position via register	-	Use other axis register for position coming via register (depth 0: main, depth 1: secondary, depth 2: auxiliary)
K424	Mean filter for position via register	1 2 - 8	Mean filter for position via register. (depth 0: main, depth 1: secondary, depth 2: auxiliary) No filtering Value * PLTI, length of the mean filter

The encoder period given by the parameter K241 (all depths) is in that case set by the following formula (the analog resolution is equal to the number of increments corresponding to the stroke):

$$K241 = \frac{\text{Stroke (nm)}}{\text{Analog resolution (inc)}}$$

When the absolute position comes via RTV, the parameter K241 must be equal to source register where the position comes from.

With an absolute position coming from a register, no interpolation is available (as K77 is not taken into account) and M241 is set to 1. This is true for all depth on K77 and M241.

As the interpolation factor is set to 1, it is necessary to use K242 and K243 to adjust the conversion in case of an encoder position coming through RTV. For example, if the interpolation factor is 8192 on M241 with the position sent through RTV, it is necessary to have for the conversion, K242=1 and K243=8192.

With this absolute position configuration, only K40=0, 1, 46 and 47 homing modes are possible.

With K79:1.<axis>=20, the position is present on M13 and ML13 and with K79:2.<axis>=20, the position is present on M15 and ML15.

When the position used by the controller comes from a register, there is no check of the signal integrity used to build this position as it is the case with 1Vpp signals directly connected to the controller.

The parameter K424 allows to smooth the input position when the update frequency of the register is smaller than 20.0 kHz (PLTI frequency). Depth 0 is used for main encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder.

7.3.7 Dual Encoder Feedback

The Dual Encoder Feedback feature has two modes of operation:

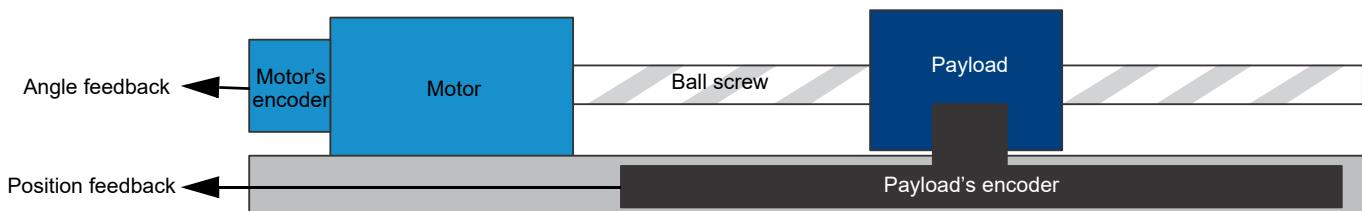
- Mode 0: Permanent Dual Encoder Feedback defined by K76:0
- Mode 1: Intermittent Dual Encoder Feedback defined by K76:1

If both depths of K76 are different than 0, the **ERR CFG DUAL FDB** error (M64=82) is raised.

7.3.7.1 Mode 0: Permanent Dual Encoder Feedback

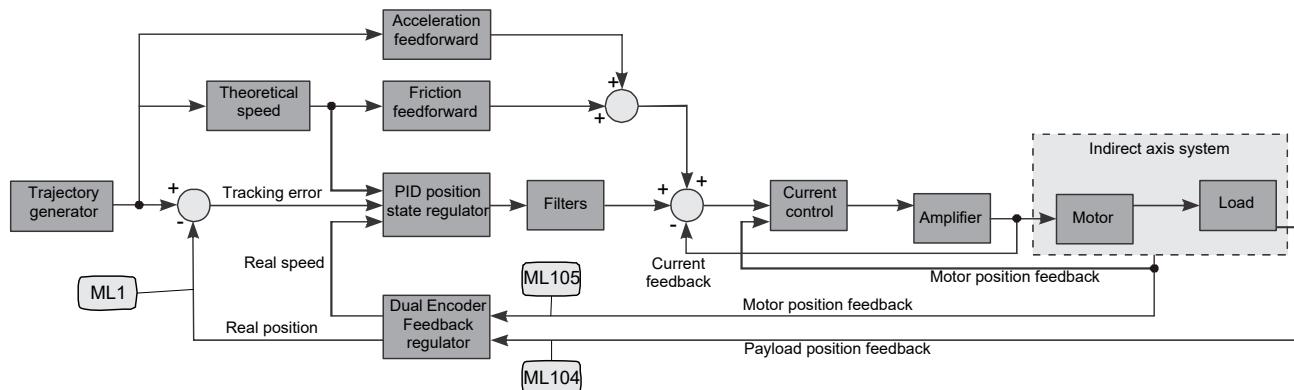
The Permanent Dual Encoder Feedback mode can be used to control an indirect axis equipped with 2

encoders: one directly mounted on the motor shaft (hereafter identified as motor position feedback) and another one mounted on the payload (hereafter identified as payload position feedback).



Remark: It is highly recommended to use the corresponding tool in ComET for setting this feature (available from ComET version 4.18A).

The position information from both encoders is used to control the indirect axis. On the other hand, everything related to motor commutation relies only on the motor position feedback. Both encoders must be connected to the same controller axis.



The parameter **K76:0** allows the user to select the Permanent Dual Encoder Feedback mode. The configuration is only taken into account at the moment the controller is switched on. To actually select this mode, the user must change K76:0 to a value other than 0, save it (with the SAV command) and reboot the controller. Setting K76:0 to 0 (default value) disables the Permanent Dual Encoder Feedback mode.

K	Name	Comment
K76	Dual Encoder Feedback mode	Set the Dual Encoder Feedback mode of operation. Depth0 for mode 0: Permanent Dual Encoder Feedback and depth1 for mode 1: Intermittent Dual Encoder Feedback.

- For all AccurET Modular controllers

K76:0 value	Payload's encoder (K79:0)	Motor's encoder (K79:1)
1	1 Vpp (K79:0=0)	TTL (K79:1=2)
2	1 Vpp (K79:0=0)	EnDat 2.2 (K79:1=3)
3	EnDat 2.2 (K79:0=3)	1 Vpp (K79:1=0)

Remark: For any of the configurations listed above, both encoders must be connected in the same connector at the controller side. Refer to the corresponding 'Hardware manual' for more information about the encoder connection.

If K76:0 has been assigned a value not listed above or not matching the corresponding configurations of K79:0 and K79:1, the **K79 BAD VALUE** error (M64=40) is raised by the controller start-up.

- For all AccurET VHP controllers

As there are more encoder interfaces available on the AccurET VHP, the parameter **K76:0** can be set to other values to enable additional position feedback configurations.

K76:0 value	Payload's encoder (K79:0)	Motor's encoder (K79:1)
1	1 Vpp (K79:0=0)	TTL (K79:1=2)
2	1 Vpp (K79:0=0)	EnDat 2.2 (K79:1=3)
3	EnDat 2.2 (K79:0=3)	1 Vpp (K79:1=0)
4	1 Vpp (K79:0=0)	HSEI encoder (K79:1=5)
5	EnDat 2.2 (K79:0=3)	HSEI encoder (K79:1=5)
6	HSEI encoder (K79:0=5)	1 Vpp (K79:1=0)
7	HSEI encoder (K79:0=5)	TTL (K79:1=2)
8	HSEI encoder (K79:0=5)	EnDat 2.1 (K79:1=4)
9	HSEI encoder (K79:0=5)	EnDat 2.2 (K79:1=3)

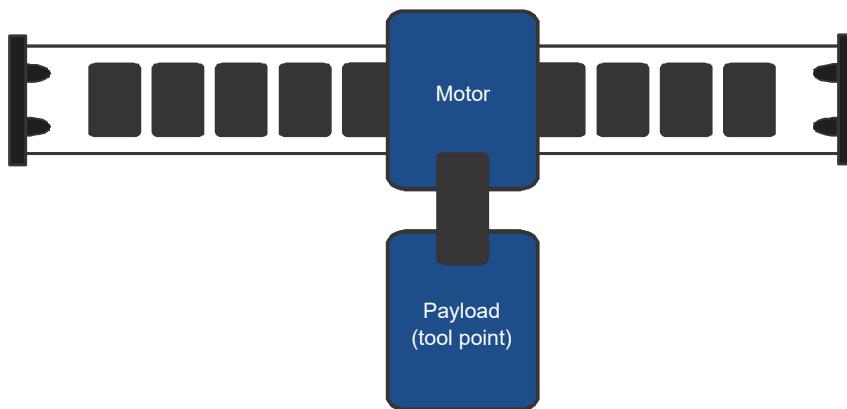
Remark: If K76:0 has been assigned a value not listed above or not matching the corresponding configurations of K79:0 and K79:1, the **K79 BAD VALUE** error (M64=40) is raised by the controller at start-up.

The monitorings **M104**, **ML104**, **M105**, **ML105**, **ML106**, **MF110** and **MF112** allow the user to measure the motor and payload real positions as well as the delta position between both.

M	Name	Comment
M104 ML104	Permanent Dual Encoder Feedback: Real payload position	Gives the real payload position in the Permanent Dual Encoder Feedback mode. The monitoring M104 corresponds to the LSL part of the monitoring ML104.
M105 ML105	Permanent Dual Encoder Feedback: Real motor position	Gives the real motor position in the Permanent Dual Encoder Feedback mode. The monitoring M105 corresponds to the LSL part of the monitoring ML105.
ML106	Real offset between motor and payload	Real offset between motor and payload. For the Permanent Dual Encoder Feedback mode, it gives the delta position between motor and payload when an absolute encoder is used.
MF110	Delta position between motor and payload	Gives the delta position between the motor and the payload.
MF112	Motor tracking error for Dual Encoder Feedback	Difference between the theoretical position (ML0) and the motor position available.

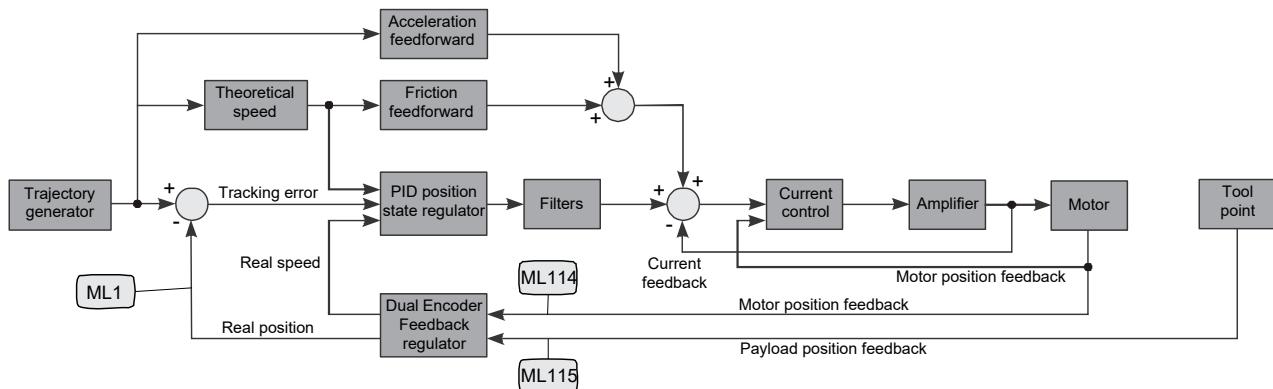
7.3.7.2 Mode 1: Intermittent Dual Encoder Feedback

The Intermittent Dual Encoder Feedback mode can be used to control a system and improve the accuracy at the tool point by using 2 encoders: one directly mounted on the axis and the other at the tool point.



The position information from both encoders is used to control the axis. Both encoders must be connected to the same controller axis. In this mode, the position feedback is then used as follows:

- Motor position feedback, in depth 0 (main encoder), is used for the commutation, homing and positioning. It must be always available.
- Payload position feedback, in depth 1 (secondary encoder), is only used to improve the positioning accuracy and might be temporarily unavailable. When it is not available, a warning is set



The parameter **K76:1** allows the user to select the Intermittent Dual Encoder Feedback mode. The configuration is only taken into account at the moment the controller is switched on. To actually select this mode, the user change K76:1 to a value other than 0, save it (with the SAV command) and reboot the controller. Setting K76:1 to 0 (default value) disables the Intermittent Dual Encoder Feedback mode.

K	Name	Comment
K76	Dual Encoder Feedback mode	Set the Dual Encoder Feedback mode of operation. Depth0 for mode 0: Permanent Dual Encoder Feedback and depth1 for mode 1: Intermittent Dual Encoder Feedback.

- For all AccurET Modular and VHP controllers

K76:1 value	Motor's encoder (K79:0)	Payload's encoder (K79:1)
20	1 Vpp (K79:0=0)	Absolute position from any register (K79:1=20)
21	EnDat 2.2 (K79:0=3)	Absolute position from any register (K79:1=20)

Remark: If K76:1 has been assigned a value not listed above or not matching the corresponding configurations of K79:0 and K79:1, the **K79 BAD VALUE** error (M64=40) is raised by the controller at start-up.

The monitorings **ML106**, **ML114**, **ML115**, **MF110** and **MF112** allow the user to measure the motor and payload real positions as well as the delta position between both.

M	Name	Comment
ML106	Real offset between motor and payload	Real offset between motor and payload. For the Intermittent Dual Encoder Feedback mode, it gives the delta position between motor and payload after the VALPOS command.
ML114	Intermittent Dual Encoder Feedback: Real motor position	Gives the real motor position in the Intermittent Dual Encoder Feedback mode.
ML115	Intermittent Dual Encoder Feedback: Real payload position	Gives the real payload position in the Intermittent Dual Encoder Feedback mode.
MF110	Delta position between motor and payload	Gives the delta position between the motor and the payload.
MF112	Motor tracking error for Dual Encoder Feedback	Difference between the theoretical position (ML0) and the motor position available.

The **VALPOS** command enables the Intermittent Dual Encoder Feedback mode. It must be used every time the payload position is recovered after an interruption.

Command	<P1>	Comment
VALPOS.<axis>=<P1>	0 1	Disable the Intermittent Dual Encoder Feedback feature Enable the Intermittent Dual Encoder Feedback feature

The payload position (available in M13/ML13) is taken into account as an absolute position. It means M13/ML13 is not affected by the IND command and the homing offset (ML5). However, the user can manually add

this offset thanks to the OFFSETSEC command (refer to [§7.12.1.2](#)) with the secondary offset value as parameter. The monitoring ML33 allows then to show the secondary offset value applied.
As for the SET command, the OFFSETSEC command sent without parameter is equal to OFFSETSEC=0.

The parameter **KF453** allows the user to smooth the transition when activating or disabling the Intermittent Dual Encoder Feedback mode with the VALPOS command. By default the smoothing time is 50msec (KF453=0.050).

K	Name	Comment
KF453	Smoothing time	Gives the smoothing time when activating or disabling the Intermittent Dual Encoder Feedback

7.3.7.3 Common configuration and registers

The following registers are common for both Dual Encoder Feedback modes.

The monitorings **M102** and **M103** can be used to obtain the configuration and status of the Dual Encoder Feedback feature:

M	Name	Comment
M102	Dual Encoder Feedback configuration	Gives the Dual Encoder Feedback configuration

- bit#0: Dual Encoder Feedback feature enabled.
- bit#1: Mode 0: Permanent Dual Encoder Feedback enabled.
- bit#2: Mode 1: Intermittent Dual Encoder Feedback enabled.
- bit#4: Allows the error management on auxiliary analog encoder.
- bit#5: Ignores errors from the secondary encoder.

M	Name	Comment
M103	Dual Encoder Feedback recovery status	Gives the Dual Encoder Feedback recovery status

- bit#0: invalid position of the secondary encoder.

The parameter **KF446** is then used for the tuning of the balance between both feedbacks in the Dual Encoder Feedback regulator.

K	Name	Comment
KF446	Accuracy convergence factor	Gives the accuracy convergence factor used by the Dual Encoder Feedback regulator

With KF446=0.0F, the Dual Encoder Feedback algorithm privileges the motor position feedback. Inversely, with KF446=1.0F, the Dual Encoder Feedback algorithm privileges the payload position feedback.

Remark: Privileging too much the motor position feedback might affect system positioning performance. On the other hand, privileging too much the payload position feedback might affect system robustness.

7.3.7.4 Axes setting for Permanent Dual Encoder Feedback mode

It is highly recommended to use the corresponding tool in ComET for setting this feature (available from ComET version 4.18A).

The following 2 steps must be followed to set an axis using the Permanent Dual Encoder Feedback mode:

- Setting the axis in single encoder feedback mode using either the motor encoder or payload encoder. ComET 'Drive setting' tool can be used for fine tuning the current loop, position loop, filters and advanced feedforward.
- Swapping of the 2 encoders and setting the Dual Encoder Feedback feature.

Example:

This example explains how to set a ball screw axis with the following characteristics:

- Motor encoder: rotary encoder (TTL) with 37500 line/rev and interpolation factor 64 (K77=0);
- Payload encoder: linear encoder (1Vpp) with 20 µm scale period and interpolation factor 8192 (K77=3);
- Ball screw pitch: 5.0 mm/rev.

Step 1a: Setting axis in single encoder feedback mode

- Start a drive setting of the motor with the rotary encoder using ComET's 'Drive setting' tool;
- Select the correct indirect configuration: indirect linear (motor: rotary and payload: linear)
- Select the motor rotary encoder as position feedback;
- Set correctly the ratio of the ball screw, belt, gear box, etc.;
- Follow through the setting procedure until the end (check the encoder direction, protections, soft limit, current limit, I_{2t}, etc.).

After this step, the registers should look like this:

	Characteristics	TTL encoder (rotary)	1 Vpp encoder (linear)
	Encoder period or line/rev.	37500 line/rev.	-
	Interpolation factor	64	-
	Encoder declaration in AccurET	Main	-
Depth	Encoder register depth	0	-
K68	Encoder reading way inversion	0	-
K76	Dual Encoder Feedback configuration	0	-
K77	Encoder interpolation shift value	0	-
K79	Encoder type (0 for 1 Vpp, 2 for TTL)	2	-
K240	Movement type conversion	0	-
K241	Encoder line per revolution	37500 line/rev.	-
K242^(*)	Position multiplication factor	4	-
K243^(*)	Position division factor	1125	-
M239	Encoder period	Not applicable	-
	Encoder line per revolution	37500 line/rev.	-
M240	Movement type conversion	0	-
M241	Encoder interpolation factor	64	-
KL55	Encoder position increment factor	2400000	-

(*): refer to the formula below

Remark: Tune the current loop regulator to get the best bandwidth without overshoot and tune the position loop regulator and advanced filters to get a good bandwidth with good robustness.

Step 1b: Setting axis using the payload encoder

- Declare payload encoder as a secondary encoder by setting the registers at depth 1 (K77, K79, K240 and K241);
- Save and reboot the controller;
- Check if both motor and payload encoders are counting in the same direction using monitoring ML1 (M1) and ML13 (M13). If this is not the case, modify K68:1 to align the direction of both encoders;
- If K68:1 has been modified, save and reboot the controller;
- Check if the direction corresponding to position increase/decrease is the one required using monitoring ML1 (M1) and ML13 (M13). If this is not the case, modify both K68:0 and K68:1 and apply the AUT = 2 command;
- Save and reboot the controller;
- Check if the relative movements of both encoders are expressed in the same ISO units using monitoring ML1 (M1) and ML13 (M13). If this is not the case, apply the following formula to the motor encoder):

- with indirect linear:

$$\frac{K243:0.<\text{axis}>}{K242:0.<\text{axis}>} = \frac{K241:0.<\text{axis}>^2}{\text{ratio} \times 10^9} = \frac{37500^2}{5 \times 10^{-3} \times 10^9} = \frac{1406250000}{5000000} = \frac{1125}{4}$$

with ratio being the ratio of linear motion per rotary movement [m/turn]

- Save and reboot the controller.

After this step, the registers should look like this:

	Characteristics	TTL encoder (rotary)	1 Vpp encoder (linear)
	Encoder period ou line/rev.	37500 line/rev.	20 µm
	Interpolation factor	64	8192
	Encoder declaration in AccurET	Main	Secondary
Depth	Encoder register depth	0	1
K68	Encoder reading way inversion	0	1
K77	Encoder interpolation shift value	0	3
K79	Encoder type (0 for 1 Vpp, 2 for TTL)	2	0
K240	Movement type conversion	0	0
K241	Encoder period	Not applicable	20000 nm
	Encoder line per revolution	37500 line/rev.	Not applicable
K242	Position multiplication factor	4	1
K243	Position division factor	1125	1
M239	Encoder period	Not applicable	20000 nm
	Encoder line per revolution	37500 line/rev.	Not applicable
M240	Movement type conversion	0	0
M241	Encoder interpolation factor	64	8192
KL55	Encoder position increment factor	2400000	Not applicable

The single encoder feedback is now set on both encoders, with the relative movements and directions coherent.

Step 2: Configuration of the Permanent Dual Encoder Feedback mode

- Swap depths 0 and 1 for all feedback registers listed above (depth 0 goes to depth 1 and vice-versa), with the exception to KL55. If the rotary encoder is an EnDat2.2, the user must select an initialization with K90≠0 and set accordingly the related parameters (refer to [S7.7](#) for more information about the initialization).
- Set K76 parameter according to the Dual Encoder Feedback mode of operation selected (in this case the Permanent Dual Encoder Feedback)
- Save and reboot the controller

Remark: The PID gains of position loop have changed their ISO value. This can lead to an unsafe situation when powering the motor because the position loop can become unstable with the new gains. Furthermore, this change of PID gains might also lead to other errors, such as "Tracking Error" or "Over Speed" because parameters K30 and K31 might have very small values. Therefore, it is very important to re-tune the PID gains of the position loop before executing the next steps.

After this step, the registers should look like this:

	Characteristics	Payload encoder	Motor encoder
	Encoder interface	1 Vpp encoder	TTL encoder
	Encoder period or line/rev.	20 µm	37500 line/rev.
	Interpolation factor	8192	64
	Encoder declaration in AccurET	Main	Secondary
Depth	Encoder register depth	0	1
K68	Encoder reading way inversion	1	0

	Characteristics	Payload encoder	Motor encoder
K76	Dual Encoder Feedback configuration	1	0
K77	Encoder interpolation shift value	3	0
K79	Encoder type (0 for 1 Vpp, 2 for TTL)	0	2
K240	Movement type conversion	0	0
K241	Encoder period	20000 nm	Not applicable
	Encoder line per revolution	Not applicable	37500 line/rev.
K242	Position multiplication factor	1	4
K243	Position division factor	1	1125
M239	Encoder period	20000 nm	Not applicable
	Encoder line per revolution	Not applicable	37500 line/rev.
M240	Movement type conversion	0	0
M241	Encoder interpolation factor	8192	64

- Select the correct homing method associated to the payload encoder and set carefully all indexation registers K40, KL41, KL42 and KL45 (HMODE, HSDP, HACC, HOFFS, etc.).
- Tune the parameter KF446 used to adjust the accuracy convergence factor. The ACF (Accuracy Convergence Factor) command is an alias of this parameter and uses the same syntax.

7.3.7.5 Axes setting for Intermittent Dual Encoder Feedback mode

Step 1a: Setting axis in single encoder feedback mode

- Perform a drive setting with the motor encoder using ComET's 'Drive setting' tool;
- Save the registers to the controller.

Step 1b: Setting axis using the payload encoder

- Set the payload encoder on depth 1 (K77, K79, K240 and K241) and check that the position is increasing in the same direction as the motor position feedback. If not change K68:1 and test again;
- Adjust K242 and K243 to have the same value in ISO units when moving the position (ML1 and ML13). Normally only depth 1 must be adjusted, but it could also be necessary to modify depth 0;
- Find the value to compensate the offset between ML1 and ML13. This value must be used with the OFFSETSEC command every time the controller is booted.

Step 2: Configuration of the Intermittent Dual Encoder Feedback mode

- It is now possible to enable the Intermittent Dual Encoder Feedback mode with K76:1;
- Save and reboot the controller, so the new configuration is taken into account;
- After having done the homing, it is possible to use the VALPOS command to activate the mode.

7.3.7.6 Other features

- **Commutation look-up table for the motor**
 - For the Permanent Dual Encoder Feedback mode, KL55 must be set with the configuration of the motor encoder (K241:1 and M241:1);
 - For the Intermittent Dual Encoder Feedback mode, KL55 must be set with the configuration of the motor encoder (K241:0 and M241:0).
- **Initialization process**
 - The position used for the different initialization (INI) processes is the motor position feedback (motor encoder). All initialization methods are possible with Dual Encoder Feedback configuration;
 - For all Dual Encoder Feedback modes, a phasing (K90!=0) is necessary (also necessary if an EnDat 2.2 encoder is on the motor). If the rotary encoder is an EnDat2.2, the user must select an initialization different from 0 (K90) and set accordingly the related parameters (refer to [§7.7](#) for more information about the initialization).
- **Homing process**
 - The Dual Encoder Feedback algorithms support all types of existing homing modes;
 - In both modes of operation (Permanent and Intermittent Dual Encoder Feedback), the homing is performed using the payload encoder only and not the motor encoder. When the payload encoder is an EnDat 2.2, no homing is required.

- **Software limits**

- In the Permanent Dual Encoder Feedback mode, for mode K36 =1 (refer to [§7.4.1](#)), the functionality must be applied on the payload position (ML104) and not to the corrected ML1 position;
- In the Intermittent Dual Encoder Feedback, for mode K36 =1 (refer to [§7.4.1](#)), the functionality must be applied on the motor position (ML114) and not to the corrected ML1 position.

- **Error management**

- The **K79 BAD VALUE** error (M64=40) is set at start-up if:
- K76 has a bad value regarding the Dual Encoder Feedback configuration
 - K76 is different than 0 and bit#0 of K302 is set (Force Control mode selected)
 - K79:0 is not coherent with the K76 value (bad payload encoder selection)
 - K79:1 is not coherent with the K76 value (bad motor encoder selection)
 - The **TRACKING DUAL FDB** error (M64=79) is raised when:
 - the difference between motor and payload positions (MF110) is greater than K30:2. This error is available for both Permanent and Intermittent modes.
 - the difference between motor and reference positions (MF112) is greater than K30:3. This error is only available in Intermittent mode.

7.3.7.7 Restrictions

Here are some restrictions when using the Dual Encoder Feedback feature:

- The Dual Encoder Feedback can only be used with 1Vpp, TTL and EnDat 2.2 encoders. Due to the hardware interface on the encoder connector, it is not possible to use simultaneously the encoder's limit switch and the TTL reference mark. It is recommended to rather connect the encoder's limit switch. However, if the limit switch is connected to a digital input (DIN) which uses another connector, then the TTL reference mark can also be interfaced.
- It is not possible to use incremental EnDat 2.2 with Dual Encoder Feedback feature.
- As the Force Control feature cannot be used with an indirect axis, the **K79 BAD VALUE** error (M64=40) will occur if the user enables the Dual Encoder Feedback and Force Control features.
- For better noise immunity, it is recommended to use two separate encoder cables sharing only the same connector at the controller side. If this is not possible, tests must be performed to ensure that disturbances do not impair system performance.
- As this feature is time consuming, it is important to limit the additional options that are enabled (e.g. Scale Mapping, Stage Mapping, Encoder Stretch, Gain Scheduling, motion delay, etc.). Use the "Processor load" tool in ComET to evaluate the execution timings.

7.3.8 No encoder and no possibility to make a power on (K79=98)

This is the default value of the parameter K79. When K79=98, no encoder is connected, no error can occur and therefore, it is not possible to make a power on. If the command PWR=1 is sent with K79=98, the controller enters in **NO PWR ON** error (M64=72). It is similar for the AUT and INI commands.

7.3.9 No encoder (K79=99)

If the user wants to connect a motor without using any encoder, K79=99 must be selected **at all depths**. In this case, no information, such as M7 monitoring for example, will be available. Refer to [§7.14](#) for more information about the stepper in open loop.

7.3.10 Position factors for EDI package

The parameters below are only used by the EDI package to calculate the position unit with indirect encoder.

K	Name	Comment
K242	Position multiplication factor	Position multiplication factor (is used to calculate the position unit with indirect single encoder and indirect dual encoder (depth 0: main, depth 1: secondary, depth 2: auxiliary))

K	Name	Comment
K243	Position division factor	Position division factor (is used to calculate the position unit with indirect single encoder and indirect dual encoder) (depth 0: main, depth 1: secondary, depth 2: auxiliary)

In ComET, all the quantities representing a position given in ISO unit can be multiplied by a factor corresponding to the value of the parameter K242 and divided by a factor corresponding to the value of the parameter K243. For the Dual Encoder Feedback feature, the parameters K242 and K243 are linked to the payload encoder for depth 0 and to the motor encoder for depth 1 (refer to [§7.3.7](#) for more information). In that way, it is possible to make the position scale bigger or smaller. The other ISO quantities (speed, acceleration...) will be adjusted accordingly. It is really interesting for an indirect measurement of position (for example if the user wants to measure the position on a linear movement which is generated by a rotary motor including the encoder).

7.3.11 Encoder monitorings

The monitorings **M40**, **M41** and **M43** allows the user to measure the signal values given by the **analog** position encoder. The **SINENC** (**SINe ENCoder**), **COSENC** (**COSine ENCoder**) and **AMPENC** (**AMPlitude ENCoder**) commands are alias of these monitorings and use the same syntax.

M	Alias	Name	Comment
M40	SINENC	Analog encoder sine signal after correction	Encoder sine signal value. Depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder.
M41	COSENC	Analog encoder cosine signal after correction	Encoder cosine signal value. Depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder.
M43	AMPENC	Analog encoder amplitude signal	Analog encoder $\sin^2 + \cos^2$. Depth 0 for principal encoder, depth 1 for secondary encoder and depth 2 for auxiliary encoder.

The conversion formula to know the measured encoder signal value in ISO units is:

For monitorings M40 and M41	For monitoring M43
$\text{Encoder value } [\hat{V}] = \frac{\text{Encoder value } [inc]}{4096 \cdot 0.74}$	$M43 [\hat{V}] = \frac{\sqrt{M43[inc]}}{2048 \cdot 0.74}$

Remark: To convert ISO units ([m/s], f.e.) and ETEL units ([upi], f.e), refer to [§22](#).

K66=4 allows the user to display on the scope the amplitude of the encoder's analog signals (refer to [§7.13](#) for more information).

7.4 Protection parameters - IMPORTANT

The registers described below are set only once. Generally, they are set automatically by ComET during the setting process. These parameters are added to the existing material protections (fuses, mechanical end stops, etc) and protect the controller as well as the machine. There are three types of protection parameters:

- **Movement limits**
- **Current limits**
- **General protection between the motor and the rest of the machine.**

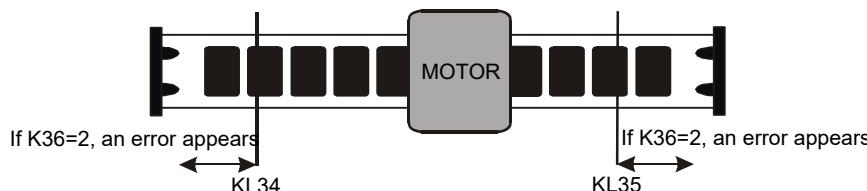
7.4.1 Movements limits

- The movement of the motor can be limited by the parameters KL34 and KL35. The parameter K36 enables or disables these software limits. Depending on the value of the parameter K36, an error may appear if the motor goes over the value of parameters KL34 and KL35. The **MODESL** (**MODE Software Limit**), **MINSL** (**MINimum Software Limit**) and **MAXSL** (**MAXimum Software Limit**) commands are alias of the parameters K36, KL34 and KL35 and use the same syntax.
- It is not possible to enable with K36 the software limits when the MMD mode is greater or equal to 17. In this case, the **ERR SOFT LIMIT** error (M64=80) is raised. When the error is set, K36 is forced to 0.

- Parameters K30 and K31 **switch automatically off the power** of the motor in case of position error or overspeed compared to the values contained in parameters K30 and K31. In that case the controller **switches in error mode**, lights a red error LED and displays a message to identify the error.

Valid for	K	Alias	Name	Value	bit#	Comment	Units
All motors	K30	-	Motor position tracking error limit	-		Absolute tracking error limit. When the tracking error (M2) is greater than K30:0, the controller raises the TRACKING ERROR error M64=23. In Gantry mode (C245>0), when the difference between master and slave positions (ML2) is greater than K30:1:0, the controller raises the GANTRY TRACKING error M64=119. In Dual Encoder Feedback (K76:0≠0 or K76:1≠0), when the difference between motor and payload positions (MF110) is greater than K30:2, the controller raises the TRACKING DUAL FDB error M64=79. In Intermittent Dual Encoder Feedback (K76:1≠0), when the difference between motor and reference positions (MF112) is greater than K30:3, the controller raises the TRACKING DUAL FDB error M64=79.	[dpi] [rdpi]
						When the speed is > K31, the controller generates an OVER SPEED error (M64=24).	[dsi] [rdsi]
	K36	MODESL	Motor position limitation mode	1 2 4	0 1 2	Use of KL34 and KL35 as limit on the target (MVE, RMVE, STE_ABS, STE_ADD, STE_SUB, STA and STI) the motor can reach. If the motor reaches these limits, no error will be generated. Used with K61=1. Not available in interpolation mode (ITP).	-
						Use of KL34 and KL35 as limit on the current position. If the motor reaches these limits, it generates an error (M64=65). These limits are tested every MLTI but only if a homing has been previously done. Available for all values of K61 and ITP.	-
						Use of KL34 and KL35 as limit on the target to generate an error (M64=66) when the movement starts (MVE, RMVE, STE_ABS, STE_ADD, STE_SUB, STA and STI). Used with K61=1. Not available in interpolation mode (ITP).	-
	KL34	MINSL	Minimum software position limit.	-		Depending on the values of K36 and K61 parameters, the motor cannot go lower than parameter KL34.	[upi] [rupi]
	KL35	MAXSL	Maximum software position limit	-		Depending on the values of K36 and K61 parameters, the motor cannot go higher than parameter KL35.	[upi] [rupi]
	K145	-	Search limit stroke mode selection (SLS mode)	0 1 2 3		SLS on mechanical end stop, positive movement. SLS on mechanical end stop, negative movement. SLS on limit switch, with positive movement. SLS on limit switch, with negative movement.	-
Rotary motors only	KL27	-	Maximum position value	-	-	Defines the maximum position range limit in rotative modes (MMD > 16). Effective on ML6 and ML7 only.	[upi] [rupi]

Example:



- Remark:** If K145=0 or 1, parameters KL41, KL42, KL43, KF44 and KL47 are taken into account (refer to [§7.9](#) for more information).
 If K145=2 or 3, parameters KL41, KL42, KL47 and K58 are taken into account (refer to [§7.9](#) for more information).
 The parameter K58 can be used as a protection parameter (refer to [§7.9.1](#) for more information).
 If K36=2, move the motor inside the stroke delimited by KL34 and KL35 otherwise the **OUT OF STROKE** error (M64=65) cannot be reset.

The **SLS** (Search Limit Stroke) command is:

Command	Comments
SLS.<axis>	Searches limit stroke according to K145 (search limit stroke mode) and returns the limit position in ML36 and ML37. KL47 is taken into account by SLS command but not in ML36 and ML37 setting. Not available in ITP mode.

- Remark:** The **MINHL** (**MIN**imum **HARDWARE** **LIMIT**) and **MAXHL** (**MAX**imum **HARDWARE** **LIMIT**) commands

are alias of the monitorings ML36 and ML37 respectively and use the same syntax
A homing must be done (with IND command) before sending the SLS command.

7.4.2 Current limits

An overcurrent in the motor phases can destroy it. Parameters **KF83**, **KF84** and **KF85** help to avoid this problem. The parameter **KF60** limits the current reference (theoretical) at the output of the position regulator. Thus, the current in the motor will be limited too.

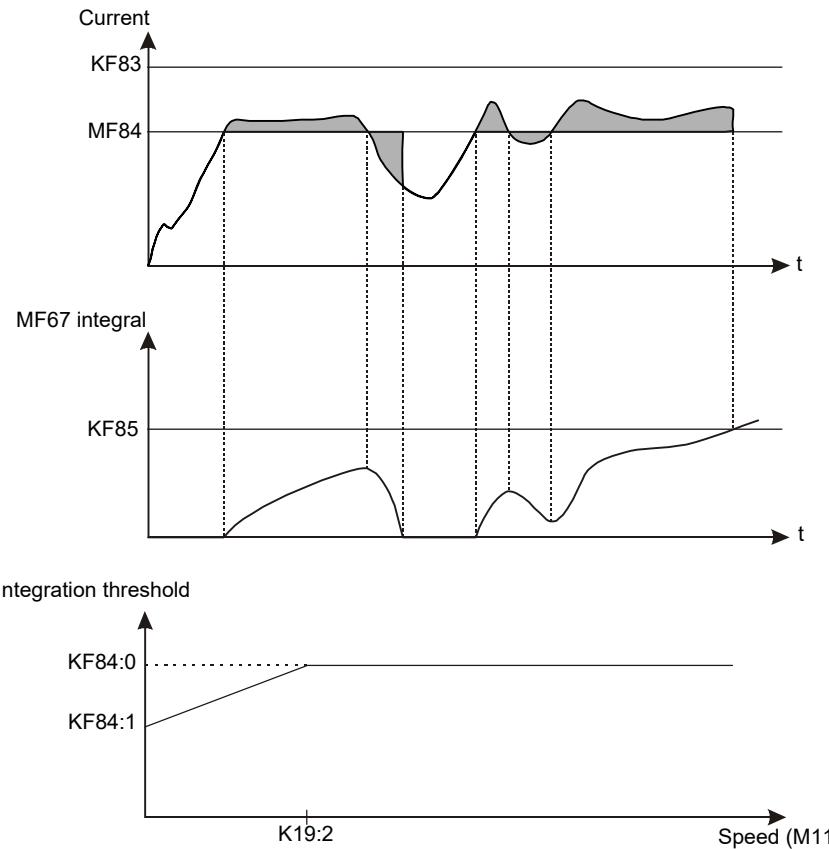
K	Name	Comment	Units
KF83	Current loop software overcurrent limit	If the current in the phases is > KF83, the OVER CURRENT error appears (M64=2 or 3).	[ci]
KF84	I2t rms current limit	The integration starts when the square of motor current is > KF84 (depth 0: motor is moving, depth 1: motor in stall). The speed limit for the changing mode is defined by K19:2.	-
KF85	I2t time limit	If the integral value is > KF85, the I2T ERROR error (M64=4) appears.	-
KF60	Current limit	Max. current reference at the position regulator output (will limit the current in the motor).	[ci]
K19	Speed level for I2t motor, gain scheduling KF4,...	Depth 2 for I2t motor, speed level from which the motor changes from stall to moving mode; value in the motor's data sheet	[dsj]

M	Alias	Name	Comment	Units
MF67	-	I2t motor current value	When monitoring MF67 is greater than parameter KF85, the controller generates an I2T ERR CTRL error (M64=4).	-
M82 MF82	IMAXC	Controller maximum current	Maximum current [A]=M82/100. This theoretically maximum value is used for calculations only.	[A]*100
MF84	-	I2 motor rms current limit	Gives the I2 motor rms current limit for I2t integration according to the speed (M11)	-
MF167	-	I2t controller current value	When monitoring MF167 is greater than controller's value, the controller generates an I2T ERR CTRL error (M64=11).	-

The parameter KF83 protects the motor from an **instantaneous overcurrent**. If the motor current gets higher than the value of parameter KF83, the **OVER CURRENT1** error (M64=2) or **OVER CURRENT3** error (M64=3) occurs.

Parameters KF84 and KF85 protect the motor from a **too high current during a too long time** which could raise the temperature high enough to burn the motor. When the instantaneous current value increases over the limit value of the monitoring MF84, an integrator is activated. As long as the current stays over the value of the monitoring MF84, it is integrated, but when it passes under the threshold, the integrator empties progressively. If the integral value is higher than the one of the parameter KF85, the controller displays an **I2T ERROR** error (M64=4).

Depending on the motor speed, the integration threshold can be different to protect the motor phases in stall condition. Below the speed value equal to M11 < K19:2, the value used for the integration threshold will be reduced proportionally to the speed to reach the stop stall value (KF84:1). It is not possible to have KF84:1 greater than KF84:0. In this case, the monitoring MF84 is limited to KF84:0 when the speed is lower than K19:2. Reaching the current in monitoring **M82 is not possible**; this value is the theoretical maximum value which can be measured for internal calculations only! (refer to [§22.2](#)). The **IMAXC (I MAXimum Controller)** command is an alias of the monitoring M82 (MF82) and uses the same syntax.



Remark: It may happen that, during a periodic movement (machine cycle), the monitoring MF84 may temporarily be overcome, and the integrator may not entirely be empty at the end of the cycle. In that case, after several cycles, an **I2T ERROR** error (M64=4) is displayed. The cycle is then **unstable**.

With a current greater than the continuous controller's current (called 'Max. full load current' in the corresponding 'Hardware Manual'), the heating times constants for the controllers are faster than for the motor, that is why the I2T ERR CTRL error (M64=11) generated at the controller level can occur BEFORE the motor I2T ERROR error (M64=4).

The value of the parameters KF83, KF84 and KF85 have to be calculated as follows.

Motor:

I_m peak:	Motor peak current [A] (this is also the max current of the application).
I_m continuous:	Motor continuous current [A].
I_m stall:	Motor stall current [A].
t:	Maximum time at I_m peak before an i^2t error [s].
I_{over} current:	Ultimate motor current value, for a motor OVER CURRENT error [A] ($= I_m$ peak [A] + ~20%)

Controller:

I_{max} controller:	Maximum current of controller [A]=M82/100
-----------------------	---

In the equations hereafter, the controller parameters (KF83, KF84, KF85 and M82) are in increment and the motor and controller specifications (I_m peak, I_m continuous,...) in ISO value. Therefore the equations are only valid for increments.

Parameter KF83: motor overcurrent limit:

$$KF83 = \frac{I_{over} \text{ current} \cdot 32768}{M82 / 100}$$

Parameter KF84: i^2t rms current level:

$$KF84:0 = \left(\frac{I_m \text{ continuous}}{M82/100} \cdot 100 \right)^2 \cdot 0.8192 \quad KF84:1 = \left(\frac{I_m \text{ stall}}{M82/100} \cdot 100 \right)^2 \cdot 0.8192$$

Remark: If a **2-phase** linear motor performs a **back and forth movement with very short strokes**, only one phase will be used. Thus, thermal load will be concentrated on the half of the motor surface. Take parameter KF84, (value obtained with formula above) and **divide it by 2: KF84 short_stroke=KF84 / 2.**

Parameter KF85: i^2t integration limit (energy):

$$KF85 = 9830 \cdot t \cdot \left(\left(\frac{I_m \text{ peak}}{M82/100} \cdot 100 \right)^2 - \left(\frac{I_m \text{ continuous}}{M82/100} \cdot 100 \right)^2 \right)$$

Example:

Motor:	$I_m \text{ peak}$	= 14 A
	$I_m \text{ continuous}$	= 7.2 A
	$I_m \text{ stall}$	= 6 A
	t	= 5 s

Controller:	$I_{\text{over current}}$	= $I_m \text{ peak} + \sim 20\% = 16.8 \text{ A}$
	EA-P2M-400-10/20A	$I_{\text{max controller}} = (M82 / 100) = 34 \text{ A}$

$$KF83 = \frac{16.8 \cdot 32768}{34} = 16191$$

$$KF84:0 = \left(\frac{7.2}{34} \cdot 100 \right)^2 \cdot 0.8192 = 367 \quad KF84:1 = \left(\frac{6}{34} \cdot 100 \right)^2 \cdot 0.8192 = 255$$

$$KF85 = 9830 \cdot 5 \cdot \left(\left(\frac{14}{34} \cdot 100 \right)^2 - \left(\frac{7.2}{34} \cdot 100 \right)^2 \right) = 61292941$$

7.4.2.1 Power consumption

A calculation allows the user to know the consumption of the machine's power. This calculation is made on 64-bit registers and the time (number of PLTI, refer to [§1.](#)) is given in order to let the application calculated the energy used. The **SCI** command allows the user to start and stop the calculation for the 'integrator' $(I_{PH1})^2 + (I_{PH2})^2 + (I_{PH3})^2$.

Command	<P1>	Comments
SCI.<axis>=<P1>	0 1	Stops the acquisition of the calculation of I2 sum Clears the monitoring M180 and ML181 and starts the acquisition of I2 sum

M	Name	Comment
M180	I2 sum PLTI counter	Represents the number of PLTI interrupt during the calculation of the integrator
ML181	I2 sum	Represents the 64-bit integrator

- For one-phase motor, ML181 is the sum of $(I_{PH1})^2$ and can be compared to MF30 or MF31.
- For two-phase motor, ML181 is the sum of $(I_{PH1})^2 + (I_{PH3})^2$ and can be compared to MF30 or MF31.
- For three-phase motor, ML181 is the sum of $(I_{PH1})^2 + (I_{PH2})^2 + (I_{PH3})^2$ and cannot be compared to MF30 or MF31.

7.4.3 General protection

7.4.3.1 Protection signals on DIN

The parameter **KL305** is a mask of digital input(s) that allows the user to enable to power on the axis. The mask allows to set a detection on any DIN (only one DIN must be used) of both axes, either on an ON state or an OFF state. When the axis is in power ON, and if the DIN state is not correct accordingly to the mask setting (KL305), the controller enters in **POWER OFF/ON** error mode (M64=26). By default, the mask KL305 is 0 and the functionality is deactivated.

K	Name	Comment										
KL305	Input state mask to power ON	Gives the input state mask to power ON the motor (bit#0-31: state 1, bit#32-63: state 0, depth 0: CDIN of both axes)										

Bit#63 to 52	bit#51	bit#50	Bit#49 to 45	bit#44	bit#43	bit#42	bit#41	bit#40	Bit#39 to 35	bit#34	bit#33	bit#32
All 0	DIN10 of axis 1 (state 0)	DIN9 of axis 1 (state 0)	All 0	DIN3 of axis 1 (state 0)	DIN2 of axis 1 (state 0)	DIN1 of axis 1 (state 0)	DIN10 of axis 0 (state 0)	DIN9 of axis 0 (state 0)	All 0	DIN3 of axis 0 (state 0)	DIN2 of axis 0 (state 0)	DIN1 of axis 0 (state 0)

Bit#31 to 20	bit#19	bit#18	Bit#17 to 13	bit#12	bit#11	bit#10	bit#9	bit#8	Bit#7 to 3	bit#2	bit#1	bit#0
All 0	DIN10 of axis 1 (state 1)	DIN9 of axis 1 (state 1)	All 0	DIN3 of axis 1 (state 1)	DIN2 of axis 1 (state 1)	DIN1 of axis 1 (state 1)	DIN10 of axis 0 (state 1)	DIN9 of axis 0 (state 1)	All 0	DIN3 of axis 0 (state 1)	DIN2 of axis 0 (state 1)	DIN1 of axis 0 (state 1)

Remark: State 1 means that the DIN associated to the function will react on a '1' state.
State 0 means that the DIN associated to the function will react on a '0' state.

7.4.3.2 Current and speed limitation on DIN

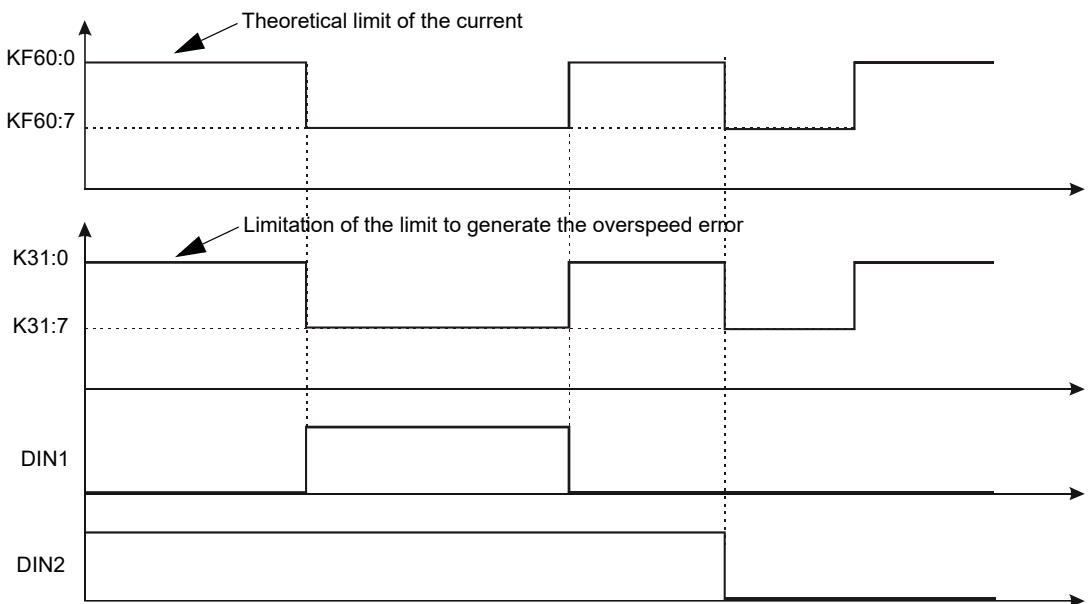
The parameter **KL302** allows the user to enable the limitation of parameters KF60 and K31 according to the state of the digital inputs. If the mask is not set, the function is deactivated. This mask lets the possibility to select a DIN on one of the two axes of the AccurET.

K	Name	Comment										
KL302	Input state mask for speed and current limitation	Input state mask for limitation KF60/K31 (bit#0-31: state 1, bit#32-63: state 0, depth 0: CDIN of both axes)										

- When the mask is set
 - If at least one of the conditions defined by KL302 is true, depth 7 of parameter KF60 (KF60:7) is kept as the limitation of the current and depth 7 of K31 (K31:7) as the value of the overspeed.
 - If none of the conditions defined by KL302 is true, depth 0 of parameter KF60 (KF60:0) is kept as the limitation of the current and depth 0 of K31 (K31:0) as the value of the overspeed.
 - If KL302=0, depth 0 of parameter KF60 (KF60:0) is kept as the limitation of the current and depth 0 of K31 (K31:0) as the value of the overspeed.
- When the mask is not set
 - The limitation is not activated, depth 0 of parameter KF60 (KF60:0) is kept as the limitation of the current and depth 0 of K31 (K31:0) as the value of the overspeed.

Example:

Here the limitation is activated if DIN1=1 or if DIN2=0:



Remark: Refer to [§8.5](#) for more information about the digital inputs.

7.4.3.3 Protection signals on DOUT / FDOUT

When a motor is integrated in a complex machine and an error is detected in the controller, it is important to send a message about it to the rest of the machine so that the other elements can adequately react. This is possible with the digital outputs and fast digital outputs. The parameter **K357** allows the user to set or reset one or two digital outputs when an error occurs. For example, DOUT1 can be connected to a relay which short circuits the motor phases in case of error, making a **magnetic brake**. The output(s) to be set or reset in case of error, are chosen via the binary value of the parameter K357. This is also the case for DOUT/FDOUT reserved for the Fast Triggers (K359 and C359).

K	Name	Value	Comment
K357	Mask of the DOUT/ FDOUT if error	-	Mask of the digital output (DOUT) or fast digital output (FDOUT) that must be set or reset when the controller is in error. bit#0-15: state1, bit#16-31: state0, depth 0: DOUT, depth 1:FDOUT

Remark: Refer to [§8.5](#) for more information about the digital inputs and outputs.

After the RST command, the signal on the output takes back its initial value.

In case of error, it is not allowed to use the parameter K357 to activate an external relay to shortcut motor phases if C10=1 or 2 (because the speed loop will be activated during the error).

7.4.3.4 Motor temperature check

It is possible to connect the temperature sensor of the motor to a digital input of the controller to know its internal temperature.

K	Name	Comment
K304	Motor overtemperature protection	Defines a mask and the status on the digital input which will be tested for the motor overtemperature protection. bit#0-15: state 1, bit#16-31: state 0. Depth 0: DIN

bit#	31--26	25	24	23--19	18	17	16	15--10	9	8	7---3	2	1	0
Hex. value	-	2000000	1000000	-	40000	20000	10000	-	200	100	-	4	2	1
DIN # on AccurET 400/ 600	-	10	9	-	3	2	1	-	10	9	-	3	2	1
DIN # on AccurET 48 / 300 VHP 48 / 100	-	10	9	-	-	2	1	-	10	9	-	-	2	1
State	0							1						

Example:

```
K304:0=0x1;           // the error occurs if the DIN1=1
K304:0=0x10000;      // the error occurs if the DIN1=0
```

Remark: The digital inputs are standard inputs then be careful when choosing the type of temperature sensor (digital only). Refer to the '**Hardware Manual**' for more information about the inputs.

7.4.3.5 Vpower DC bus voltage

Parameters **K146**, **K147**, **K148** and **K149** are used to give the errors and warnings limits values of the input voltage.

K	Alias	Name	Value	Comment	Units
K146	UVWL	Vpower undervoltage warning	0 > 0	Disables test on Vpower (not possible to disable on VHP48, VHP100 and ZxT) Warning activated: Vpower level is tested. If Vpower[V]*100<K146, the controller generates a W UNDER VOLTAGE warning (M66=10)	[V]*100
K147	UVEL	Vpower undervoltage error	0 > 0	Disables test on Vpower (not possible to disable on VHP48, VHP100 and ZxT) Error activated: Vpower level is tested. If Vpower[V]*100<K147 the controller generates an UNDER VOLTAGE error (M64=9)	[V]*100
K148	OVWL	Vpower overvoltage warning	0 > 0	Disables test on Vpower Warning activated: Vpower level is tested. If Vpower[V]*100>K148, the controller generates a W OVER VOLTAGE warning (M66=4)	[V]*100
K149	OVEL	Vpower overvoltage error	0 > 0	Disables test on Vpower Error activated: Vpower level is tested. If Vpower[V]*100 > K149, the controller generates an OVER VOLTAGE error (M64=6)	[V]*100

The **UVWL** (UnderVoltage Warning Limit), **UVEL** (UnderVoltage Error Limit), **OVWL** (OverVoltage Warning Limit) and **OVEL** (OverVoltage Error Limit) commands are alias of the parameters K146, K147, K148 and K149 respectively and use the same syntax. **The undervoltage warnings and errors can occur only if the motor is powered on.**

The monitoring **M91** is used to indicate the DC input voltage level (Vpower).

M	Name	Comment	Units
M91	Vpower measurement	Gives the DC input voltage level (Vpower): Vpower[V]=M91/100	[V]*100

7.4.3.6 Fuse check

It is also possible to monitor the state of some fuses with the monitoring **M140**.

M	Name	Value	Comment
M140	Mask for fuse control	0 1 2	Fuse is not broken. The encoder fuse is broken. The encoder is no more powered. The DOUTs fuse is broken. The DOUTs are no more powered.

Remark: If the encoder fuse is broken, the **ENCODER FUSE KO** error (M64=35) will appear.

If the DOUTs fuse is broken, the **DOUT FUSE KO** error (M64=14) will appear.

7.4.3.7 Miscellaneous protections

The **bit#1** of the parameter **K33** allows the user to activate the **SWITCH LIMIT** error (M64=30) when the limit switches are reached during a movement (except during the IND or SLS command). This error is generated only if the controller is in 'Power On'.

- To reset the error **when K61=1** (refer to [§7.6](#)), the following procedure is recommended:
 - Reset the error (with RST.<axis> command)
 - Send the command: PWR.<axis>=1
 - Move the motor out of the limit switch (with MVE.<axis> command).

- To reset the error when **K61=1** or **K63=1** (refer to [§7.6](#)), the following procedure is recommended:
 - Clear bit#1 of K33 (otherwise during the next 'Power On', the same error will come back)
 - Reset the error (with RST.<axis> command)
 - Send the command: PWR.<axis>=1
 - Move the motor out of the limit switch (with MVE.<axis> command)
 - Set to 1 the bit#1 of parameter K33

K	Name	Value	bit#	Comment
K33	Input mode	2	1	Enables the use of limit switches

Remark: If K33=2, the choice of the limit switches is set by the parameter K58.

The parameter K58 can be used for the homing as well as for axis protection. However it is not possible to manage two different limit switches (one for axis protection, one for homing) at the same time.

Refer to [§8.5](#) for more information about the digital inputs.

7.5 Errors and warnings handling

Errors may be detected by the controller if a protection limit is exceeded or if a hardware failure occurs. Each error corresponds to a value of the monitoring **M64**. It shows which error occurred to handle it adequately. Some limits will give a **warning** before the error appears to allow the user to solve the problem. Each warning corresponds to a value of the monitoring **M66**. The **ERRCD** (ERRor CoDe) and **WARCD** (WARning CoDe) commands are alias of these monitorings and use the same syntax.

When an error is identified, the **errors reference list** ([§21](#)) is used to allow the user to identify the cause of the problem. Refer to [§20](#) for the **warning references list**.

M	Alias	Name	Comment
M64	ERRCD	Error code	Number of the first occurred error
M66	WARCD	Warning code	Number of the occurred warning with highest priority (refer to §7.5.4)

Remark: Refer to [§7.5.4](#) for more information about the warning management.

7.5.1 Errors reset: RST and RSD

Two different commands can be used to reset the errors:

- RST** command (ReSeT) **resets most of the errors** that can occur. It is a software reset.
- RSD** command (ReSet Drive) resets the hardware board. It is a hardware reset which consists in switching off and on the controller. Therefore, if an autorun sequence is present (function `autostart()`), it will start again.

Remark: Refer to [§10](#), if the RSD and RST command are used when the error propagation is activated. In a sequence, these commands are generally written in the function `errhandler()` which is the function in which the program execution goes on in case of error.

Command	Comment
RST.<axis>[=<P1>]	Resets the error flags of the controller (bit#10 of SD1). It also exits the 'torque off mode' in stage protection mode with the parameter <code><P1>=123</code> (refer to §9)
RSD.<axis>=255	Hardware resets of the controllers

When one of the condition defined by the parameters **K303** is true, a hardware reset is activated. It has the same function as a RSD command.

K	Name	Comment
K303	Input edge mask for RSD	Defines the mask of the digital inputs (rising & falling edge) to reset the controller like a RSD command. bit#0-15: rising edge. bit#16-31: falling edge. Depth 0: DIN, depth 1: FDIN

Example:

If during a movement, the position error x_e becomes too big, the controller switches to an error mode. The power is cut off and the ComET software shows the **TRACKING ERROR** error (M64=23). This error can be reset with the RST or RSD command:

- RST . 1** The error is reset, the LED lights green and ComET displays for example **ACCURET READY**, as long as the cause which has produced the error is not there any more. The motor position is kept and the instruction PWR.1=1 is sufficient to reset the motor under control.
- RSD . 1=255** The board is reset and the motor position is erased. A new initialization has to be done to find again the absolute motor position.

7.5.2 Errors propagation on the TransnET

On all AccurETs and UltimET, there is a mechanism of errors propagation running through the TransnET. It brings two main advantages compared to the use of the UltimET's parameters *KL630 and *KL632 (refer to the '**UltimET User's Manual**' for more information about these two parameters).

- Possibility to define 4 independent error groups of axes
- Increased protection due to the fact that the error information runs at a lower software level

Remark: This mechanism works **also** if the stage protection is deactivated.

This mechanism is managed by two parameters:

- The parameter **K141** allows the user to define the group in which the error must be published when an error occurs on the axis:

K	Name	Comment
K141	Error published group mask	Gives the error propagation published group mask: bit#0 for group 1 (by default), bit#1 for group 2, bit#2 for group 3 and bit#3 for group 4.

The monitoring **M141** allows the user to check the value of the error propagation register on the TransnET.

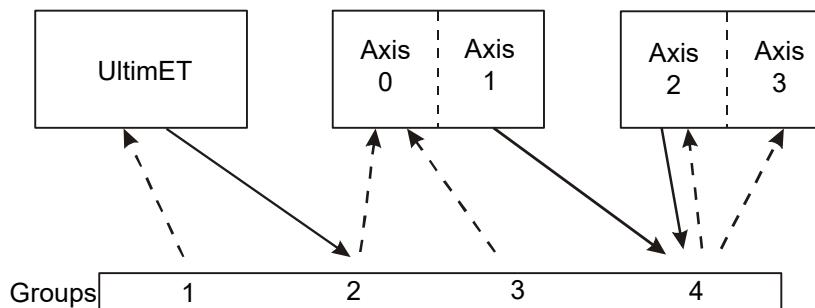
M	Name	Comment
M141	TransnET error groups status	Gives the value of the error propagation register on the TransnET.

- The parameter **K140** allows the user to define which error group the axis belongs to. When the bit corresponding to its group(s) is set to 1 on the TransnET, the axis goes immediately in **EXTERNAL ERROR** error (M64=116):

K	Name	Comment
K140	Error checked group mask	Gives the error propagation checked group mask: bit#0 for group 1 (by default), bit#1 for group 2, bit#2 for group 3 and bit#3 for group 4.

Remark: On the UltimET side, both modes are available: *KL630 and *KL632 or *K140 and *K141 but the one with the parameters *K140 and *K141 must be used in priority.
Parameters K140 and K141 are set to 1 in the AccurET at the end of the drive setting.

It is not possible to power on an axis as long as one device belonging to the same error group is in error. Now a RSD command will create an error on all devices. This command must be followed by the *RST.!=123 when the stage protection is activated. It is not possible to reset only one axis of an error group and perform on it a power on. To do this power on, all axes of the error group must be reset.

Example of use

The setting corresponding to the above-mentioned drawing is as follows:

- Errors put into groups 1 to 4

```
*K141=0x2; //UltimET signals its error in group 2
K141.0=0x0; //Axis 0 does not signal its error in any group
K141.1=0x8; //Axis 1 signals its error in group 4
K141.2=0x8; //Axis 2 signals its error in group 4
K141.3=0x0; //Axis 3 does not signal its error in any group
```

- Errors generated from groups 1 to 4

```
*K140=0x1; //UltimET must enter in error when group 1 has an error
K140.0=0x6; //Axis 0 must enter in error when groups 2 or 3 have an error
K140.1=0x0; //Axis 1 does not watch any group
K140.2=0x8; //Axis 2 must enter in error when group 4 has an error
K140.3=0x8; //Axis 3 must enter in error when group 4 has an error
```

Remark: If the stage protection is activated (C10=1 or 2), when an error occurs on one of the two axes, the second one enters in error too.

7.5.3 Dynamic braking

The dynamic braking is a kind of braking that uses the motor as a generator to transform kinetic energy into heat. Once this feature is enabled, the controller creates a short-circuit between the motor terminals. Then when the axis moves, a current is generated in the motor phases that leads to the energy transformation mainly due to the resistances of the motor. Consequently, the motor brakes.

The performance of this braking mode is very variable and depends on the speed and the characteristic of the motor (resistance and back EMF). In many applications, the more advanced feature "Stage Protection" (refer to [§9](#) for more information) will be more suitable to control the system in case of error. The dynamic braking is typically used to brake the motor when an error arises. However, it can also be used to prevent a sudden movement when powering off an axis, for example with a vertical axis.

Warning: This function is not a certified functional safety feature. This function is managed by software and a voltage is present at the motor's terminal (continuous alternating switching of the transistors (GND - Vpower))! Due to this voltage, it is strictly forbidden to touch the system (controller, cable and motor) as long as the mains is not switched off.

7.5.3.1 Dynamic braking in case of error

The **bit#3** of parameter **K33** is used to enable or disable the braking when an error occurs (refer also to [§7.9](#) for more information about the parameter K33). Once enabled (K33|=0x8), the controller behavior will depend on the error type in case of error. Indeed, some errors will not enable the dynamic braking in order to protect the hardware. Refer to the errors list in [§21](#) to see which ones allow the activation ('brake on') and which ones do not ('brake off').

K	Name	Value	bit#	Comment
K33	Input mode	8	3	Enables the dynamic braking controlled by the transistor in case of error

- If a 'brake off' error type arises, the braking will not be reactivated by a 'brake on' error type. It can be done only after a 'PWR.<axis>=1' or RSD.<axis>=255 command.
- If a 'brake on' error type arises, followed by a 'brake off' one, this second error type will not be displayed but the braking will be deactivated.
- The 'RST' command does not have any effect on the braking mode (in this case, both 'ERROR' and 'POWER ON' LEDs are OFF). The latter can be reset only after a PWR.<axis>=1 or RSD.<axis>=255 or by resetting and setting again bit#3 of the parameter K33. If the braking is activated, a vertical motor will not suddenly fall against the mechanical end stop after the 'RST' command.

7.5.3.2 Dynamic braking when motor in power off

The **bit#2** of parameter **K33** is used to enable or disable the dynamic braking mode when the motor is powered off. Once the bit#2 is set (K33 |= 0x4), the dynamic braking is always activated if the motor is not powered.

Warning: If the bit#2 of K33 is set and saved in Flash memory, as soon as the AccurET boots up, the Dynamic Braking will be enabled and a continuous switching from high and low side of the bridge transistors will occur! A voltage will be therefore immediately present on the motor's terminals and it is strictly forbidden to touch the system (controller, cable and motor) as long as the mains is not switched off!

K	Name	Value	bit#	Comment
K33	Input mode	4	2	Enables the dynamic braking controlled by the transistor in case of power off

Selecting the dynamic braking when powering off the motor automatically activates the dynamic braking mode in case of error (bit#3).

7.5.3.3 General behavior

- During the braking, there is a protection against the overcurrent to protect the controller and the system. The braking is deactivated (power bridge opened) when the current in one phase is bigger than 75% of the software overcurrent error (parameter KF83). The braking is reactivated when the current in all phases is again smaller than this limit.
- During the braking, there is also a protection against the I2t motor to protect the controller and the system. The braking is deactivated when the monitoring MF67 (I2t motor value) is bigger than 75% of the I2t motor error (parameter KF85). The braking is reactivated when the monitoring MF67 is again smaller than this limit.
- During the braking, there is also a protection against the I2t controller to protect the controller and the system. The braking is deactivated when the monitoring MF167 (I2t controller value) is bigger than 75% of the I2t controller error. The braking is reactivated when the monitoring MF167 is again smaller than this limit.
- If the power supply voltage has been removed (even if it is not the reason of the error), caution must be taken before switching on the power voltage again. In fact, the dynamic braking is based on short-circuit principle at the terminals of the motor via the power output transistors of the position controller. To share the amount of current going to the motor, the power output transistors switch alternatively (high side and low side). The losses created by this switching generate a current on the DC bus coming from the power supply.

Depending on the number of position controller connected to the power supply, the length of motor cables,..., this current can be enough to keep continuously the power supply in Inrush mode. If this is the case, the next power on will create an **INRUSH P. SUPPLY** error (M64=7) with the ETEL power supply.

To prevent and avoid it, it is highly recommended to disable the dynamic braking mode (clear bit#3 of K33) before switching ON the power supply voltage, then the error can be reset and the dynamic braking can be re-enabled (set bit#3 of K33) when the inrush process of the power supply is finished (no more **W INRUSH P.SUPPLY** warning, M66=3).

7.5.4 Warning management

Warnings are used to draw the attention of the user to some specific non-critical condition or to alert them when some limit is about to be reached. They are enabled/disabled with parameter **K166** and observed by monitoring **M166** and **M66**. Several warnings can be signaled (active) at the same time. Each one is represented by its own bit in K166 and M166.

Parameter **K166** is a bit field allowing the users to mask (or not) each warning message individually. The default value for this parameter is 0x0 and means no warning is masked. When a bit of K166 is set, the corresponding warning is masked and will not appear in M66, in bit#23 of M63 and in bit#23 of M60.

K	Name	Comment
K166	Warning mask	Gives the bits masking the corresponding warning messages

Example:

If the user wants to mask the «Under Voltage» warning, K166.<axis>=0x400

Monitoring **M166** is a bit field allowing the users to see all signaled warnings. When a warning is present, the bit related to this warning is set to 1. The bit field of M166 is identical to the one of K166. M166 cannot be changed by K166 (a warning cannot be masked in M166).

M	Name	Comment
M166	Warning flag	Indicates the mask of all warning present

Each warning has a fixed relative priority. The bit number of the signaled non-masked warning having highest priority is shown by monitoring **M66** (or **WARCD** alias). The presence of any signaled non-masked warning is indicated by setting bit #23 (warning) in M63. ComET shows the presence of an enabled non-masked warning by putting an yellow circle on top of the 'AccurET' icon in its status bar. If there is simultaneously an error signaled, it is the red circle for the error that has the priority. The bit definitions of M166 and K166 are identical.

Bit# or M66	Displayed message	Value in M166 / K166	Priority
1	W I2T MOTOR	0X2	0 (highest)
2	W OVER TEMPERAT	0x4	2
3	W INRUSH P.SUPPLY	0x8	3
4	W OVER VOLTAGE	0x10	4
5	W ENCODE AMPLITU	0x20	5
6	TRACKING WARNING	0x40	6
8	W DIGIT. HALL	0x100	8
9	WuCONTRO LSYNCHRO	0x200	7
10	W UNDER VOLTAGE	0x400	11
11	W I2T CTRL	0x800	1
12	W POWER RELAY	0x1000	12
13	WPHASING FAILURE	0x2000	13
15	W PWM UPDATE	0x8000	14
17	W ENDAT ENCODER	0x20000	16
19	W SEQUEN BRKPOINT	0x80000	18
24	TRIGGER MISSED	0x1000000	23
25	W TRIG TOO LAT	0x2000000	24
26	SEC ENCO AMPLITUD	0x4000000	25
27	AUX ENCO AMPLITUD	0x8000000	26
28	JERK TIME IGNORED	0x10000000	27
29	TRIGGER CONTINUO	0x20000000	28
30	UPLOAD TOO SLOW	0x40000000	29

Remark: Refer to [§20.](#) for more information about the warning messages.

7.6 Normal reference mode (K61=1)

The parameter **K61** allows the user to set the reference mode (type of input reference given to the controller). The standard reference mode (also called normal reference mode) is set when K61=1 (when K61≠1, an advanced reference mode is selected). This mode is the most frequently used mode (described up to now in this manual). To use it, the user needs to set:

K	Name	Value	Comment
K61	Position reference mode	1	Standard mode with set point generator movement profiles.
		3	Controller controlled by a speed reference defined by parameters K220, K221, K222, K223 and KF224 (refer to §8.2).
		4	Controller controlled by a position reference defined by parameters K220, K221, K223 and KF224 (refer to §8.2).
		36	Like K61=4, but the actual position is kept as a reference when the controller is switched on (refer to §8.2).

- a type of movement defined by parameter K202 (or the MMD command which is an alias of parameter K202). Refer to [§8.3.1.1](#) for more information.
- the references of the movement which are (refer to [§7.12.3](#) for more information):
 - a final position to reach defined by the command MVE.<axis>
 - the maximum speed defined by parameter KL211 (or the SPD command which is an alias of parameter KL211)
 - the maximum acceleration defined by parameter KL212 (or the ACC command which is an alias of parameter KL212)
 - the jerk time defined by parameter K213 (or the JRT command which is an alias of parameter K213) when a S-curve movement is selected

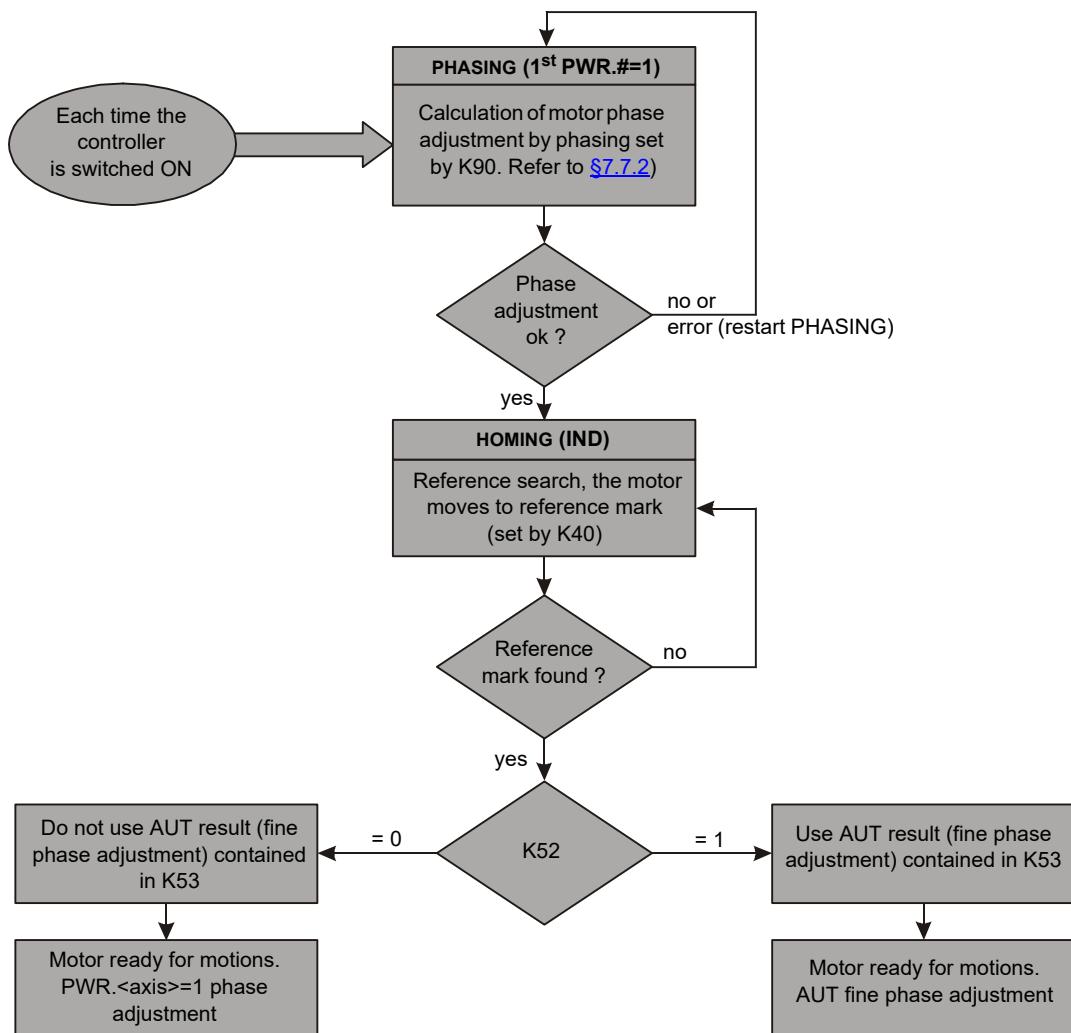
All these references are calculated by the set point generator generating the movement trajectory. Refer to [§8.1.1](#) for more information about the complete diagram of the regulation loop.

Remark: Refer to [§8.2.1](#) for more information about the use of the depth 1, 2 and 3 of the reference parameters.
 Refer to [§8.2](#) for more information about the other values of parameter K61.
 To use the standard state regulator, the parameter K78 must be equal to 0 otherwise it correspond to a stepper mode (refer to [§7.14](#) for more information).

7.7 Initialization

To have a direct drive motor working properly, it has to go through several processes:

- The phasing (parameter K90)
 After the first 'Power On' of the controller, the position of the motor's coils towards the magnets must be defined. It is done with a measurement.
- The homing (parameter K40)
 When the phasing is done, the motor is able to move. The real position of the motor with regard to its own physical environment must then be defined. To do so, a reference must be detected (except for absolute encoder) This reference can be a reference mark (also called index), a home switch, a limit switch or a mechanical end stop (refer to [§7.9](#) for more information).
- Phase shift adjustment
 During the next 'Power On', it is necessary to do a new phasing as the measurements can vary in a range of +/- 10% from a phasing to another (especially with an phasing by pulses). Of course, this would influence the system's performances in the same range. To avoid this problem, a phase shift adjustment must be performed before the first initialization.
 A phase shift adjustment is done during the first 'Power On' of the motor. This command will do a phasing then an homing and will store the coil position towards the magnets and the absolute position. Thus, after each homing we have the same value of phasing.



7.7.1 Phasing basics

7.7.1.1 Phasing use

The motor must be initialized before powering it if an incremental encoder is used. This is the case in most controllers applications. On the other hand, we do not need a phasing if we use a single-phase motor or when we use an absolute EnDat encoder. In all these cases, K90=0.

7.7.1.2 Phasing purpose

The Back-EMF is induced in the motor by the induction of the magnets. For an optimum working of the motor, the current into the coils has to be in phase with the B-EMF.

The phasing procedure determines the position of the motor's coils towards the magnets poles in order to inject the good current shape into the coils (giving the maximum force to the motor). In fact, it avoids calculating the initial position of the pointer in the look-up table of the phases commutation. The look-up table of the phases commutation is a continuity of points making a sine, used to inject the currents into the coils in function of the position of the magnets.

The phasing is done at each first 'Power On' of the motor or after each INI command.

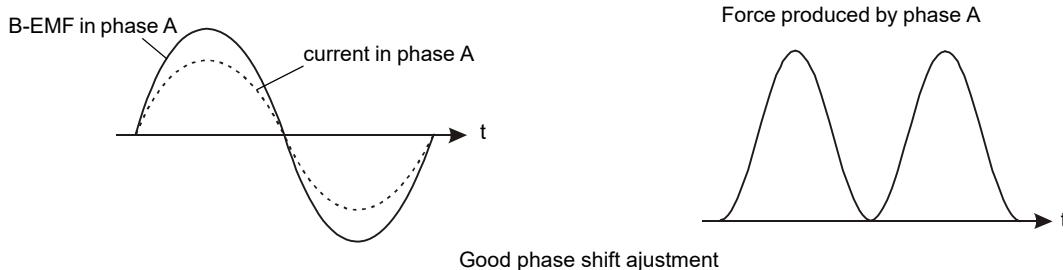
After a phasing, the position of the motor towards the magnets is precise enough to allow displacements. However, if the phasing is not correctly done, it is possible that the motor cannot give the maximum force. Sometimes the motor gives no force at all or worse, it gives a maximum force but in the wrong direction. In those cases, we will talk about phase shift adjustment and, in the last case, about an inverted phase shift adjustment. Refer to [§7.7.1.3](#) and [§7.9](#) for more information about the phase shift adjustment.

Remark: With the **INI** (phasing previously called **INItialization**) command, it is possible to restart a phasing cycle at anytime but the controller must be in 'Power Off' mode first. After the INI command, the controller is in the same configuration than after the first 'Power On' that is why a homing must then be done again. If K52=1 after the INI command, the fine phase adjustment (parameter K53) does not work because the IND command must be sent after the first 'Power On' or the INI command.

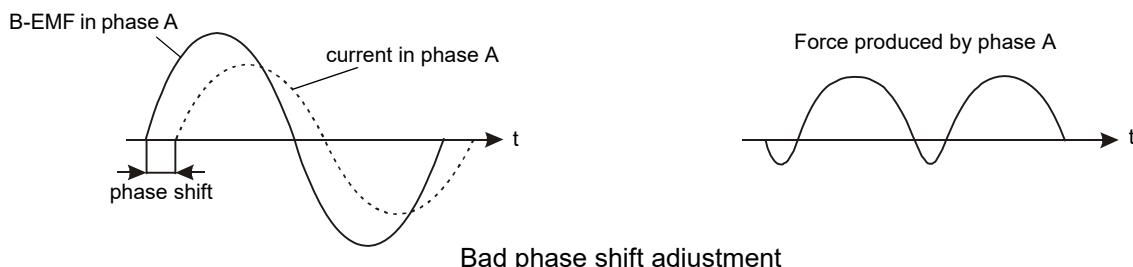
Command	Comment
INI.<axis>	Starts the phasing procedure.

7.7.1.3 Phase shift adjustment

To obtain the **maximum force** from the motor, the current injected in the coils (phases) by the controller must be in phase with the motor induced voltage (B-EMF). Ideally, this **phase shift should be null**. This offset correction is called **phase shift adjustment**, also called **PSA** in this document.

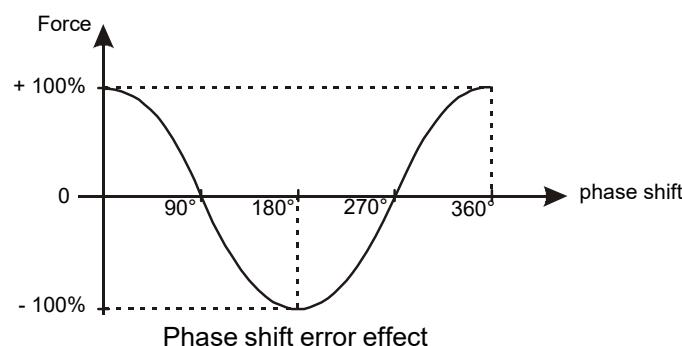


If the phase shift adjustment is **not correct**, the motor will not give its maximum force. It can even happen that no force at all is supplied, or even worse, the force is provided in the wrong direction.



If the motor phase shift is between 1° to 5°, the motor will still deliver around 95% of its nominal force. A higher phase shift will significantly reduce the force.

You can see below that for a phase shift as bad as 90° to 270° off the correct value, a force is provided in the wrong direction (with 180° the maximum force is provided in the wrong direction). In this case, the position loop becomes unstable and the **motor speeds suddenly up in the wrong direction!** In that case, there is 'an inverted phase shift adjustment'.



This is the speed error v_e which pushes the motor to speed up in the case of an inverted phase shift adjustment. In a regular process, the position loop regulator usually tries to decrease v_e , but as in our case the force F provided by the motor is opposed to the reference force F_c , the motor will move in the direction that tends to increase v_e rather than to decrease it.

7.7.1.4 Commutation look-up table parameters

The currents sent in the motor phases have a sinusoidal form. These types of current are calculated with the current reference generator. It is important to remember that the motor look-up table has a sinusoidal function period.

The parameter **K53** allows the user to have a repetitive adjustment after a homing:

- with an absolute EnDat encoder, the parameter K53 is directly taken into account whatever the value of the parameter **K52** (no homing is needed).
- AUT=8 allows the user to find the parameter K53 if K52=1 (refer to [§7.8.1](#) for more information about the AUT command).

K	Name	Value	Comment
K52	Look-up table mode	0 1	After a homing: Leave out parameter K53 Replaces the phase shift adjustment by parameter K53
K53	Look-up table phase adjustment	-	Motor phase shift adjustment value according to the position of the reference mark

7.7.2 Phasing processes (parameter K90)

Several phasing processes may be performed. The parameter **K90** allows the user to choose the phasing mode:

K	Name	Value	Comment
K90	Phasing mode and commutation	0 1 2 3 4 5 6 7	No phasing (with 1-ph. motor or absolute EnDat encoder). The value stored in K53 is used if K52=1 except for absolute EnDat encoder Phasing with current pulses (3-phase ironcore motors only). The value stored in K53 is used if K52=1 Phasing by constant current in the motor phases (ironcore and ironless motor). The value stored in K53 is used if K52=1 Phasing with digital Hall sensors (mode 1) until the index is found then commutation by position encoder. The value stored in K53 is used if K52=1 Phasing with digital Hall sensors (mode 2) until the first edge of signal then commutation by position encoder. When index is found, the value stored in K53 is used if K52=1 Phasing and commutation with digital Hall effect sensor only Small movement phasing. The value stored in K53 is used if K52=1 Small movements phasing. Repeated automatically 3 times when phasing process fails. The value stored in K53 is used if K52=1

Remark: The depth 1 of the parameter K90 is used for AUT.x=8 only when K90=2, 6 or 7.

7.7.2.1 K90=1: Phasing by pulses

K	Name	Comment	Units
KF91	Phasing pulse-current level	Pulse amplitude	[ci]
K98	Phasing voltage rate	Phasing power voltage rate in percent with K90=1. This is useful for low inductance motor	%

In this type of phasing (available only with 3-phase ironcore motor), pulses are sent with an amplitude given by the parameter **KF91**. During the phasing, the voltage present in the motor can be reduced with the parameter **K98** (useful if you have a low inductance motor driven by a 300VDC position controller).

- K98=100 corresponds to a full DC bus voltage (100%)
- If the parameter K98 is included between 25 and 100 : the voltage is equal to: $\frac{DCbus \cdot K98}{100}$

Example:

K98=50

For a controller working at 300 VDC, the voltage used for the phasing is 150 V (half of the DC bus voltage).

The method by pulses has the advantage of practically not moving the motor (some microns) but it is not as precise (about 10%) as a phasing by constant current.

Remark: **The phasing mode by pulses is not available on the AccurET VHPs. The K90=1 BANNED error (M64=149) is raised on the AccurET VHPs when the PWR or INI command is sent with K90=1.**

7.7.2.2 K90=2: Phasing by constant current

Parameters **KF92**, **K93**, **K94** and **K97** concern only the phasing by constant currents.

K	Name	Comment	Units
KF92	Phasing constant current level	Maximum current level when phasing with constant current.	[ci]
K93	Phasing ending point	Pointers final position in the look-up table of the current loop.	-
K94	Phasing maximum time	Maximum time allowed before the motor is in a stable balance point. Time [s] = $K94 \cdot 10^{-3}$	[ms]
K97	Phasing starting point	Pointers initial position adjustment in the look-up table of the current loop.	-

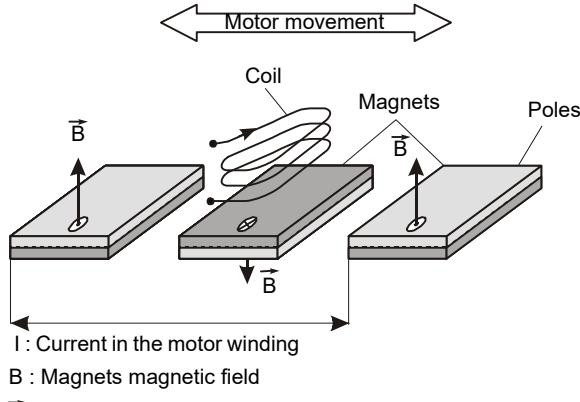
Remark: Refer to [§7.8.1](#) for more information about the autosetting with constant current.

Phasing by constant current:

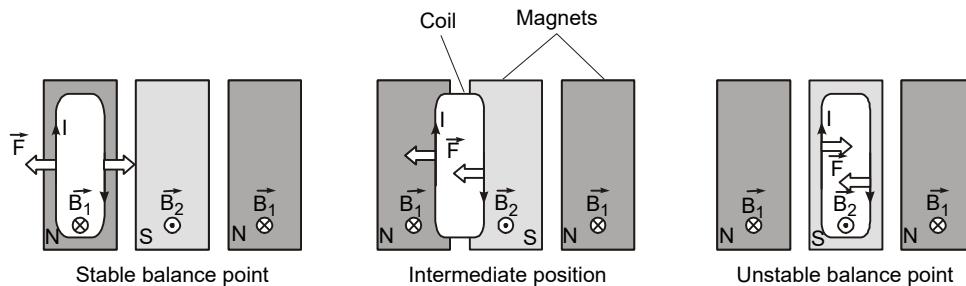
Supposing the user does a phasing by applying a constant current to one of the phase and a 0 current to the others. The Laplace force F_1 generated by the current and the magnetic field will move the motor. When the powered coil reaches a magnet pole, this force becomes null and the motor stops.

There are two possible configurations: either the coil has reached a stable balance point or a unstable one. In both cases the motor stops, but if the balance point is unstable, an external force, even weak, is enough to eject it from this position to the next stable balance point.

Stable and unstable balance points alternate every 16mm (for most ETEL's motors) if the magnetic period is equal to 32mm or every 32mm if the magnetic period is equal to 64mm.

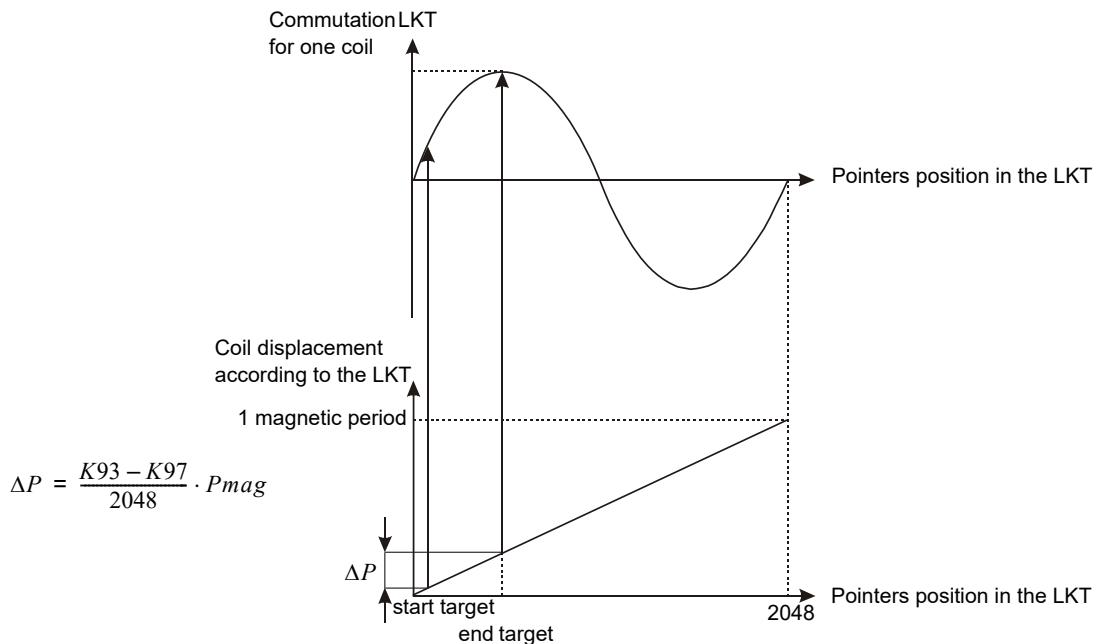


In fact, during a phasing, distinct constant current are sent into each phase of the motor instead of a single current in a single phase. In this case the principle remains exactly the same but the stable and unstable balance points are shifted with respect to the magnet pole according to the ratio between the injected currents.



If a motor has reached an unstable balance point, it means that it was already in this position at the beginning

of the phasing. This case must be avoided because the motor will not move when constant current are sent and it will be in 'inverted phase shift adjustment' (force F generated by the motor in opposition to force reference Fc). To avoid that problem, we move linearly according to the time, the position of the pointers in the look-up table. To simplify the drawing, the look-up table of only one phase is represented below:

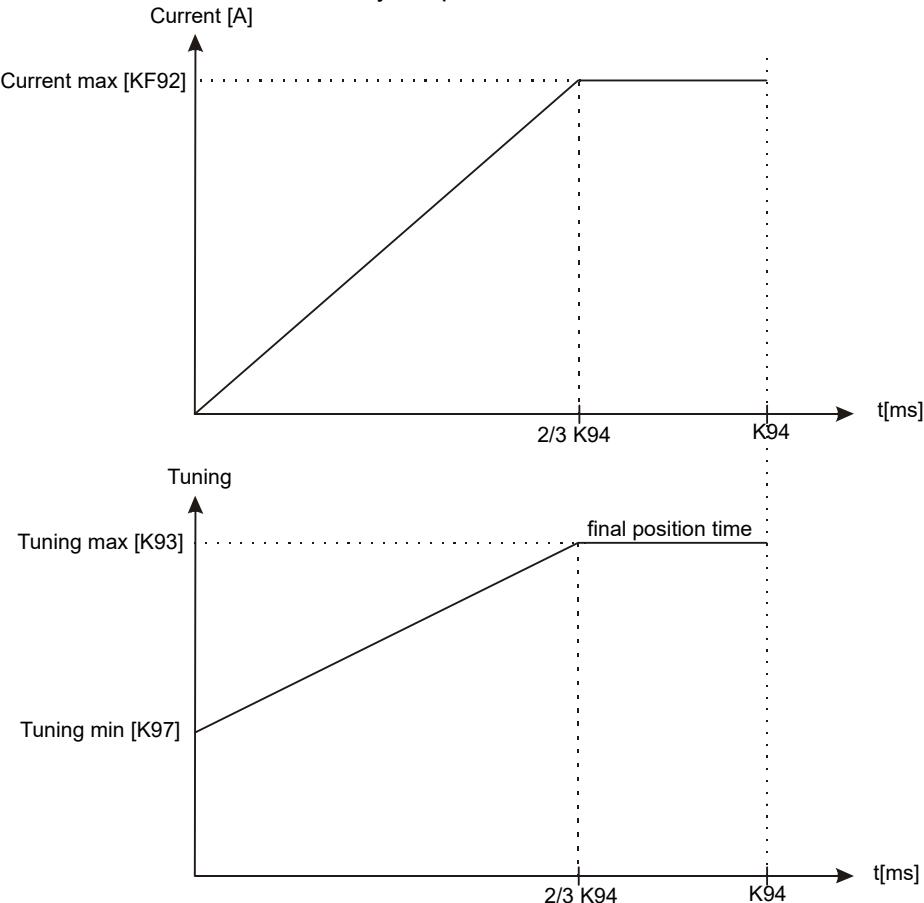


'Pmag' represents the magnetic period mentioned in the data sheet of the corresponding motor.

Pmag represents the magnetic period mentioned in the data sheet of the corresponding motor.

Moreover, when the phasing has to be done near a mechanical end stop, those same parameters allow the user to move the final stable balance point inside the admitted stroke of the motor if it is outside of it.

To avoid any sudden movement, the constant current is not immediately applied to the phases but it is linearly increased until the maximum value set by the parameter **KF92** is reached.



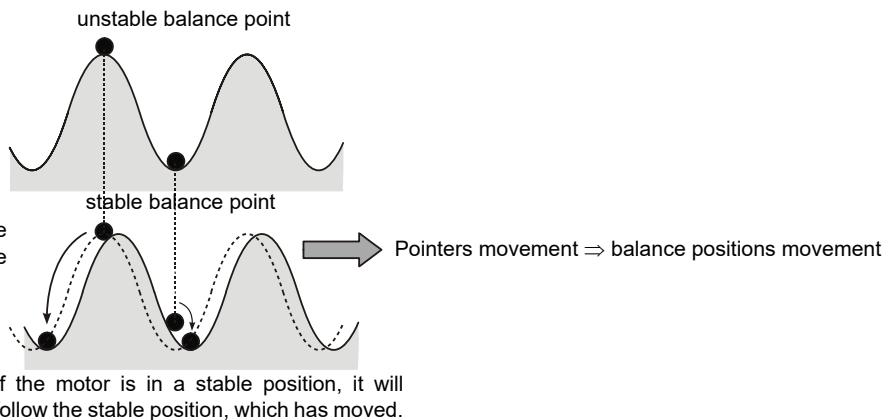
The time elapsed since the start of the phasing determines the end of the phasing by constant current. It is determined by the parameter **K94** and is expressed in seconds:

$$Time[s] = K94 \times 10^{-3}$$

We consider that the motor has reached a stable balance point position after this lapse of time.

To sum up, here are the three possible initial configurations:

- Case 1: The motor is not in a balance point position (stable or unstable) and moves to a stable balance point position when the current in the phase is increased.
- Case 2: The motor is in a stable balance point position and when the current increases, it follows the shift of the balance point position.
- Case 3: The motor is in an unstable balance point position and when the balance point moves, the motor is confronted to the same problem as case 1 and moves when the current increases to a stable balance point position.



In all cases, the motor reaches at the end of the phasing a stable balance point position. Moving the pointers in the motor look-up table is useful when the motor is initially in the same configuration as case 3.

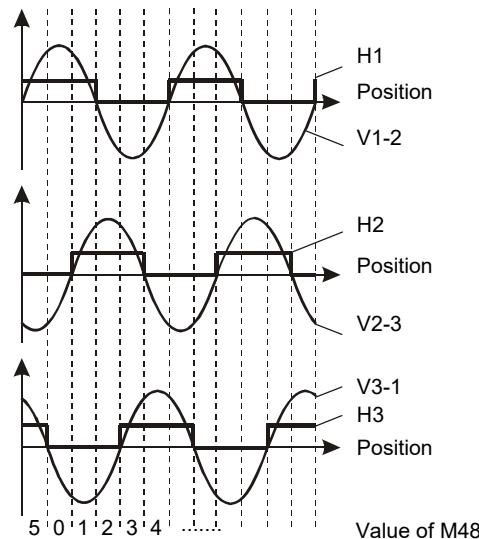
7.7.2.3 Phasing with digital Hall effect sensor

It is possible to interface digital Hall effect sensor only with three-phase motors. Three digital inputs (H1, H2 and H3), specific or not, can be used to connect a digital Hall effect sensor if K90=3, 4 or 5. The digital Hall effect sensor is connected to these 3 inputs (refer to the '**Hardware Manual**' for more information).

The monitoring M48 allows the reading of the counter of the digital Hall effect sensor:

M	Name	Comment
M48	Digital Hall effect sensor signal	Gives the value of the digital Hall effect sensor's counter only if K90=3, 4 or 5

On the following graph, the Hall signals and the sine voltages between the motor phases (V1-2, V2-3 and V3-1) are displayed:

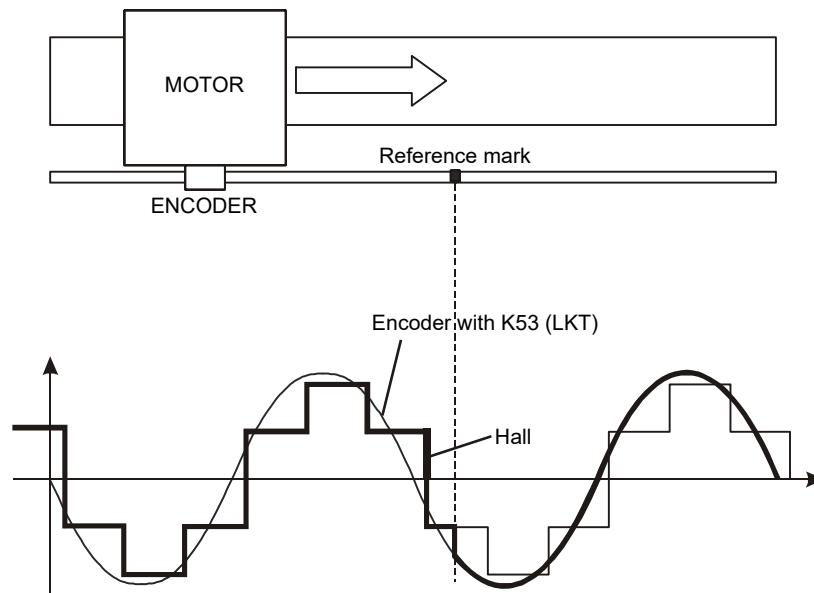


One period of the signals is divided into 6 different parts. These parts (as mentioned above) are associated to the following values of the monitoring M48: 5, 0, 1, 2, 3 and 4 respectively.

Remark: M48=255 indicates that there is a cabling problem of the digital Hall effect sensor.
 If there is a wrong value coming from the Hall effect sensor, a **W DIGIT. HALL** warning (M66=8) or a **ERROR HALL** error (M64=191) can occur.

- **K90=3: Phasing with digital Hall effect sensor (mode 1)**

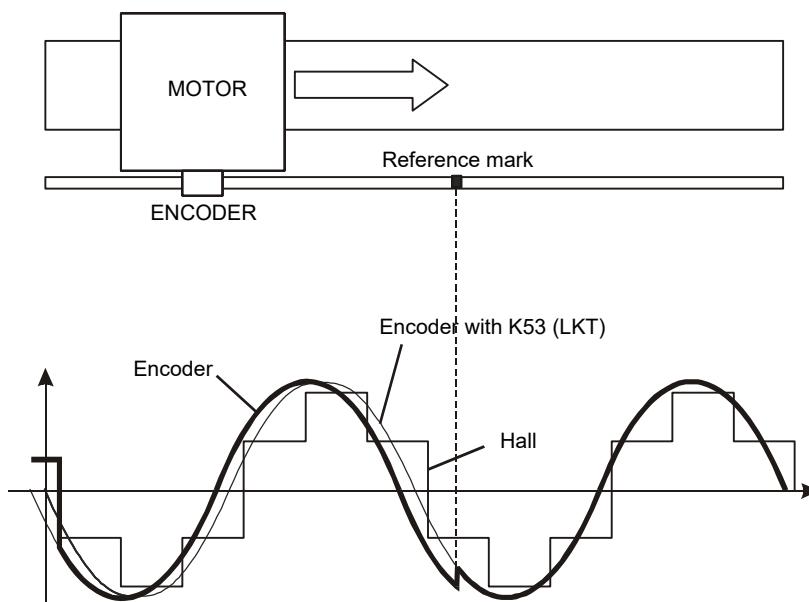
The commutation is done with a digital Hall effect sensor up to the reference mark is found then the position encoder is used with the value stored in the parameter K53 (phase shift adjustment).



Remark: Refer to the '**Hardware Manual**' for more information about the connection of the Hall effect sensor.

- **K90=4: Phasing with digital Hall effect sensor (mode 2)**

The commutation is done with a digital Hall effect sensor up to the first edge of the Hall effect and with the position encoder up to the reference mark is found. From then on, the commutation is done with the position encoder with the value stored in the parameter K53 (phase shift adjustment).

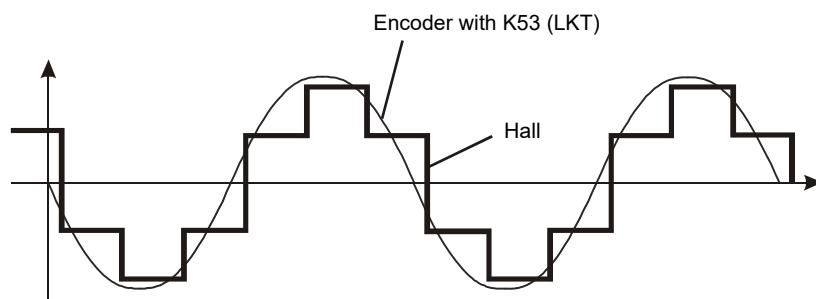
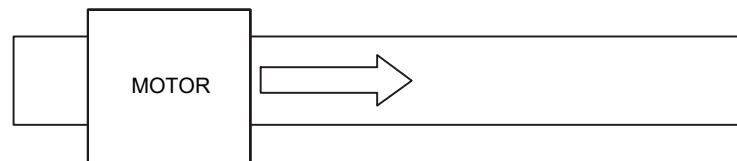


Remark: When the parameter K90 or K68 is modified, the command AUT=10 must be executed to re-calculate the value of the parameter K53.

Refer to the '**Hardware Manual**' for more information about the connection of the Hall effect sensor.

- K90=5: Phasing and commutation with digital Hall effect sensor only**

The commutation is only done with the digital Hall effect sensor.



Remark: When the parameter K90 or K68 is modified, the command AUT=10 must be executed to re-calculate the value of the parameter K53.
Refer to the '**Hardware Manual**' for more information about the connection of the Hall effect sensor.

7.7.2.4 K90=6 and 7: Small movement phasing

This type of phasing allows the user to initialize a motor by moving as little as possible. This type of phasing can be used with any types of motors (ironless and ironcore). The motor is initialized by measuring the movement induced by a succession of current pulses. There are two parameters to set to achieve the phasing:

K	Name	Comment	Units
KF91	Phasing pulse-current level	Pulse amplitude	[ci]
K101	Phasing time	Phasing time	[ms]

The current level used for the phasing may widely vary, depending mainly on the moving inertia. ETEL advises to begin the setting with a current level (given by the parameter **KF91**) at 5% of the current limit (parameter KF60). The user should hear a low oscillation noise at the first 'Power On'.

The parameter **K101** is useful to slow down the motor between the different pulses when using a high inertia system. The default value of the parameter K101 is 1000 [incr]=1 [sec]. When K101=0, the total phasing time is about 60ms, but a short phasing fits only a certain type of application and is not 100% reliable with most systems.

During the tuning of the small movement phasing, it is recommended to check the value of the monitoring MF95:

M	Name	Comment	Units
MF95	Phasing fitting coefficient	Fitting coefficient of the successive phasing moves	[%]

This monitoring gives an indication on the reliability of the phasing. A perfect phasing typically yields a value smaller than 2 %. A value smaller than 5 % can be considered as good, but any bigger value would require an improvement of the setting. First by increasing the value of KF91, secondly by increasing the current loop bandwidth (KF80, KF81).

When K90=6 or 7, two possible phasing errors may occur:

INITIALI LOW CUR error (M64=153): this error appears when the movement is too small during the phasing process. The value of the current (parameter KF91) is too low and should be increased in this case.

INITIALI HIGH CUR error (M64=154): this error appears when the movement is too important ($> \pm 20\%$ of the magnetic period) during the phasing process. The value of the current (parameter KF91) is too big and should be lowered in this case.

Remark: The difference between K90=6 and K90=7 is that with K90=7, the phasing is automatically repeated 3 times when the process fails. **However, the setting must be done ONLY with K90=6** and only afterwards K90=7 can be used. If after three tries, it is still unsuccessful, the controller generates one of the two above-mentioned errors. When K90=7, the controller waits for K101 + 1sec. between two successive executions and a **WPHASING FAILURE** warning (M66=13) occurs as long as the phasing process is running.

7.8 Autosetting

7.8.1 AUT command

The **AUT** command (**AUTomatic setting**) calculates automatically parameters **KF80** and **KF81** (current loop proportional & integral gain). Parameters **K56** (motor phases connection), **K53** (motor phase shift adjustment) if K52=1 and **K98** (DC bus voltage) are also set. This command **initializes** the motor. Therefore the adequate values must be conveniently stored in parameters KF91 to K97.

Command	<P1>	bit#	Comments
AUT.<axis>=<P1>	1	0	Tunes the proportional and integrator gain of the current loop parameters KF80, KF81, and DC power voltage rate parameter K98.
	2	1	Searches motor phases and sets parameter K56.
	8	3	Sets parameter K53 (fine phase adjustment) if K52=1.

Remark: The AUT command is only possible when the controller is in 'Power Off' mode (no current in the motor phases). As the AUT command is a bits field, it is possible to execute, for example, AUT.1=10 (bits 1 & 3).

During the autosetting process, the message '**AUT CMD IN PROG**' is displayed.

Command		Calculated parameters						Read parameters					
<P1>	bit#	K53	K56	KF80	KF81	K98	Short movement phasing (KF91 and K101 with K90:1)	Constant current phasing (KF92, K93, K94 and K97 with K90:1)	Homing (K32, K52, K58, K40 to KL48)	Encoder reading way inversion (K68)	Definition mode	K133	
1	0			OK	OK	OK							
2	1		OK				X	X		X			
8 (*)	3	OK					X	X		X	X	X	X

(*): The initialization launched when the AUT.x=8 command is sent, is defined by depth 1 of parameter K90.

- If K90:1=6 or 7, the phasing is executed by constant current
- If K90:1=6 or 7, the phasing is executed by short movement with parameters KF91 and K101.

Remark: The parameter KF80 can only be calculated with the AUT command for **ironcore motors**. With an ironless motor, the value of the parameter KF80, calculated with the AUT command, can be completely wrong.

Refer to [§7.7.2.2](#) for more information about the phasing with constant current.

The parameter **K56** is first calculated when executing a phasing by constant current. Then, the motor moves of a distance equal to 45° of the electric period.

Remark: During the AUT command, the **TIMEOUT AUT CMD** error (M64=156) will appear if the motor did not move after the time defined by the parameter K94 or if no value has been found for the parameter K56.

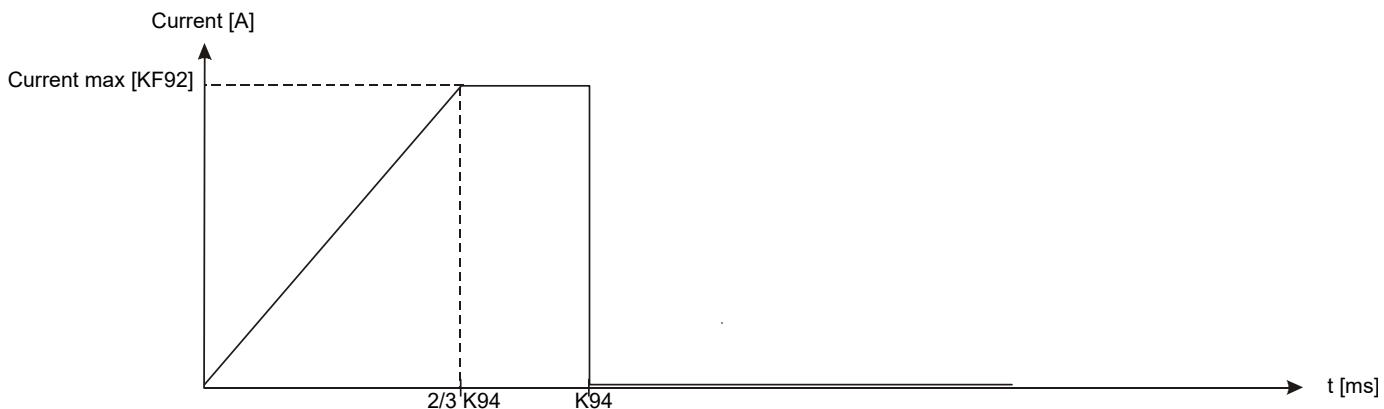
When the index is found, the controller calculates the phase shift adjustment with respect to the index and this value is stored in the parameter K53. To perform this, phasing parameters by constant current and homing parameters need to be correctly programmed.

Remark: With an absolute EnDat encoder as main encoder, there is no homing. Thus, there is the phasing and then the reading of the absolute position.

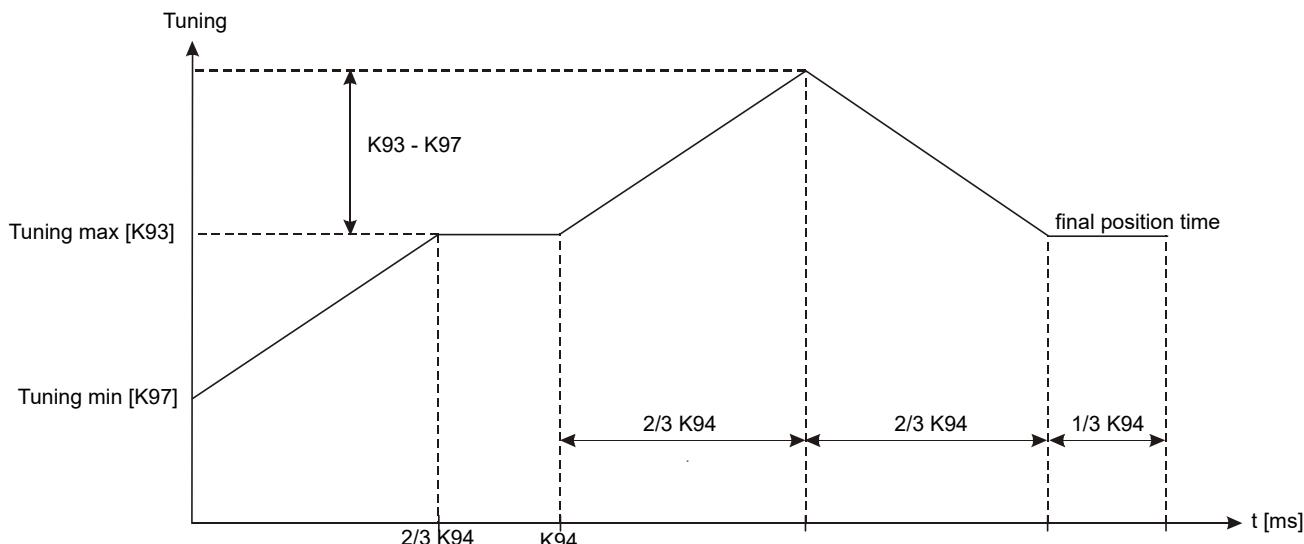
Parameters KF80, KF81, K98 are calculated first, then the parameter K56, and finally the parameter K53.

With the bit#0 of the parameter **K133**, it is possible to choose the principle for finding the fine phase adjustment (parameter K53) when using the AUT command. The parameter K133 can be used only when K90:1 ≠ 6 or 7. This method allows a better compensation of the friction forces.

- If bit#0=0
 - Step 1: The look-up table pointer moves from parameters K97 to K93 during 2/3 of the time defined by the parameter K94.
 - Step 2: The process waits for 1/3 of the time defined by parameter K94.
 - Step 3: Calculation of the position value of the motor's coil towards the magnet (after the homing, this value is used for the calculation of the fine phase value (parameter K53)).



- If bit#0=1
 - Step 1: The look-up table pointer moves from parameters K97 to K93 during 2/3 of the time defined by the parameter K94
 - Step 2: The process waits for 1/3 of the time defined by the parameter K94.
 - Step 3: First calculation of the position value of the motor's coil towards the magnet.
 - Step 4: Move away from parameter K93, in the same direction from a value given by parameters K93 to K97 during 2/3 of the time defined by the parameter K94.
 - Step 5: Move back to parameter K93 during 2/3 of the time defined by the parameter K94.
 - Step 6: The process waits for 1/3 of the time defined by the parameter K94.
 - Step 7: Second Calculation of the position value of the motor's coil towards the magnet.
 - Step 8: The average between the two values is used for the position value of the motor's coil towards the magnet (after the homing, this value is used for the calculation of the fine phase value (parameter K53)).



The best calculation of the parameter K53 after the AUT.x=8 command is achieved as follows:

- For horizontal movement (restoring force negligible) => K90:1.x=2 (phasing by constant current) and K133.x=1.
- For vertical movement (restoring force non-negligible) => K90:1.x=6 (small movement phasing).

7.8.2 PWR command

The **PWR** command (**PoWeR**) initializes the motor then **supplies the motor phases** ('Power On') (it can also power it off). The phasing is done according to parameters K90 to K98. If a phasing has already been executed, it is not executed again and only the power in the phases is applied. When the PWR command is sent, the position loop must be correctly regulated because the motor is under control. The PWR command must be executed each time the motor is switched on.

Remark: If the PWR command is sent while the mask of the parameter KL305 (refer to [§7.4.3.1](#)) is not set, the controller enters in **POWER OFF/ON** error mode (M64=26).

When the power is cut off with the PWR.1=0 command, the motor position keeps on being calculated permanently. The motor position value is not erased.

Command	<P1>	Comment	Read parameters
PWR.<axis>=<P1>	0 1	Motor power switched off. Phasing and switches on the power in the phases.	K90 to K98 and KL305

Example:

```
PWR.1=1; // Phasing is done and the phases are powered. The motor is set in position and is // ready to move. 'Power On' is displayed on ComET.
```

```
PWR.1=0; // The power is switched off. It is possible to freely move the motor by hand but the // position value is always calculated.
```

7.9 Homing

7.9.1 Homing purpose

After a phasing (refer to [§7.7.1](#)), a **homing** has to be carried out to determine the motor absolute position. The movement of the motor can highly vary during a homing according to the type of encoder used. Some display several reference marks at regular distance from each other to know the motor absolute position (multi-reference mark encoder) and others display only one (mono-reference mark encoder). Finally some encoders have no reference marks and the homing has to be done against a fix **mechanical end stop** or against a **limit switch** or a **home switch**. The **IND.<axis>** command starts the homing (it can be used only if the power is on and K61=1 but not in interpolated mode (ITP)). With an absolute EnDat encoder, there is no homing as it gives an absolute position. After having found the motor absolute position, a constant phase shift adjustment value is calculated for each powered system (therefore, the motor's force will be the same after each homing).

Command	Comment
IND.<axis>	Starts a homing process

The monitoring **M5** (ML5) allows the user to display the distance (dpi) covered due to the homing and the initial phasing.

M	Name	Comment
M5 ML5	Homing offset	Covered distance to the homing and the initial phasing (dpi). The monitoring M5 corresponds to the LSL part of the monitoring ML5.

Remark: The homing status can be monitored with bit#2 of M60 (refer to [§8.10](#)). This bit is set when the controller has finished the homing process with success.
Refer to [§12.6](#) for more information about the homing in gantry mode.

7.9.1.1 Determination of the motor absolute position

For **multi-reference marks encoders**, the motor moves and knows its absolute position as soon as it finds two successive reference marks (they are coded). It means that the position origin, also called '**0 machine**', is set. If during a next homing the motor uses two different successive reference marks, the '**0 machine**' remains set in the same position.

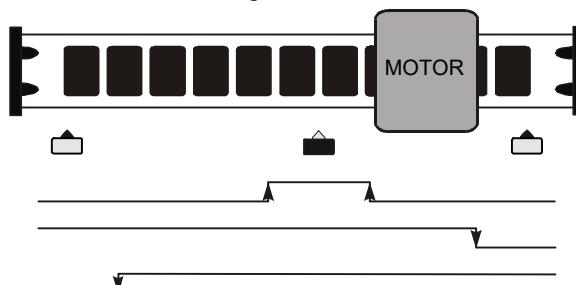
Remark: With a multi-reference marks encoder linear motor, the '**0 machine**' can appear outside the total motor stroke.

Immediately after having set the '**0 machine**', the controller adds up the value contained in the parameter KL45 [upi] (with an absolute EnDat encoder, parameter KL45 is added to the absolute position). This value is given in [upi] (refer to [§22.1.1](#)). This procedure places the '**0 machine**' at any position set by the user according to the application. It is possible to move the '**0 machine**' to the current position of the motor with the **SET** command if the power is on and K61=1 but not in interpolated mode (ITP).

For **mono-reference mark encoders** or with a **mechanical end stop**, **home switch** or **limit switch**, the motor moves until it reaches the reference mark and places there the '**0 machine**'. Then the controller also adds up the value contained in the parameter KL45 [upi]. The motor does not exactly stop on the reference mark.

The controller allows the user to manage two types of home switch and limit switch:

- External home/limit switch (not integrated in the encoder). In that case, the positive limit switch (reached with a positive movement of the motor) must be connected to the digital input 10 (DIN10), the negative limit switch (reached with a negative movement of the motor) must be connected to the digital input 9 (DIN9) and the home switch must be connected to the digital input 2 (DIN2). There can be only limit switches or only home switch or both together.



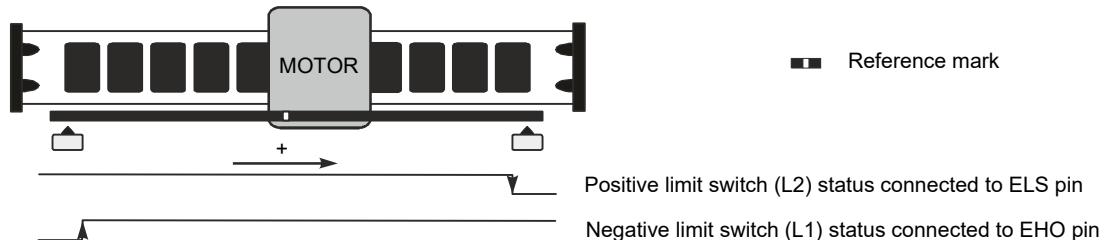
Home switch status given by DIN2

Positive limit switch status given by DIN10

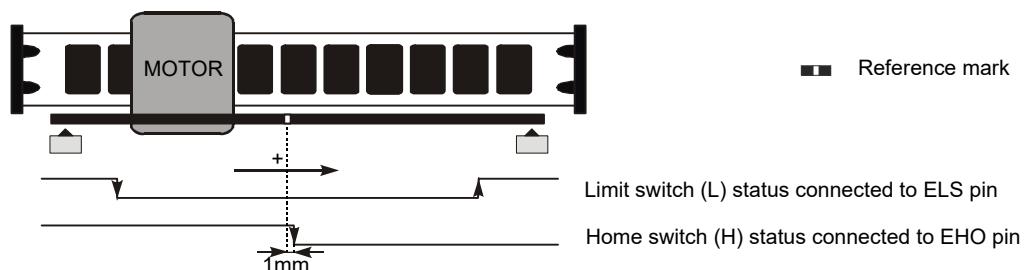
Negative limit switch status given by DIN9

- Encoder's home/limit switch. In that case, they are integrated in the encoder's head and the controller allows the user to connect them directly to the EHO and ELS pins of the encoder's connector (refer to the corresponding '**Hardware Manual**' for more information about these pins).

- If there are two limit switches, generally called L1 for the negative one and L2 for the positive one, they can be connected as follows:



- If there are one limit switch and one homing switch(*), generally called L for the limit switch and H for the home switch, they can be connected as follows:



(*): The home switch is positioned within a gap of 1mm.

Bit#1, 2 and 3 of the parameter K32 allow the user to invert either the home switch or the limit switch (which means to choose the polarity) depending on the homing mode selected by the parameter K40. When a motor is not positioned on a home switch, the latter is set to '0', and when it is positioned on it, it is set to '1'. However these values can be inverted as well as for the home switch. Bit#4 and 5 of the parameter **K32** allow the user to swap the limit switch (cross positive limit switch and negative limit switch) if the user has selected the encoder reading way inversion (K68:0=1).

K	Name	Value	bit#	Comment
K32	Limit switch and home switch inversion	2	1	Inverts the polarity of the home switch (only DIN2).
		4	2	Inverts the polarity of the limit switches or home switch from the encoder.
		8	3	Inverts the polarity of the limit switches from the digital inputs.
		16	4	Swap limit switches or home switch from the encoder.
		32	5	Swap limit switches from digital inputs.

The parameter **K58** allows the user to define the type of limit switch used because it is also possible to select the limit switches of the encoder.

K	Name	Value	Comment
K58	Limit switch mode	0	No limit switch
		1	Limit switches on DIN9 and DIN10
		2	Limit switches L1 / L2 (L1=EHO pin signal and L2=ELS pin signal)
		3	Limit switches L / H (L=ELS pin signal and H=EHO pin signal)

Remark: The parameter K58 can be used for the homing as well as for axis protection. However it is not possible to manage two different limit switches (one for axis protection, one for homing) at the same time.

The monitoring **M44** shows the status of the home switch and limit switch signals coming from the encoder's connector.

M	Name	Value	bit#	Comment
M44	Limit switch and home switch status	1	0	Indicates the state of the EHO pin signal (L1 or H)
		2	1	Indicates the state of the ELS pin signal (L2 or L)

The homing parameters define the type of reference mark used (parameter **K40**) as well as the behavior of the motor when, for instance, it reaches a mechanical end stop before a reference mark (parameters **KL47** and **KL48**). Thanks to these parameters the required speed and acceleration for the search of a reference mark (parameters **KL41** and **KL42**) are set as the movement during a homing is a trapezoidal movement. Current and maximum position error for an end stop detection are set via parameters **KL43** and **KF44**.

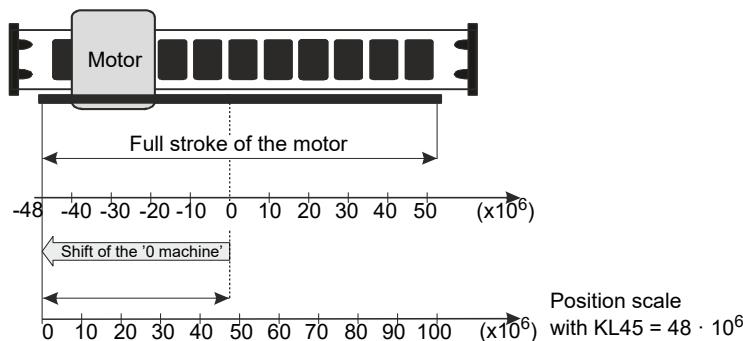
After a homing, the speed (parameter **KL211**), the acceleration (parameter **KL212**) are modified by the values stored in parameters **KL41** and **KL42** and the movement type is modified to MMD=1. The **HSPD** (Homing SPeeD), **HACC** (Homing ACCeleration) and **HOFFS** (Homing OFFSet) commands are alias of the parameters **KL41**, **KL42** and **KL45** respectively, and use the same syntax.

K	Alias	Name	Comment	Unit
K40	HMODE	Homing mode	Chooses the required homing mode (refer to §7.9.2).	-
KL41	HSPD	Homing speed	Motor speed during a homing.	[usi] [rusi]
KL42	HACC	Homing acceleration	Motor acceleration during a homing.	[uai] [ruai]
KL43	-	Max pos error for end stop detection	Maximum position error for mechanical end stop detection (KL43 < K30).	[dpi] [rdpi]
KF44	-	Max current for end stop detection	Maximum current limit for mechanical end stop detection (KF44 < KF60).	[ci]
KL45	HOFFS	Offset for absolute position	Offset added to the position after having found the index or the absolute position with an absolute EnDat encoder.	[upi] [rupi]
KL46	-	Homing movement stroke	The motor will cover the distance defined by parameter KL46 for K40=20, 21, 24 to 27, 36 to 39.	[upi] [rupi]
KL47	-	Mvt. at init. on limit switch or end stop	Movement after homing on mechanical end stop or limit switch.	[upi] [rupi]
KL48	-	Mvt to go out of an index or home switch	Movement to leave an index or a home switch if the motor is positioned on the top of it at the start or find it from the wrong way.	[upi] [rupi]

Once the homing procedure done (according to parameter **K40**) or the absolute position known (in case of absolute EnDat encoder), the position of the '0 machine' is determined and modified by an offset given by the parameter **KL45** to place the origin position in the requested position.

Example:

A homing with a mono-ref. mark encoder linear motor is realized. The user wants to position the '0 machine' on the left, at the beginning of the motor stroke. He can do it entering the value $48 \cdot 10^6$ in parameter **KL45** as follows:



7.9.1.2 Calculation of the constant phase shift adjustment for each setting

The phasing determines the motor position according to the magnets by initializing the pointers in the look-up table of the current loop. This value can vary from one time to another. The **AUT** command and parameters **K52** and **K53** (refer to [§7.7.14](#)) enable the controller to avoid that. When the motor is **first** switched on, the **AUT** command allows a precise calculation of the motor phase shift adjustment **according to the reference mark** (**AUT** phase shift adjustment). This phase shift adjustment value is stored in the parameter **K53**. Then each time the motor is switched on, the **INI** command calculates a phase shift adjustment value and the **IND** command moves the motor until the reference mark is reached. At that moment, the **INI** phase shift adjustment value is replaced (according to the parameter **K52**) by the value contained in the parameter **K53** (**AUT** phase shift adjustment). Therefore the motor phase shift adjustment is always the same each time the motor is switched on, and no longer depends on the phasing (refer to [§7.7.1](#)).

7.9.2 Homing modes

The parameter K40 allows the user to select the homing mode according to the devices in the system. The **HMODE (Homing MODE)** command is an alias of this parameter and uses the same syntax. In the table hereafter, the symbol (OK) indicates the device used for the homing and the cross (x) indicates another existing device. For example, K40=6 stands for a homing with a positive movement towards a home switch for a system provided with limit switches to detect stroke ends.

K40	Direction	Mechanical end stop	Home switch (DIN2)	Limit switch (DIN) K58=1	Limit switch (L1/L2) K58=2	Limit switch (L/H) K58=3	Mono-ref. mark encoder	Multi-ref. mark encoder	Remark
0	+	OK							
1	-	OK							
2	+	X	OK						
3	-	X	OK						
4	+	X		OK	OK	OK			
5	-	X		OK	OK	OK			
6	+		OK	X	X	X			
7	-		OK	X	X	X			
8	+	X					OK		
9	-	X					OK		
10	+			X	X	X	OK		
11	-			X	X	X	OK		
12	+	X						OK	
13	-	X						OK	
14	+			X				OK	
15	-			X				OK	
16	+	X					OK+DIN2		Particular homing in which the controller only selects a ref. mark if the input DIN2 is set to 1.
17	-	X					OK+DIN2		Particular homing in which the controller only selects a ref. mark if the input DIN2 is set to 1.
18	+			X	X	X	OK+DIN2		The trip is defined by parameter KL46
19	-			X	X	X	OK+DIN2		The trip is defined by parameter KL46
20	+	X						OK	The trip is defined by parameter KL46
21	-	X						OK	The trip is defined by parameter KL46
22									Immediate homing. The actual position is the ref. mark.
24	+	X					OK		The trip is defined by parameter KL46
25	-	X					OK		The trip is defined by parameter KL46
26	+	X					OK		The trip is defined by parameter KL46 if mechanical end stop
27	-	X					OK		The trip is defined by parameter KL46 if mechanical end stop
28-33									Reserved for future use
34						X	OK		The trip is defined by the status of the limit switch and home switch
35						X	OK		The trip is defined by the status of the limit switch and home switch
36	+	X	OK						The trip is defined by parameter KL46 if mechanical end stop
37	-	X	OK						The trip is defined by parameter KL46 if mechanical end stop
38	+	X					OK		The trip is defined by KL46 after having found the mechanical end stop
39	-	X					OK		The trip is defined by KL46 after having found the mechanical end stop

K40	Direction	Mechanical end stop	Home switch (DIN2)	Limit switch (DIN) K58=1	Limit switch (L1/L2) K58=2	Limit switch (L/H) K58=3	Mono-ref. mark encoder	Multi-ref. mark encoder	Remark			
40 - 43				Reserved for future use								
44	+			X	X	X	OK		The trip is defined by KL46 after having found the limit switch			
45	-			X	X	X	OK		The trip is defined by KL46 after having found the limit switch			
46									Homing without effect on the position. It is used with absolute EnDat encoder in order to not change the position			

Remark: Refer to [§12.6](#) for more information about the homing in gantry mode.
With an incremental EnDat encoder, it is only possible to execute a homing with the modes K40=8, 9, 10, 11, 16, 17, 18, 19, 24, 25, 26 and 27.

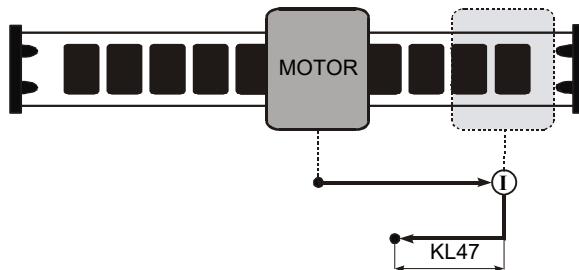
Here is a description of the symbols used in this paragraph:

- (I) Reference position
- Reference mark
- (O) Zero machine
- Motor position
- Motor trip

Remark: In the following examples, the reading head is positioned in the middle of the motor.

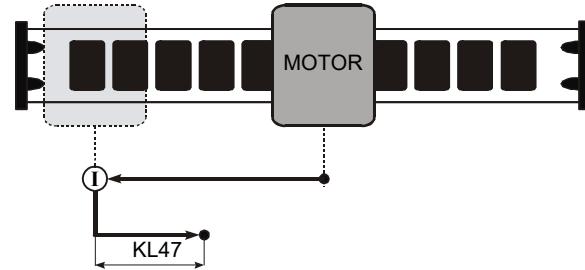
K40=0:

Homing against a mechanical end stop with a positive movement. After having found the mechanical end stop, the motor moves back the distance given by the parameter KL47.



K40=1:

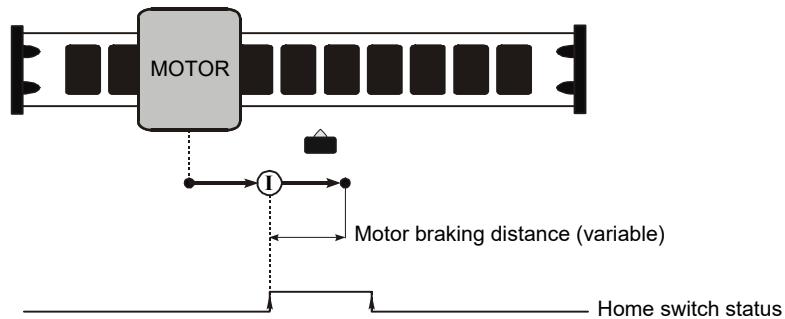
Homing against a mechanical end stop with a negative movement. After having found the mechanical end stop, the motor moves back the distance given by the parameter KL47.



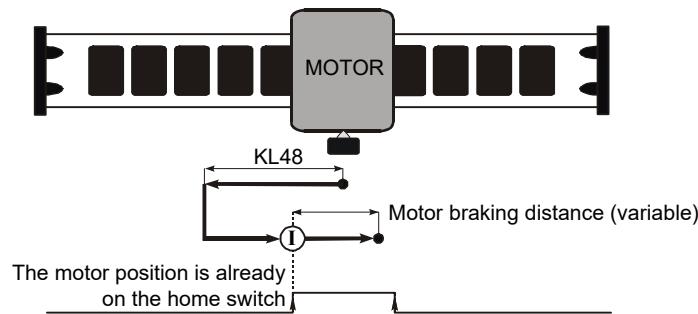
K40=2:

Homing on a home switch with a positive movement. To have the home switch always in the same position the motor must find it when moving in the determined direction. There are three possibilities:

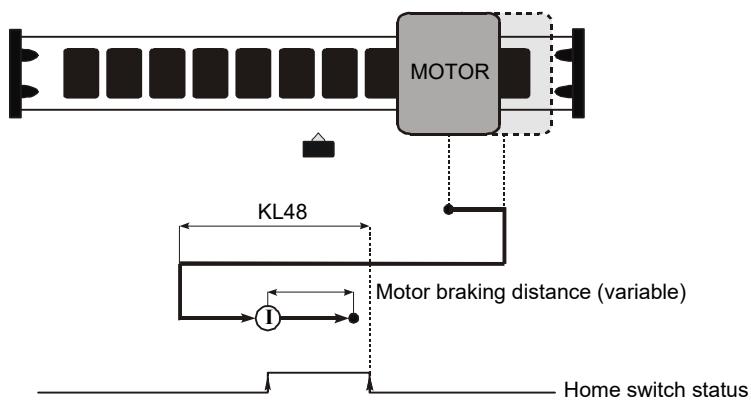
1. The motor is on the left of the home switch at the beginning of the homing. It moves and directly meets the home switch



2. The motor is on the home switch at the beginning of the homing. It moves the distance given by parameter KL48 in the opposite direction of the homing to leave the home switch. Then it moves back towards the home switch 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.



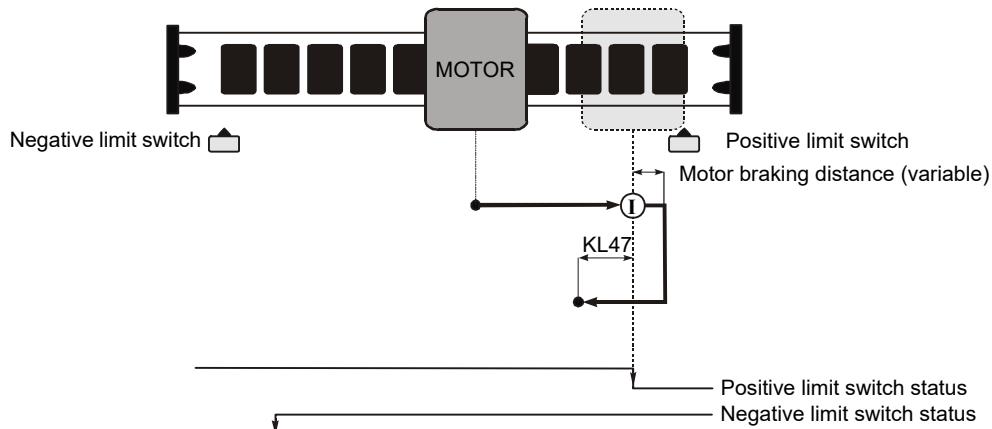
3. The motor is on the right of the home switch at the beginning of the homing. After having found the mechanical end stop, it comes back. When it finds the home switch (in the wrong direction), it keeps moving the distance given by the parameter KL48 before changing direction a second time to find the home switch 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.

**K40=3:**

Same as K40=2 with a negative movement.

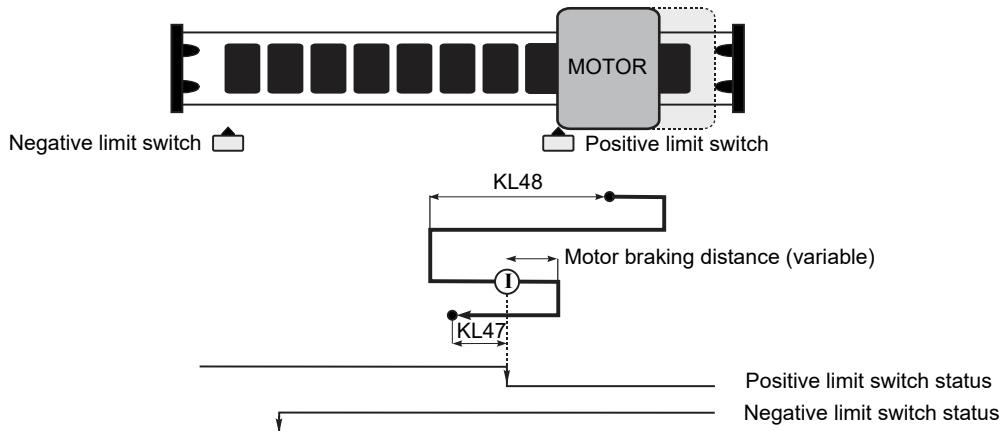
K40=4:

Homing on a limit switch with a positive movement. After having found the limit switch, the motor moves back the distance given by the 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 the parameter KL48.

If the motor is on the right of the limit switch at the beginning of the homing, it comes back after having found the mechanical end stop. When it finds the limit switch (in the wrong direction), it keeps moving the distance given by the parameter KL48 before changing direction a second time to find the limit switch in the right direction. The parameter KL48 must be bigger than the limit 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. After having found the limit switch, the motor moves back the distance given by the parameter KL47.

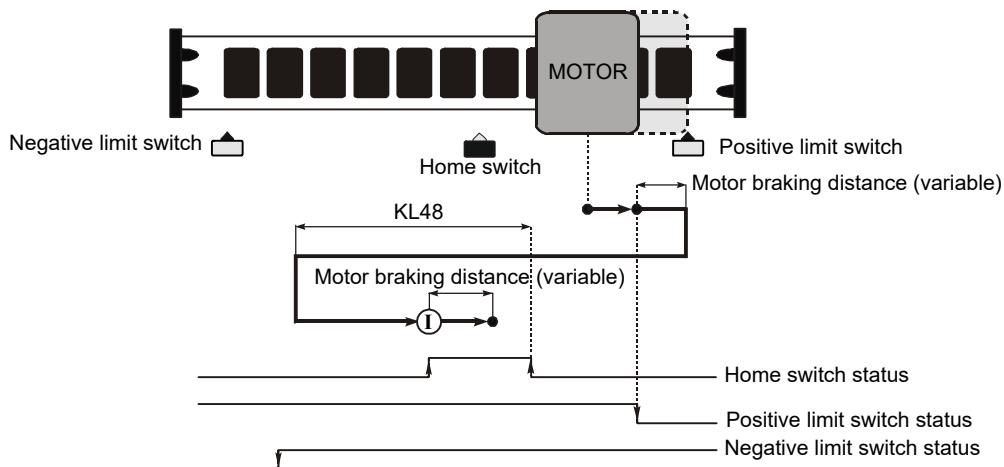


K40=5:

Same as K40=4 with a negative movement.

K40=6:

Same as K40=2 but the motor changes direction when meeting a limit switch instead of a mechanical end stop. Only case 3 is shown here. The motor is on the right of the home switch at the beginning of the homing. After having found the limit switch, it comes back. When it finds the home switch (in the wrong direction), it keeps moving the distance given by the parameter KL48 before changing direction a second time to find the home switch 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.



If the motor is on the home switch at the beginning of the homing, the motor moves of the value given by the parameter KL48. 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.

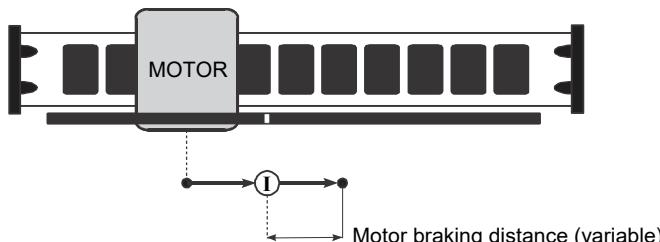
K40=7:

Same as K40=6 with a negative movement.

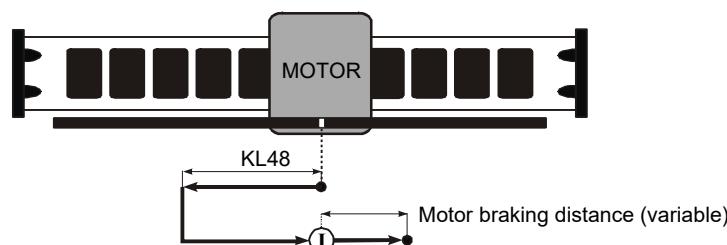
K40=8:

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:

1. 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.

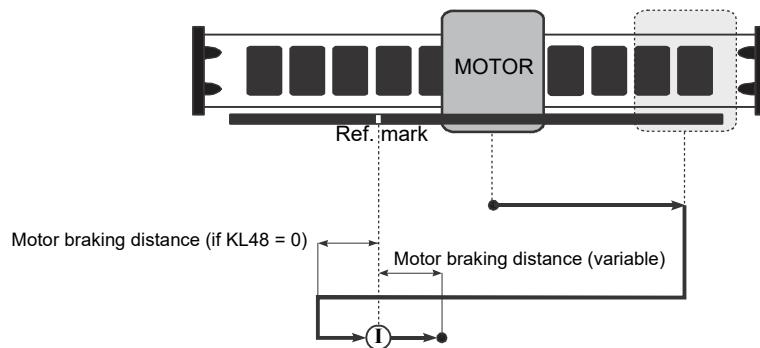


2. The motor is on the mono-reference mark at the beginning of the homing. It moves the distance given by parameter KL48 in the opposite direction of the homing to leave the mono-reference mark. Then it moves back towards the mono-reference mark in the right direction. The parameter KL48 must be bigger 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.



3. 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.



With an incremental EnDat 2.2, the homing process stops as soon as the reference mark is found. Consequently the accuracy of the process depends on the initial position.

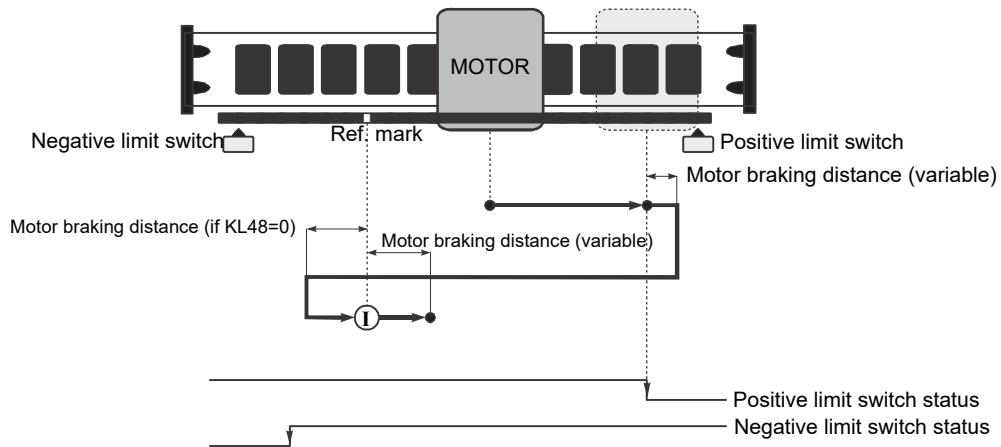
K40=9:

Same as K40=8 with a negative movement.

K40=10:

Same as K40=8 but the motor changes direction when meeting a limit switch instead of mechanical end stop. Only case 3 is shown.

If the motor is on the limit switch, it moves back the distance given by the parameter KL48.



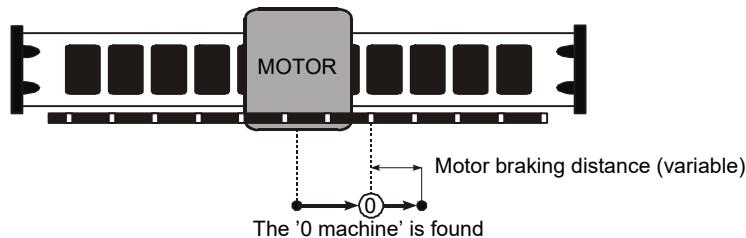
With an incremental EnDat 2.2, the homing process stops as soon as the reference mark is found. Consequently the accuracy of the process depends on the initial position.

K40=11:

Same as K40=10 with a negative movement.

K40=12, 13, 14 or 15:

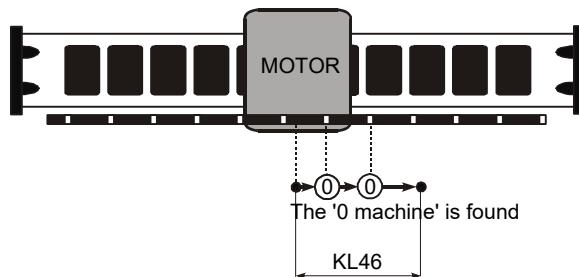
Homing with a multi-reference mark with a positive movement. The motor needs to find only two successive reference marks to determine its **absolute position** (the '0 machine' is always positioned at the same place regardless the two reference marks found). If the motor finds one reference mark followed with a mechanical end stop (K40=12 or 13) or with a limit switch (K40=14 or 15) the reference mark is not considered and the motor starts the homing in the opposite direction. If the motor does not find two successive reference marks (or a mechanical end stop or a limit switch) during this movement, the **MULT IDX SEARCH** error (M64=61) will appear.

**K40=16, 17, 18 or 19:**

They are specific homings where a reference mark is detected by the controller only if DIN2 input is set to 1. With an incremental EnDat 2.2, the homing process stops as soon as the reference mark is found. Consequently the accuracy of the process depends on the initial position.

K40=20:

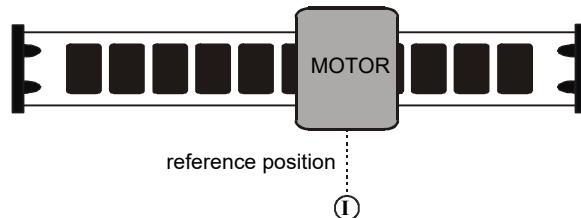
Homing with a multi-reference mark with a positive movement. The motor moves the distance defined by the parameter KL46 to find 2 successive reference marks. If the motor does not find two successive reference marks (or a mechanical end stop) during this movement, the **MULT IDX SEARCH** error (M64=61) will appear.

**K40=21:**

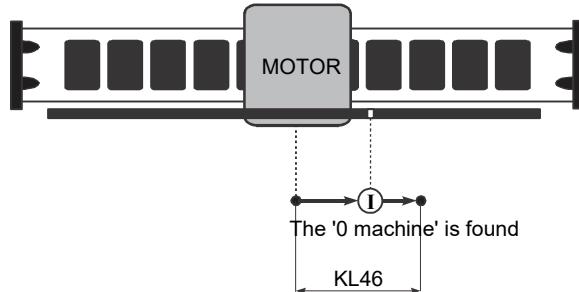
Same as K40=20 with a negative movement.

K40=22:

The present position is the reference position.

**K40=24:**

Same as K40=20 but with a single reference mark. If the motor does not find the reference mark (or a mechanical end stop) during this movement (defined by the parameter KL46), the **SING IDX SEARCH** error (M64=62) will appear.



With an incremental EnDat 2.2, the homing process stops as soon as the reference mark is found. Consequently the accuracy of the process depends on the initial position.

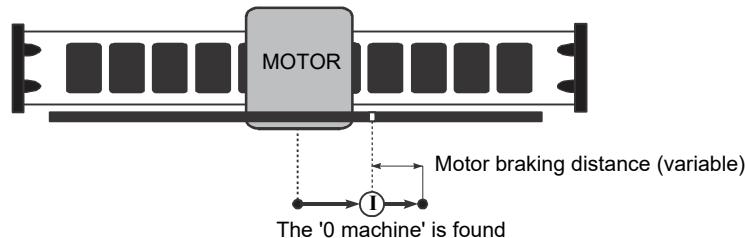
K40=25:

Same as K40=24 with a negative movement.

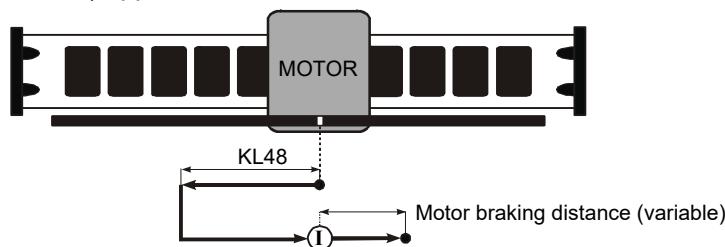
K40=26:

Homing on a mono-reference mark with a positive movement. There are three possibilities:

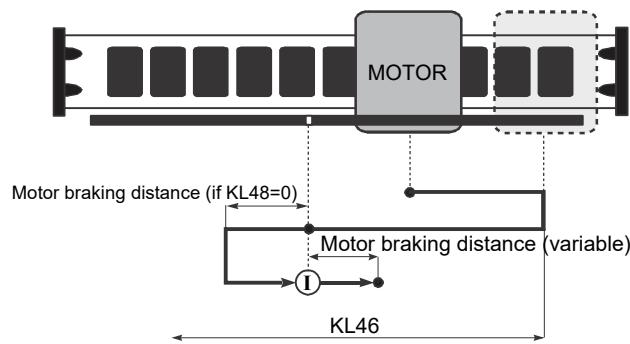
1. The motor is on the left of the index at the beginning of the homing. It moves and meets the index.



2. 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 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.



3. The motor is on the right of the index at the beginning of the homing. It moves up to the mechanical end stop and then comes back the distance given by the parameter KL46. If the index is not found during the return, the **SING IDX SEARCH** error (M64=62) appears.



With an incremental EnDat 2.2, the homing process stops as soon as the reference mark is found. Consequently the accuracy of the process depends on the initial position.

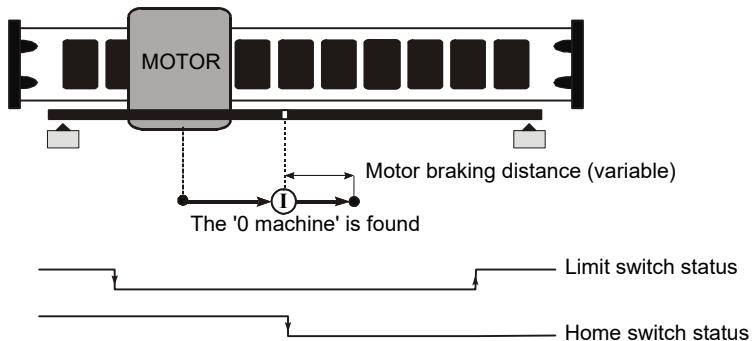
K40=27:

Same as K40=26 with a negative movement.

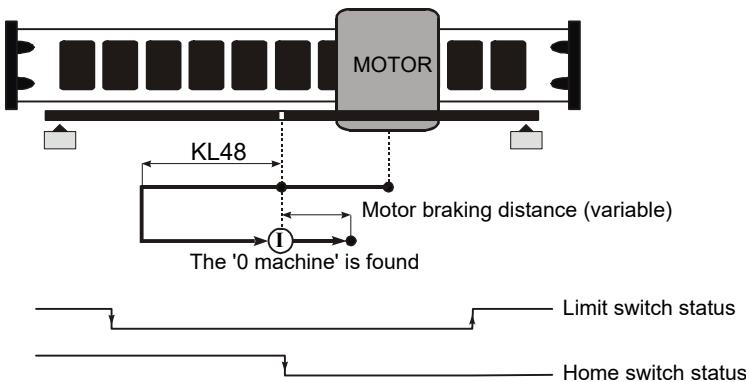
K40=34:

Homing on a mono-reference mark with a home switch and a limit switch signal. It is possible only with certain types of encoder. There are three possibilities:

- If the home switch signal is equal to 1 and on the left hand side of the reference mark, the motor moves in the positive direction up to the reference mark. If the limit switch is found before the reference mark, the **SING IDX SEARCH** error (M64=62) appears.



- If the home switch signal is equal to 1 and on the reference mark, the motor 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 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.
- If the home switch signal is equal to 1 and on the right hand side of the reference mark (due to the gap), the motor moves in the positive direction. When the homing signal changes to 0, the movement goes in the opposite direction until the reference mark is found. Once it is found, a movement in the negative direction of the distance given by the parameter KL48 is done and then another movement in the positive direction is done up to the reference mark. If the limit switch is found before the reference mark, the **SING IDX SEARCH** error (M64=62) appears.
- If the home switch signal is equal to 0, the motor moves in the negative direction until the reference mark is found. Once it is found, a movement in the negative direction of the distance given by the parameter KL48 is done and then another movement in the positive direction is done up to the reference mark. If the limit switch is found before the reference mark, the **SING IDX SEARCH** error (M64=62) appears.

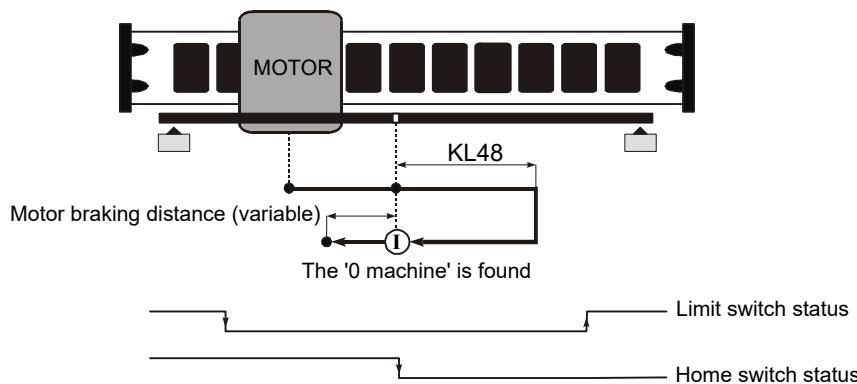


Remark: If K58≠3 when K40=34, the **HOME NOT POSSIBLE** error (M64=69) will appear instead of the **SING IDX SEARCH** error (M64=62).

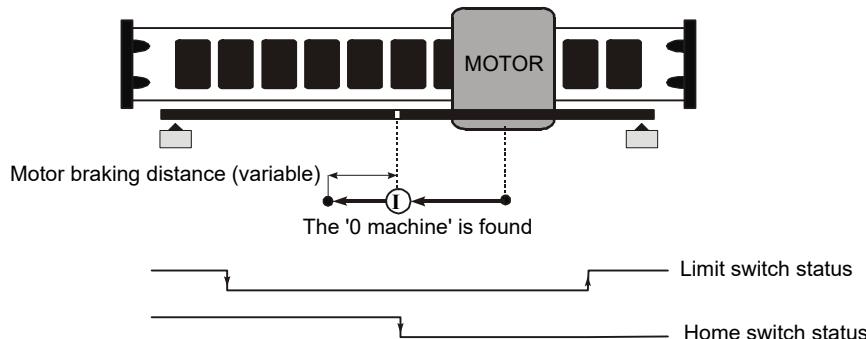
K40=35:

Homing on a mono-reference mark with a home switch and a limit switch signal. It is possible only with certain types of encoder. There are three possibilities:

- If the home switch signal is equal to 1 and on the left hand side of the reference mark, the motor moves in the positive direction up to the reference mark. When the reference mark is found, the motor keeps moving in the positive direction, the distance given by the parameter KL48 and then the motor moves back in the negative direction up to the reference mark. If the limit switch is found before the reference mark, the **SING IDX SEARCH** error (M64=62) appears.



2. If the home switch signal is equal to 1 and on the reference mark, the motor 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 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.
3. If the home switch signal is equal to 1 and on the right hand side of the reference mark (due to the gap), the motor moves in the positive direction. When the homing signal changes to 0, a movement in the negative direction is done until the reference mark is found. If the limit switch is found before the reference mark, the **SING IDX SEARCH** error (M64=62) appears.
4. If the home switch signal is equal to 0, the motor moves in the negative direction until the reference mark is found. If the limit switch is found before the reference mark, the **SING IDX SEARCH** error (M64=62) appears.

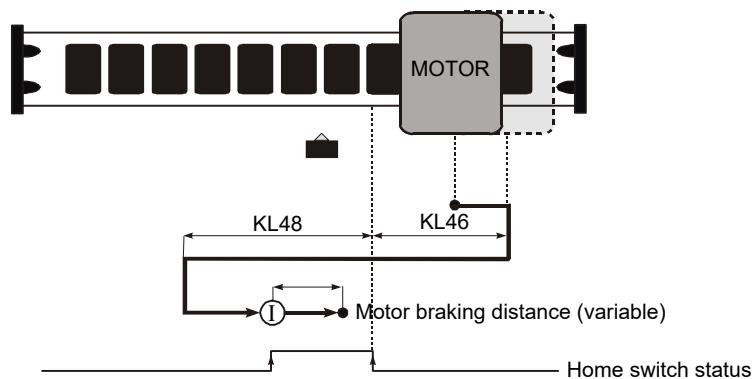


Remark: If K58≠3 when K40=35, the **HOME NOT POSSIBLE** error (M64=69) will appear instead of the **SING IDX SEARCH** error (M64=62).

K40=36:

Same as K40=2 except for case 3 which is different.

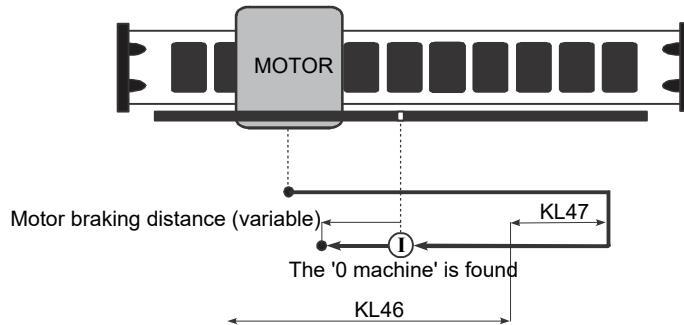
The motor is on the right of the home switch at the beginning of the homing. It moves up to the mechanical end stop and then comes back the distance given by the parameter KL46. If the home switch is not found during the return, the **SING IDX SEARCH** error (M64=62) appears. If the home switch is found (in the wrong direction), the motor moves the distance given by the parameter KL48 in the opposite direction of the homing to leave the home switch (parameter KL48 must be bigger than the home switch length). Then, it comes back towards the home switch in the right direction.

**K40=37:**

Same as K40=36 with a negative movement.

K40=38:

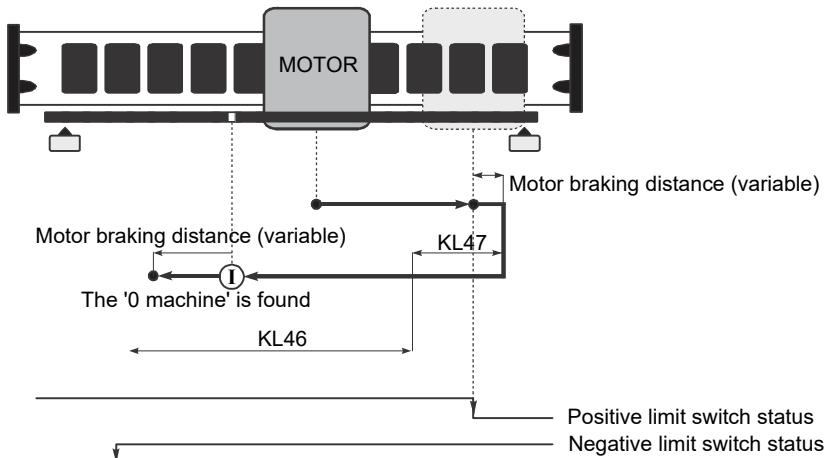
Homing on a mono-reference mark but only after having found the mechanical end stop. Once found, the motor moves back the distance given by the parameter KL47 and finds the reference mark with the distance given by the parameter KL46. If the index is not found within the distance given by the parameter KL46, the **SING IDX SEARCH** error (M64=62) appears (the distance given by the parameter KL46 must be smaller than the one between both mechanical end stop).

**K40=39:**

Same as K40=38 with a negative movement.

K40=44:

Homing on a mono-reference mark but only after having found the limit switch. Once found, the motor moves back the distance given by the parameter KL47 and finds the reference mark with the distance given by the parameter KL46. If the index is not found within the distance given by the parameter KL46, the **SING IDX SEARCH** error (M64=62) appears (the distance given by the parameter KL46 must be smaller than the one between both limit switches).



K40=45:

Same as K40=44 with a negative movement.

K40=46:

Homing without effect on the position. It is used with absolute EnDat Encoder in order to not change the position. The offset for absolute position (KL45) is not taken into account with this homing method.

7.10 Wait commands

These commands temporarily stop the execution of a sequence (normal commands) during all the waiting time and then continue from the following line. When one of the wait commands is used, only the entry used to send this command is waiting and it is not possible to execute another normal command on the waiting entry before the end of the wait. The monitoring **M101** allows the user to know which entry is waiting:

M	Name	Comment
M101	Wait entry	Bit field indicating that an entry is waiting (=1)

Example:

The following commands are in a sequence:

```
MVE .0=1000000L;
WTM=0 ;
X2 .0=0 ;
MVE .0=..... ;           // Start a movement and if X1.0=3; comes through USB before the end of the
                           // movement, then X1.0=3; is executed before the end of the movement
```

7.10.1 WTT command

The **WTT** command (**WaIT Time**) makes a pause in the sequence progression. The pause duration is set with the parameter of the command.

Command	<P1>	Comment	Units
WTT.<axis>=<P1>	Pause duration	Waits for <P1> * MLTI (400µs). Sets bit in M101 as long as it waits and then clears it	[MLTI]

Remark: If the parameter (P1) is not a 32-bit integer, the **BAD CMD PARAM** error (M64=70) occurs. Refer to [§7.10.7](#) to know how to clear a 'wait' command.

Example:

The Manager Loop Tine Interrupt (MLTI) is equal to 400 µs.

```
WTT .1=500 ;           // The pause duration is equal to 500 × 400 µs=200 ms.
```

7.10.2 WTM and WTP commands

The **WTM** command (**WaIT for end of Movement**) allows the user to wait for the end of the current theoretical movement before going on with the execution of the sequence. If several movement commands (MVE, STA, etc.) are successively sent, it is not necessary to introduce the WTM command between each movement to wait for the end of the preceding one because the controller does it automatically, unless **concatenated movements** are executed (refer to [§8.3.1.1](#)).

Command	Comment
WTM.<axis>	Waits for the end of the current movement before continuing the execution of the sequence. Sets bit in M101 during all the movements and clears it at the end of the movement

The **WTP** command (**WaIT** for **P**osition) allows the user to wait for the motor to cross the position specified in its parameter, in any direction, before going on with the execution of the sequence from the following line.

Command	<PL1>	Comment	Units
WTP.<axis>=<PL1>	Motor position	Waits for the motor to cross position <P1> before going on with the execution of the sequence. Sets bit in M101 as long as it does not reach this position and then clears it	[upi] [rupi]

Remark: If the parameter (P1) is not a 64-bit integer, the **BAD CMD PARAM** error (M64=70) occurs.
 If the specified position has not been crossed yet, and the movement ends, the command following the WTP command is executed at the end of the movement.
 If the motor is not moving when the WTM or WTP commands are sent, they are ignored. Both commands are generally preceded by the MVE or STA command because the motor must be moving.
 Refer to [§7.10.7](#) to know how to clear a 'wait' command.

7.10.3 Wait for Window: WTW command

The **WTW** command (**WaIT** for **W**indow) allows the user to define a 'window' around the target position to reach (refer to [§8.8](#) for more information about the definition of the window) when the standard reference mode is selected (K61=1) but not in interpolation mode (ITP). The WTW command stops the execution of the sequence on the used axis until the motor meets the window.

Command	<P1>	Comment
WTW.<axis>=<P1>	0	Waits for the bit 'in-window' (bit#5 of SD1) to be at 1. If the theoretical trajectory is finished (bit#4 of SD1 at 0) when this command is executed, the controller acknowledges the command without testing if the motor is in the window or not. WTW.<axis>=0 command must not be preceded by a WTM.<axis> command because the WTW command must be executed before reaching the theoretical position
	1	Waits for the bit 'in-window' (bit# 5 of SD1) to be at 1 without taking into account if the theoretical trajectory is finished or not
	2	Restarts a new test by clearing the bit 'in-window' (bit#5 of SD1) without taking into account if the theoretical trajectory is finished or not

Remark: Refer to [§7.10.7](#) to know how to clear a 'wait' command.
 Sending the WTW command without parameter, is equivalent to WTW.<axis>= 0

The monitoring **M38** allows the user to know the time between the WTW command and the setup of the bit# 21 of M63.

M	Name	Comment
M38	In-window time	Gives the time between the WTW command and the setup of the bit#5 of M60 (motor in the windows defined by K38 and K39)

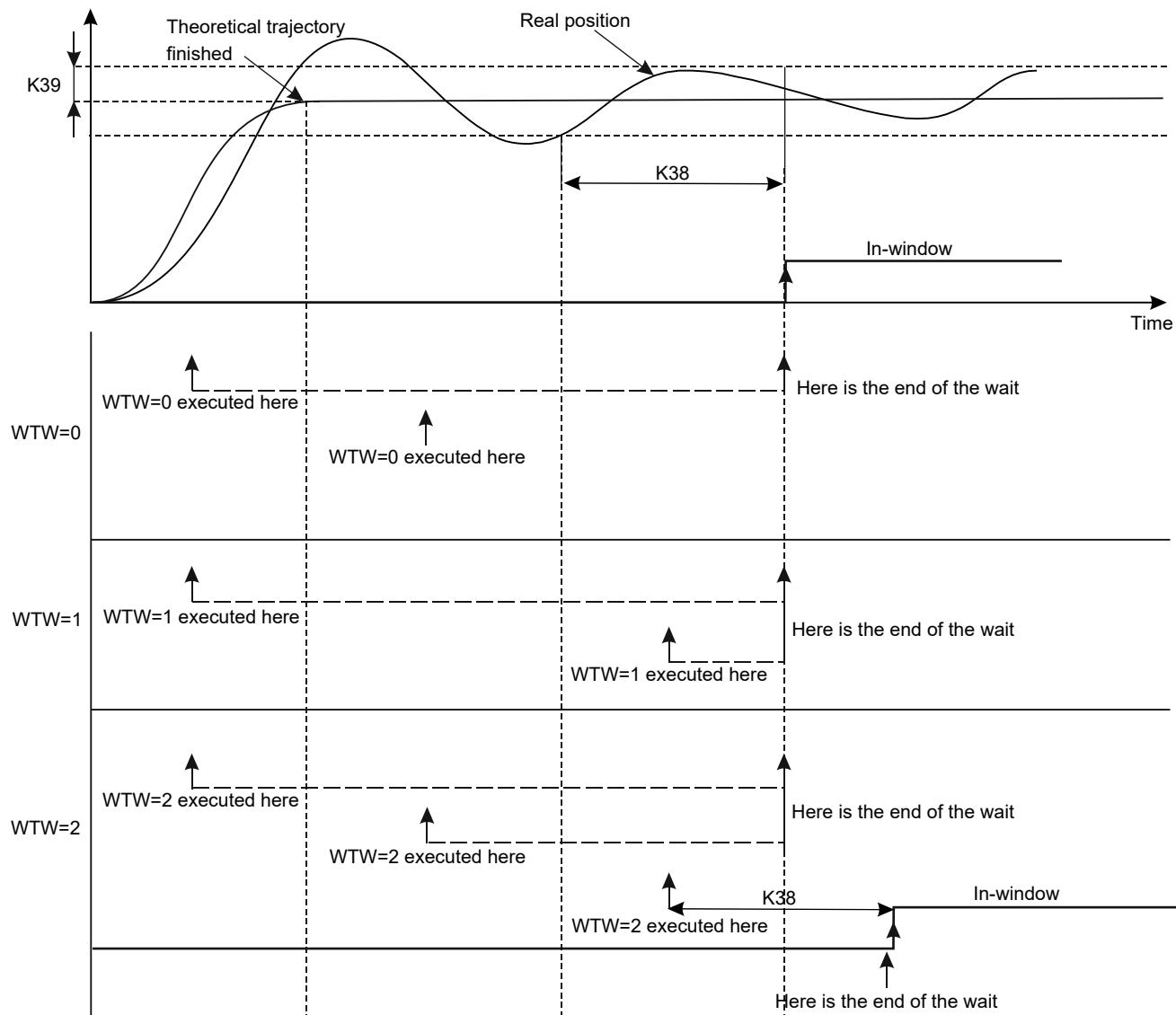
The monitoring **M138** allows the user to know the maximum tracking error during the WTW command and the setup of bit#5 of the monitoring M60.

M	Name	Comment
M138	Max. tracking error during WTW command	Gives the maximum tracking error during WTW command and the setup of bit#5 of M60 (motor in the windows defined by K38 and K39)

The monitoring **M139** allows the user to know the maximum tracking error during in-window time and the setup of bit#5 of the monitoring M60.

M	Name	Comment
M139	Max. tracking error during in-window time	Gives the maximum tracking error during in-window time and the setup of bit#5 of M60 (motor in the windows defined by K38 and K39)

Here is a graphical representation of the above-mentioned explanations:



7.10.4 WTS command

The **WTS** command (**WaIT Speed window**) allows the user to define a 'Speed Window' around the target speed (KL211.<axis>) when the standard reference mode is selected (K61.<axis>=1) and only in trapezoidal configuration (K202.<axis>=1).

Command	Comment
WTS.<axis>	Wait during a motion, when the real speed (usi) is in a window of speed (+/-KL181) during a time (K180 in MLT) around the speed target KL211

The parameters **K180** and **KL181** are needed to set this command:

K	Name	Comment	Unit
K180	Speed window time	Gives the duration of the speed window (used with WTS command)	[mlti]
KL181	Speed range window	Gives the speed range of the window (used with WTS command)	[usi]

Remark: The bit#9 of M60 as well as the bit#22 of M63 are defined to give the 'in_speed_window' information.

The WTS command is disabled during the homing process to optimize the CPU resources. Refer to [§7.10.7](#) to know how to clear a 'wait' command.

The motor is considered in the 'speed window' when the real speed (M11) is between [desired speed (KL211) - KL181] and [desired speed (KL211) + KL181] during a minimum time given by the parameter K180 without interruption.

- If the WTS command is sent when the axis is not moving, the command is immediately acknowledged and the 'in_speed_windows' bit of M60 and M63 are set to 0.
- If parameter K180 or parameter KL181 is set to 0, the command is immediately acknowledged and the 'in_speed_windows' bit of M60 and M63 are set to 0.
- If the WTS command is sent and the real speed (M11) never reaches the theoretical speed (KL211), the command is acknowledged at the end of the movement and the 'in_speed_windows' bit of M60 and M63 are set to 0.
- The CAM factor (KF205) is taken into account for the 'speed window' parameter (KL181) but nor for the time (K180).
- In gantry level 2 (C245=2), the 'in_speed_windows' bit of M60 and M63 are set to 1 only when both axes of the gantry are in the 'speed window'. In this mode, KL181 and K180 must have the same value on both axes of the gantry.

7.10.5 Wait on bits: WBS and WBC commands

The **WBS** (Wait bit Set) and **WBC** (Wait bit Clear) commands test one or several bits of X, K or M registers and go on with the execution of the sequence if the bits are set to 1 (WBS) or to 0 (WBC). The bits are numbered from the right to the left from 0 to 31.

Command	<P1>	<P2>	Comment
WBS.<axis>=<P1>,<P2>	Register to test	Mask value	Waits for the specified bits (which have their mask value included in <P2>) of the specified register <P1> to be all set. Sets bit in M101 as long as the condition is not true and then clears it
WBC.<axis>=<P1>,<P2>	Register to test	Mask value	Waits for the specified bits (which have their mask value included in <P2>) of the specified register (<P1>) to be all cleared. Sets bit in M101 as long as the condition is not true and then clears it

Remark: If at least one parameter (P1 or P2) is not an 32-bit integer, the **BAD CMD PARAM** error (M64=70) occurs.

Refer to [§7.10.7](#) to know how to clear a 'wait' command.

The field <P2> must contain the **mask** that selects the bits to be tested in the register included in the field <P1>. This mask is obtained by transforming in binary the value contained in <P2>. The bits with 1 are those tested by the WBS and WBC commands.

Here is an example with <P1> corresponding to the monitoring M50 (value of the digital inputs). If <P2>=2 then only bit#1 is tested. If <P2>=513, bit#0 and bit#9 are tested. If <P2>=773, bits 1, 3, 8 and 9 are tested:

<P2> values		DIN10	DIN9	-	-	-	-	-	DIN3	DIN2	DIN1
		Bit# 9	Bit# 8	-	-	-	-	-	Bit# 2	Bit# 1	Bit# 0
2	2	0	0	-	-	-	-	-	0	1	0
513	200	1	0	-	-	-	-	-	0	0	1
773	305	1	1	-	-	-	-	-	1	0	1
Decimal	Hexa.	Binary									

Remark: Refer to [§8.5](#) to know the digital inputs available for each controller.

Example:

WBS and WBC commands are particularly useful to test the state of one or several digital inputs. For example, to continue the execution of a sequence only if the digital input DIN2 is activated, the following command will be used :

```
WBS .1=M50.1,2;      // The sequence goes on only when DIN2 is set to 1. Every bit of the monitoring M50
                      // represents a digital input of the controller.
WBS .1=DIN.1,2;      // Same effect as above but the alias DIN is used instead of the monitoring M50.
```

Testing various inputs simultaneously by choosing the adequate mask is also possible. For example, the following command has to be used to continue the sequence when DIN1 **and** DIN2 are set to 1:

```
WBS .1=M50.1,3;      // The sequence only goes on when the digital inputs DIN1 and DIN2 are set to 1.
```

7.10.6 Wait on values: WPL, WSL, WPG and WSG commands

The **WPL** command (**Wait Parameter Lower than**) allows the user to wait for the register specified in <P1> to be lower than the value given in <P2> to continue the sequence execution. The **WSL** command (**Wait Signed Lower than**) works like the WPL command but in this case the parameters <P1> and <P2> can be signed.

The **WPG** command (**Wait Parameter Greater than**) allows the user to wait for the register specified in <P1> to be greater than the value included in <P2> to continue the sequence execution. The **WSG** command (**Wait Signed Greater than**) works like the WPG command but in this case the parameters <P1> and <P2> can be signed.

Command	<P1>	<P2>	Comment
WPL.<axis>=<P1>,<P2>	Register to test	Value of register	Waits for the register <P1> to be lower than the value of <P2>. Sets bit in M101 as long as the condition is not true and then clears it.
WSL.<axis>=<P1>,<P2>	Register to test	Value of register	Waits for the register <P1> to be lower than the value of <P2>. <P1> and <P2> can be signed. Sets bit in M101 as long as the condition is not true and then clears it.
WPG.<axis>=<P1>,<P2>	Register to test	Value of register	Waits for the register <P1> to be greater than the value of <P2>. Sets bit in M101 as long as the condition is not true and then clears it.
WSG.<axis>=<P1>,<P2>	Register to test	Value of register	Waits for the register <P1> to be greater than the value of <P2>. <P1> and <P2> can be signed. Sets bit in M101 as long as the condition is not true and then clears it.

Remark: If at least one parameter (P1 or P2) is not an 32-bit integer, the **BAD CMD PARAM** error (M64=70) occurs.

This function is not only dedicated to the movements but also to all K, M or X registers. If the values specified in <P2> are never reached by the chosen register, the pause has an infinite duration. If the condition is already met when the command is executed, the sequence goes on immediately.

The value in <P2> is only taken into account when the WPL, WSL, WPG or WSG commands are executed.

Refer to [§7.10.7](#) to know how to clear a 'wait' command.

Example:

Here is the extract of a sequence. The motor is supposed to be initially at the position 0.

```
ACC .1=500000L;          // Definition of the maximum acceleration amax.
SPD .1=200000L;          // Definition of the maximum speed vmax.
MVE .1=300000L;          // The motor moves to the position 300000 at a speed of 200000 and an
                        // acceleration of 500000 increments.
WPG .1=M11.1,10000;      // X2.1=1 is executed only when the motor real speed given by the monitoring M11
                        // is over 10000 increments.
X2 .1=1;                 // Value 1 is attributed to the user variable X2.
```

7.10.7 Clear wait commands

All the 'wait' commands described in [§7.10](#) temporarily stop the execution of other normal commands on the same entry (of the 'wait' command) during all the waiting time. During this waiting time, the entry is in a busy state. The **CLRWAIT** command allows the user to clear a pending 'wait'.

Command	Comment
CLRWAIT.<axis>=<P1>	Clear busy state due to a wait command. The definition of <P1> is the same as for the monitoring M101 (Wait entry; refer to §7.10) except for wait coming from the sequence (bit#5 to 7). The bit#5 to 7 which are reserved must be set to 0.

The CLRWAIT command could be sent as a normal command as well as an urgent command. The advantage of the urgent command is that this command will not be blocked by a previous pending 'wait' command (if present) on the same entry.

7.11 Emergency stop: HLT, HLB, HLO

The three commands HLT, HLB and HLO allow the user to **stop the movement (not in interpolated mode)** of the motor as well as the **execution** of the **user sequence**.

The **HLT** command (**HaLT** sequence and movement) stops the movement with the maximum deceleration that the controller can supply and interrupts the current sequence execution. In current mode (K63.<axis>≠0), the **TORQUE MODE ERR** error (M64=28) occurs if the HLT command is sent.

Remark: The sudden deceleration generated by the HLT command, may seriously damage the mechanical parts of the system.

The **HLB** command (**HaLt** sequence and **Brake**) stops the movement without overrunning the authorized motor maximum deceleration defined by the parameter **KL206** and interrupts the current sequence execution. In current mode (K63.<axis>≠0), the **TORQUE MODE ERR** error (M64=28) occurs if the HLB command is sent.

The **HLO** command (**HaLt** sequence and power **Off**) interrupts the current sequence execution and switches off the power in the motor phases (PWR.1=0).

Remark: The motor may **keep moving** for a while, depending on the system's moving part inertia.

It is advised to use the HLO command only if the controller is set with a relay short circuiting the phases of the motor when the power is switched off to have a magnetic brake.

Command	Comment	Read parameters
HLT.<axis>	Stops the sequence depending on K144 value and brakes the motor movement using an infinite deceleration	-
HLB.<axis>	Stops the sequence depending on K144 value and brakes the motor using programmed deceleration given by KL206	KL206
HLO.<axis>	Stops the sequence depending on K144 value and performs a power off	-

For the HLT command, the deceleration is in fact not necessarily infinite as it depends on the speed and the jerk time of the current movement. This value can be calculated as follows:

$$\text{Deceleration } [\text{m/s}^2] = \frac{\text{M10 } [\text{m/s}](\text{at the command execution})}{(\text{MLTI } [\text{s}] \times \text{K213 } [\text{Inc}])}$$

If K213=0 Inc, the controller will use the same formula but with K213=1 Inc.

Remark: Refer to [§15.5](#) for more information about the use of the HLT, HLB et HLO commands with the parameter K144.

7.12 Basic movements

A movement is made between two points, on a trajectory limited by the speed, the acceleration and the jerk. The basic movements features, described in this part of the manual, are using only **S-curve** and Rotary **S-curve** movements types. Advanced users may also refer to [§8.3](#), for other movements types description.

7.12.1 Zero machine

7.12.1.1 Main encoder

The **SET** command (**SET '0 machine'**) defines the motor current position value.

Command	<PL1>	Comment	Units
SET.<axis>=<PL1>	- 2^{47} to $(2^{47}-1)$	'0 machine' positioning.	[upi], [rupi]

This command is generally used at the beginning of a sequence (after the homing) to place the '0 machine' in a different position than the reference mark because after a homing, the '0 machine' is automatically placed where the reference mark is (in case of a mono-reference mark encoder). This command must be used if the power is on and K61=1 but not in interpolated mode (ITP).

Remark: After the homing, the '0 machine' is set on the reference mark but **the motor does not exactly stop on it** because of the motor braking distance. This distance is determined by its speed when it crosses the reference mark and with parameters KL41 and KL42 (homing speed and acceleration). Therefore it is really important to execute a MVE command (MVE.1=0L, for instance) to place the motor in a precise pre-defined position before starting the SET command.

When the SET command is used, it does not change the value of the KL34 and KL35 parameters (to avoid software limits changes). However, it does not change the internal values and in that case, monitorings **ML434** and **ML435** allow the user to know the position of the software limits.

M	Name	Comment	Units
ML434	Min. software position limit	Gives the minimum software position limit	[upi]
ML435	Max. software position limit	Gives the maximum software position limit	[upi]

After an IND command, ML434=KL34 and ML435=KL35. After a SET command, ML434 is not equal to KL34 and ML435 is not equal to KL35.

7.12.1.2 Secondary encoder

The **OFFSETSEC** command allows the user to add an offset on the secondary encoder position given by ML13.

Command	<PL1>	Comment
OFFSETSEC.<axis>=<PL1>	- 2^{47} to $(2^{47}-1)$	Apply an offset on the secondary encoder position with the value given by <PL1>

Remark: The OFFSETSEC command send without parameter (<PL1>) is equal to **OFFSETSEC.<axis>=0**.

M	Name	Comment	Units
ML33	Secondary position offset	Gives the secondary position offset value applied	[dpi2]

7.12.1.3 Auxiliary encoder

The **OFFSETAUX** command allows the user to add an offset on the auxiliary encoder position given by ML15.

Command	<PL1>	Comment
OFFSETAUX.<axis>=<PL1>	-2 ⁴⁷ to (2 ⁴⁷ -1)	Apply an offset on the auxiliary encoder position with the value given by <PL1>

Remark: The OFFSETAUX command send without parameter (<PL1>) is equal to OFFSETAUX.<axis>=0.

M	Name	Comment	Units
ML35	Auxiliary position offset	Gives the auxiliary position offset value applied	[dpi3]

7.12.2 Linear or rotary movement

The parameter **K202** (also available via the **MMD** alias) defines the **type of movement**.

K202=1 (or MMD=1): defines a linear movement (S-curve movement)

K202=17 (or MMD=17): defines a rotary movement (rotary S-curve movement)

Remark: These two movements are available only with the standard position reference mode (K61=1) and not in interpolated mode (ITP). Refer to [§8.3.1.1](#) for more information.

It is not possible to enable with K36 the software limits when the MMD mode is greater or equal to 17. In this case, the **ERR SOFT LIMIT** error (M64=80) is raised. When the error is set, K36 is forced to 0.

7.12.3 Trajectory parameters and start movement

The parameters **KL211**, **KL212** and **K213** allow the user to memorized the maximum speed, the acceleration and the jerk time values when an (S-curve) movement is executed. They define the movements trajectory and are available with K61=1 but not in interpolated mode (ITP).

K	Alias	Name	Comment	Units
KL210	TARGET	Target position	Gives the target position, with TARGET command.	[upi], [rupi]
KL211	SPD	Maximum speed	Pre-programmed maximum speed, with SPD command	[usi], [rusi]
KL212	ACC	Max. acceleration	Pre-programmed maximum acceleration, with ACC command	[uai], [ruai]
K213	JRT	Jerk time	Pre-programmed jerk time, with JRT command	[MLTI]

Remark: TARGET, SPD, ACC and JRT are alias of these parameters and use the same syntax.

- The **TARGET** command defines the position x_{final} to reach during a movement (it does not start the movement). The position is available on 48 bits used only with STA and STI commands (refer to [§8.3.5](#))
- The **SPD** command (**SPeeD**) defines the maximum speed v_{max} for a movement. The speed in [usi] and [rusi] is limited to $(2^{31} * (MLTI/PLTI)) - 1$. For example, 17179869183 with the PLTI running at 50 μs and the MLTI at 400 μs .
- The **ACC** command (**ACCeeration**) defines the maximum acceleration a_{max} during a movement. The acceleration in [uai] and [ruai] is limited to $(2^{31} * (MLTI/PLTI)^2) - 1$. For example, 137438953471 with the PLTI running at 50 μs and the MLTI at 400 μs .
- The **JRT** command (**JeRk Time**) defines the jerk time value for an S-curve movement. This is not exactly the derivative part of the acceleration (i.e. the jerk), but the time for the acceleration to reach its defined

value. The acceleration variation is thus slowed down and the complete S-curve movement is lengthened by the value of the jerk time. JRT corresponds to the length in MLTI increments of the filter smoothing the acceleration. Consequently, an effect is only visible from the value JRT=2 increments, JRT=0 or 1 are equivalent and have no impact. The JRT max value is Tmax=500x400 μs =200 ms (with a MLTI set to 400 μs).

Here is the formula giving the jerk (J) according to the acceleration (A) and the jerk time (T) with ISO units:

$$J \text{ [m/s}^3\text{]} = \frac{A \text{ [m/s}^2\text{]}}{T \text{ [s]}}$$

Remark: The jerk (J) is not always the same. If the movement reaches a constant speed, the above-mentioned formula can be applied. However, if a constant speed is not reached, the following formula must be used:

$$\frac{A \text{ [m/s}^2\text{]}}{T \text{ [s]}} < J \leq 2 \cdot \frac{A \text{ [m/s}^2\text{]}}{T \text{ [s]}}$$

The **MVE (MoVEment)** command allows the user to make an absolute movement. It is also possible to give the parameters KL211 (SPD), KL212 (ACC) and K213 (JRT) with only one command. With this command, it is possible to modify the final position, the speed and the acceleration even if the motor is already moving. However, in concatenated movement (MMC=1), if the motor is moving when the command is sent, the new jerk time value is ignored and the "JERK TIME IGNORED" warning (M66=28) is set as long as the motor moves.

Command	<P>	Comment
MVE.<axis>=<PL1>[,<PL2>[,<PL3>[,<P4>]]]	<PL1> <PL2> <PL3> <P4>	Target position corresponding to KL210 at depth 0 (-2 ⁴⁷ to 2 ⁴⁷ -1) Speed for the movement (optional parameter) corresponding to KL211 (at depth 0) Acceleration for the movement (optional parameter) corresponding to KL212 (at depth 0) Jerk time for the movement (optional parameter) corresponding to K213 (at depth 0)

Remark: Only integer 64-bit can be used for PL1, PL2 and PL3, and integer 32-bit for P4
If a CAM is activated (refer to §8.3.7), the speed, acceleration and jerk time will be modified.

The monitoring **M163** gives the different stages of a movement as follow:

M	Name	Comment
M163	State motion for advanced Gain Scheduling	1: theoretical movement is on-going 2: motor is in settling phase 3: motor is in window

It can be used for example with the function "Advanced Gain Scheduling" for adapting the gains depending on the motion phase.

Example:

```
PWR.1=1;           // Current is supplied in the phases (after a phasing, if it was the 1st PWR).
IND.1;             // The motor moves up to the reference mark.
WTM.1;             // Waits until the movement is finished.
MVE.1=0L,200000L,500000L; // The motor moves exactly on the reference mark with the maximum speed
                           vmax and the maximum acceleration amax
WTM.1;             // Waits until the movement is finished.
MMD.1=1;           // Selects a linear S-curve movement.
JRT.1=200;          // Definition of the jerk time: =200x400  $\mu\text{s}$ =80 ms
```

Until now the motor is still positioned on the reference mark.

```
MVE.1=300000L;    // The motor moves to position 300000 [upi] with a speed of 200000 [usi] and an
                   // acceleration of 500000 [uai].
WTM.1;             // Waits until the movement is finished.
```

```

MVE.1=10000L; // The motor moves to position 10000 [upi] with the same speed and acceleration than
// before.

WTM.1; // Waits until the movement is finished.

MVE.1=-15000L; // The motor moves to an absolute negative position -15000 [upi] with the same speed
// and acceleration than before.

WTM.1; // Waits until the movement is finished.

```

7.12.3.1 Relative movements

The **RMVE** (Relative MoVEment) command allows the user to make a relative movement from the current position. It is also possible to give the parameters KL211 (SPD), KL212 (ACC) and K213 (JRT) with only one command. With this command, it is possible to modify the final position, the speed and the acceleration even if the motor is already moving. However, in concatenated movement (MMC=1), if the motor is moving when the command is sent, the new jerk time value is ignored and the "JERK TIME IGNORED" warning (M66=28) is set as long as the motor moves.

Command	<P>	Comment
RMVE.<axis>=<PL1>[,<PL2>[,<PL3>[,<P4>]]]	<PL1> <PL2> <PL3> <P4>	Target position corresponding to KL210 at depth 0 (-2 ⁴⁷ to (2 ⁴⁷ -1)) Speed for the movement (optional parameter) corresponding to KL211 (at depth 0) Acceleration for the movement (optional parameter) corresponding to KL212 (at depth 0) Jerk time for the movement (optional parameter) corresponding to K213 (at depth 0)

RMVE.1=300000L **Relative movement:** The motor moves in the positive direction of 300000 increments with respect to its actual position.

RMVE.1=-300000L **Relative movement:** The motor moves in the negative direction of 300000 increments with respect to its actual position.

Caution: **When converting ISO value to increment value, a position shift can occur with relative movements due to the rounding error of the ISO conversion.**

Example:

If 10cm=4275.5 increments, the ISO conversion will give 4276. Here is a process performing 4 x (+10cm) and 1 x (-40cm) on the axis 1, starting from 0 position.

```

RMVE.1=0.1; // The absolute position 4276 increments.

WTM.1;

RMVE.1=0.1; // The absolute position 8552 increments (instead of 8551).

WTM.1;

RMVE.1=0.1; // The absolute position 12828 increments.

WTM.1;

RMVE.1=0.1; // The absolute position 17104 increments (instead of 17102).

WTM.1;

RMVE.1=-0.4; // The corresponding position is -17102 increments so the motor will be at +2 increments
// and not at 0 as expected.

```

To avoid such a problem, the user can work with increment value, knowing that 10cm cannot exactly be reached.

```

RMVE.1=4275L; // Rounding error.

WTM.1;

RMVE.1=4276L; // The absolute position is correct (8551 increments).

WTM.1;

RMVE.1=4275L; // Rounding error.

WTM.1;

RMVE.1=4276L; // The absolute position is correct (17102 increments).

WTM.1;

RMVE.1=-17102L; // The motor comes back at 0 position.

```

Or the best solution is to perform an absolute movement to be able to come back at the 0 position by replacing the RMVE.1=-17102L; by MVE.1=0L;

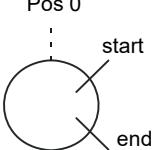
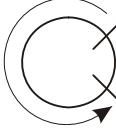
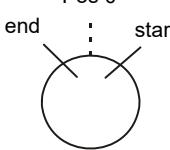
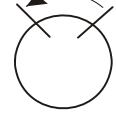
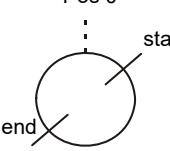
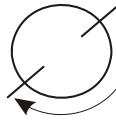
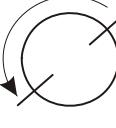
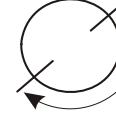
7.12.4 Rotary S-curve movement

When a rotary movement type is selected ($MMD > 16$), it is necessary to define the rotation way with the parameter **K209** (ClockWise (CW) or Counter-ClockWise (CCW)) and the controller position counter's limit value with the parameter **KL27** (refer also to [§7.4.1](#)).

Remark: These parameters are also used for advanced rotary movements types (refer to [§8.3](#) if you are an advanced user).

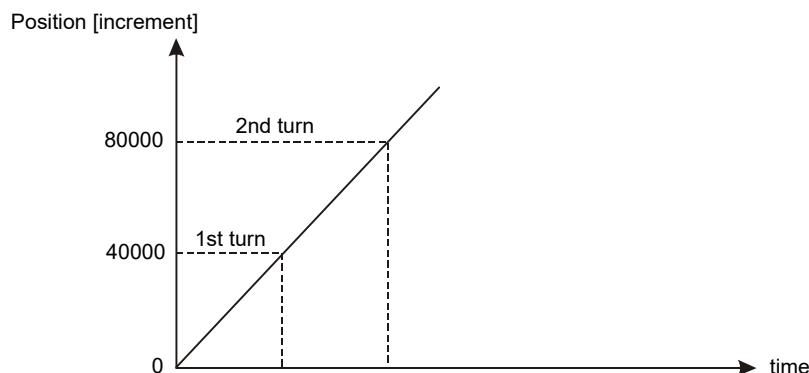
K	Name	Value	Comment
K209	Rotation way selection	0 1 2	Positive (CW) rotary movement Negative (CCW) rotary movement Shortest way to reach the target (CW or CCW)

7.12.4.1 Rotation way (parameter K209)

	K209=0 Positive (CW)	K209=1 Negative (CCW)	K209=2 Shortest (CW or CCW)
Pos 0 			
Pos 0 			
Pos 0 			

7.12.4.2 Position counter's limits (parameter KL27)

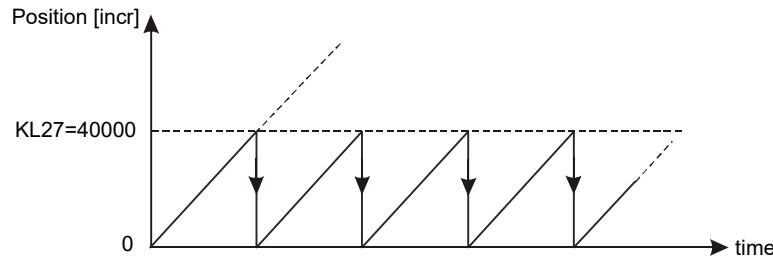
A maximum limit is required for the position counter (refer to [§7.4.1](#)). For a rotary motor revolving at constant speed:



The measured position increases continuously and after a lapse of time, it will be so big that the controller's

position counter will not be able to handle it (counter overflow). To avoid it, as soon as the motor reaches the position programmed in the parameter KL27, the position counter is brought back to the value 0.

If KL27= 40000L (user defined number of increments), the position measurement versus time will be:



Remark: It is recommended to set the parameter KL27 to the value corresponding to one complete machine cycle.

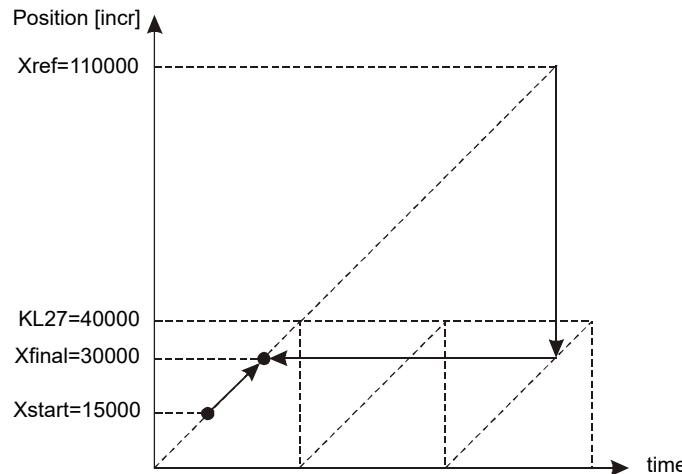
If a position target is given over the limit ($|Xref| > KL27$), it is brought back to a value: $0 < Xfinal < KL27$.

Example 1:

Xstart=15000L

MVE.1=110000L or RMVE.1=+95000L (means: Xref=110000L > KL27=40000L)

Xfinal=30000L (brought back to a value < KL27)

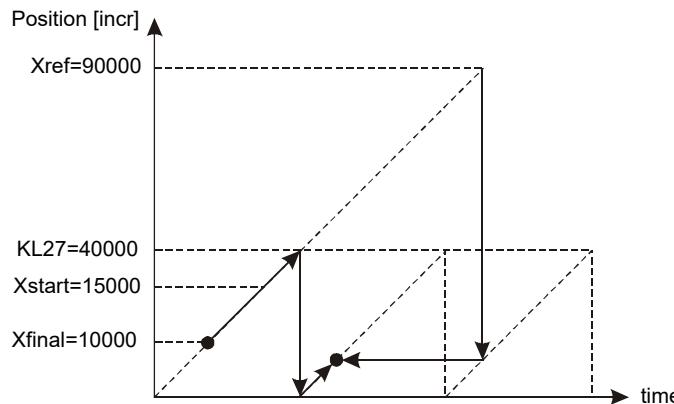


Example 2:

Xstart=15000L

MVE.1=90000L or RMVE.1=75000L (means: Xref=90000L > KL27=40000L)

Xfinal=10000L (brought back to a value < KL27)



7.13 Controller software characteristics

The following monitorings indicate useful data for the user (they cannot be modified).

M	Alias	Name	Comment	Units
M70	CTYP	Controller type	Gives the type of controller. Depth0: Product type (32 for AccurET Modular 400/600, 33 for AccurET Modular 48, 34 for AccurET Modular 300, 36 for AccurET VHP 48, 39 for AccurET VHP100, 42 for AccurET VHP 48 ZxT); depth1: Hardware version. For the complete product codification, refer to M85.	-
M71	-	Controller boot software version	Gives the software boot version of the controller (format is similar to M72)	-
M72	VER	Controller firmware version	Gives the firmware version of the controller	-
M73	SER	Controller serial number	Gives the serial number of the controller	-
M85	ARTICLE	Controller article number	Gives the article number string. The 25 strings of the article number are read using 7 depths of M85. Each depth shows 4 strings (in ASCII)	-
M86	-	Active communication mode	Gives the active communication mode defined with register C1 (Manager mode): M86=0 for TransnET, M86=1 for TCP/IP, M86=2 for EtherCAT. This monitoring is updated during the boot of the drive and is not affected by a modification of C1 during the runtime.	
M87	-	Controller axis number	Gives the current axis number seen on the communication interface TransnET, USB or TCP/IP.	-
M88	-	DIP switch axis number	Gives the current state of the DIP switch	
M89	-	Flash axis number	Gives the axis number defined by the AXI command and saved in the flash memory. As long as no AXI command is processed, M89 is equal to 0.	
M90	CTEMP	Controller temperature	Gives the temperature of the controller measured by a thermostat on the heat sink	[°C]
M91	UBUS	Controller +Vpower measurement	Gives the DC power voltage (Vpower) Read M91: +Vpower[V]=M91/100	100·[V]
M95	-	Display's string	Shows the strings on the software display. The 16 strings of the controller display are read using the 4 depths of M95. Each depth shows 4 strings (in ASCII)	-

The **CTYP** (Controller TYPe), **VER** (VERsion number), **SER** (SERIAL number), **ARTICLE** (ARTICLE number), **CTEMP** (Controller TEMperature) and **UBUS** commands are alias of these monitorings and use the same syntax. When using the **VER** command, the controller sends a firmware version, as a hexadecimal number, under the following form:

VER.1=0x**WWWXXYY**, with the following possible values:

	WWW: version # (3 digits)				XX: revision index (2 digits)										YY: status (2 digits)	
Values examples	100	...	120	...	00	01	...	40	41	...	80	81	82	...	00	80
Meaning	1	...	1.20	...	α_0	α_1	...	β_0	β_1	...	A	B	C	...	released	in dev.

For example, the hexadecimal value 0x**2008000** means that the firmware 2.00A, in released version, is in the controller.

The monitoring **M90** shows the controller temperature in Celsius degrees. If the temperature inside the controller is greater than a defined limit (refer to the table below), the **W OVER TEMPERAT** warning (M66=2) or the **OVER TEMPERAT** error (M64=5) will appear. If the thermostat inside the controller is faulty, the **SENSOR TEMP ERR** error (M64=13) will appear.

The temperature limits of functioning are defined as follow:

Device	Warning limit	Error limit
AccurET 48	75 °C	85 °C
AccurET 300	70 °C	80 °C
AccurET 400/600	69 °C	79 °C
AccurET VHP 48	70 °C	80 °C
AccurET VHP 100	60 °C	65 °C

Remark: This sensor is deactivated when the **BRIDGE FAULT** error (M64=131) occurs.

The parameter **K66** allows the user to display on the ComET software the following information:

K	Name	Value	Comment
K66	Display mode	1	Displays normal information
		2	Displays the temperature [°C] of the controller
		4	Displays the amplitude of the encoder's analog signals
		32	Displays DC power voltage (Vpower) [V]

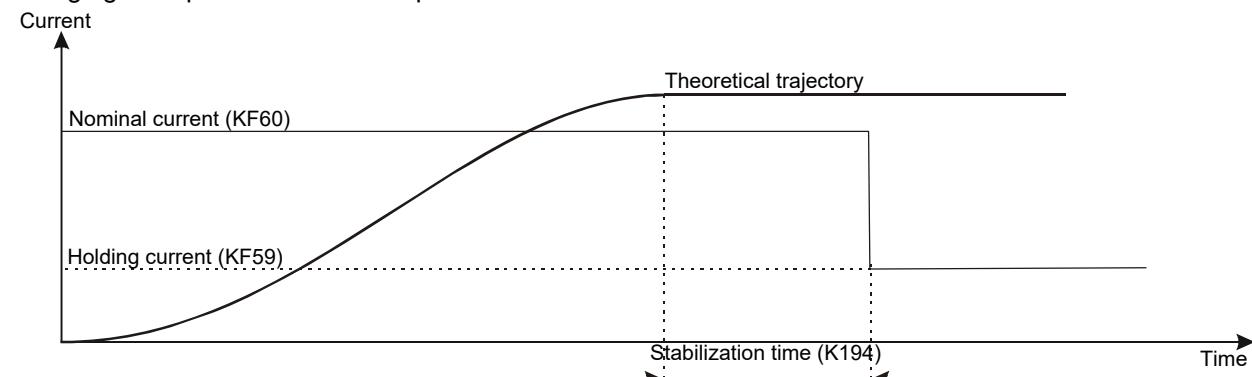
7.14 Stepper in open loop

It is possible to manage two or three-phase stepper motor in open loop which means that an encoder is not needed. It is then not possible for the user to know if some steps have been lost or not and all the registers and commands referring to the real position cannot be used any more. The stepper mode is activated by the parameter **K78**.

K	Name	Value	Comment
K78	Position controller mode	0	State regulator
		1	Stepper mode

Remark: The parameter K78 is only read when the controller is **switched on**. This parameter must be saved with the SAV command when it is changed and then the controller must be switched off and on to integrate this new data.

During the movement in standard position reference mode (K61=1 and not in interpolation mode (ITP)), a current called 'nominal current' and set by the parameter **KF60** is injected into the phases of the motor to follow the theoretical trajectory. When the movement is finished, another current called 'holding current' and set by the parameter **KF59** is injected into the phases of the motor to keep the position. It means there is always current present in a stepper motor when it is powered on. The holding current must then be high enough to keep the position of the motor but not too high to avoid higher temperature in the phases of the motor. At the end of the theoretical trajectory, it is possible to set a stabilization time with the parameter **K194** before changing from parameter KF60 to parameter KF59.



The current is injected into the motor as soon as the motor is in 'Power On' (PWR.<axis>=1). The 'moving' bit (bit#4 of SD1 alias of monitoring M60) is set to 1 during the movement and the stabilization time. In external reference mode (K61≠1 or K63=1, refer to [§8.2](#) for more information), **only** the parameter KF60 is taken into account and the user has to change its value during the movement.

In one turn, the number of drive increment [dpi], given by the parameter K55 or KL55 is equal to $2048 \times p$ with p is the number of pairs of poles (this number of pairs of poles (for a two-phase stepper, one pair of pole represents 4 steps) is given by the parameter K54). Then, $1 \text{ dpi} = 360^\circ / (p \times 2048) = 360^\circ / (K54 \times 2048)$.

Example:

For a two-phase stepper:

$K54 = \text{Step number per turn} / 4$, $K55 / KL55 = K54 \times 2048$ and $K241 = \text{Step number per turn} / 2$

- **Phasing mode**

The only phasing mode available with stepper in open loop is the phasing by constant current (K90=2) which means that parameters KF92, KF93, KF94 and KF97 must be set correctly (refer to [§7.7.2.2](#) for more information). During the phasing with stepper in open loop, the motor will move of maximum one magnetic period.

- **Homing mode**

The homing mode is normally used to define an absolute position. As there is no encoder used with stepper in open loop, the only homing mode which can be used are as follows:

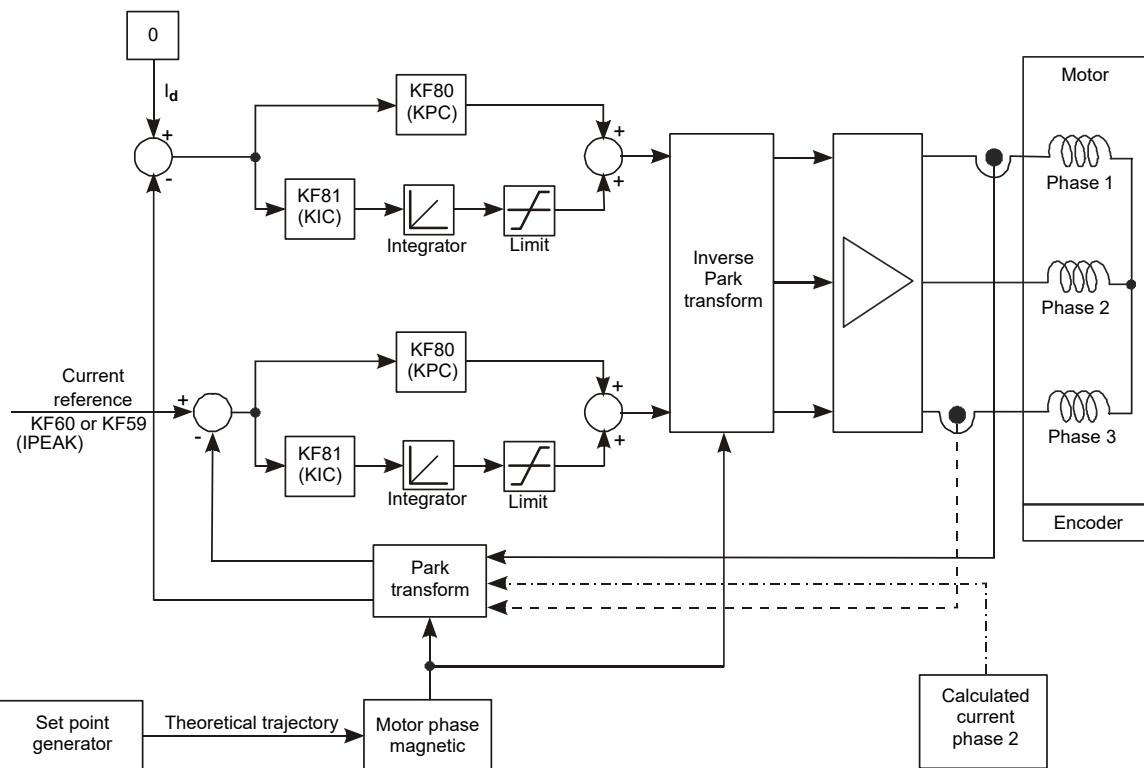
- Homing mode with home switch: K40=6 and 7 (refer to [§7.9.2](#) for more information)
- Homing mode with limit switch: K40=4 and 5 if bit#0 of parameter K58 is equal to 1 (refer to [§7.9.2](#) for more information)
- Immediate homing: K40=22 (refer to [§7.9.2](#) for more information)

The homing is made on the PLTI interrupt (refer to [§1.](#) for more information). The home switch and limit switch are tested at each PLTI. It is not possible to make a homing on mechanical end stops as there is no way to detect them.

- **Reference mode**

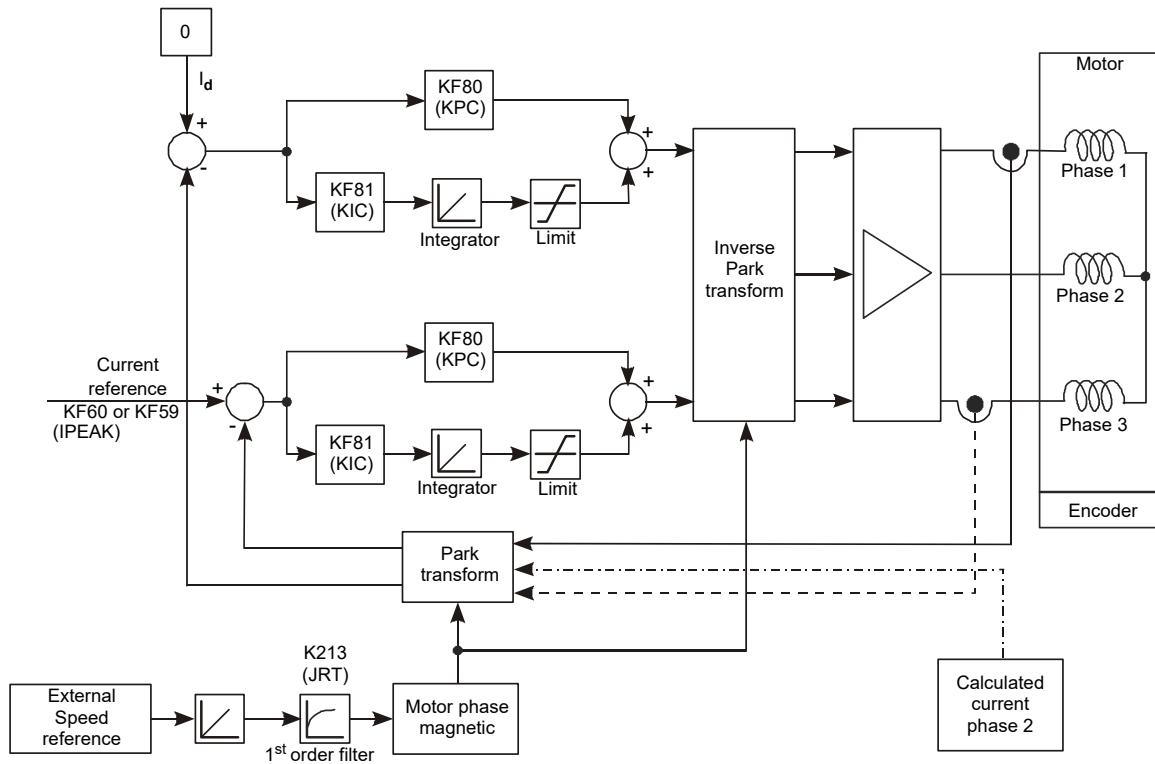
With stepper in open loop, the reference mode (defined by parameter K61) which can be used are:

- Standard position reference mode: K61=1 but not in interpolation mode (ITP (refer to [§7.6](#) for more information))



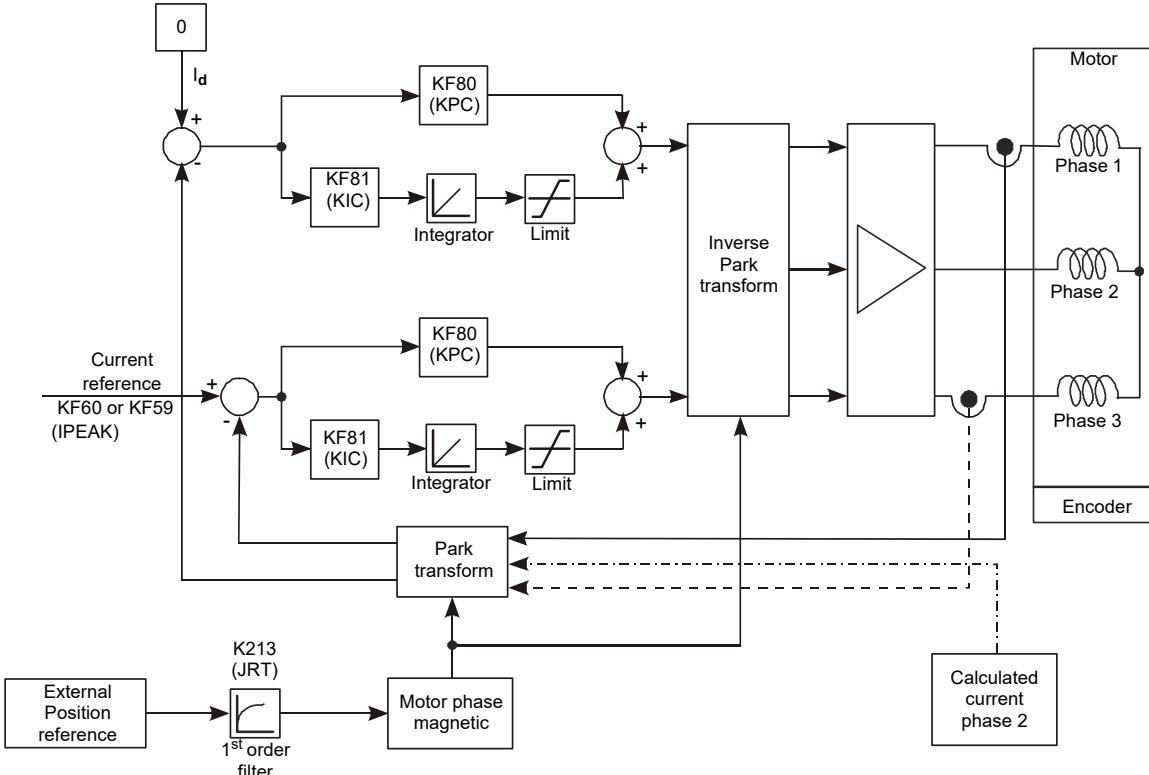
Remark: The regulation loop principle is identical with a three-phase stepper motor.

- Speed reference mode: K61=3 (refer to [§8.2.1](#) for more information)



Remark: The regulation loop principle is identical with a three-phase stepper motor.

- External position reference mode: K61=36 or 4 (refer to [§8.2.1](#) for more information)



• Other parameters

Some parameters (other than those mentioned above), previously described in this manual, must be used with stepper in open loop:

- Commutation look-up table parameters K52 and K53 must be equal to 0 (refer to [§7.7.1.4](#) for more information)

- Force inversion given by the parameter K56 (refer to [§7.2](#) for more information). Changing the force in stepper mode as the same effect than changing the movement direction
- Encoder interpolation shift value given by the parameter K77 must be equal to 0
- Current loop parameters given by parameters KF80 and KF81 (refer to [§8.1.2.5](#) and [§8.1.2.4](#) for more information)
- Current limits given by parameters KF83, KF84 and KF85 (refer to [§7.4.2](#) for more information)
- Motor phase number given by the parameter K89 must be equal to 20, 21 or 30, 31, 34 (refer to [§7.2](#) for more information)
- The number of period per turn given by parameter K241 which is equal to K54 x 2 (refer to [§7.3](#) for more information)

7.15 Power Sag Detection

The Power Sag Detection is a feature allowing a (nearly) normal functioning of the motors in case of sudden and short power voltage drop. When such a failure occurs, the Power Sag Detection temporarily reduces an on-going acceleration to keep enough energy in the controller to avoid an **UNDER VOLTAGE** error (M64 = 9). Common parameters **C100**, **C101** and **C102** must be used for the setting of this feature.

The sag detection is based on the signal PS RDY coming from the power supply (refer to the corresponding "Hardware Manual" for more details). Therefore, this feature is only available on AccurET 300 and AccurET 400/600, as only these products have a power supply provided by ETEL.

Common parameter C100 enables the feature and defines the maximum allowed sag time during which the acceleration is reduced. As soon as a voltage drop is detected, the **PWR SAG DETECTED** warning (M66 = 14) is raised. If the normal power voltage is not recovered after the C100 time is elapsed, the **POWER SAG** error (M64 = 250) is raised.

By default, an on-going acceleration is temporarily stopped (i.e. set to 0) as soon as a voltage drop is detected. The speed is then maintained constant as long as the power voltage is not back at his normal value. However, it is also possible to reduce the speed during the voltage drop to keep more energy in the controller. This can be done thanks to the common parameter C101.

In addition to the PS RDY signal, it is also possible to detect a power failure by monitoring the controller input voltage M91. This mode is enabled with the common parameter C102 and his advantage is to be able to react to small voltage drops. However, this mode has a slower detection time for full voltage drops than the PS RDY signal.

C	Name	Comment
C100	Maximum Power Sag duration	Enables the feature and defines the maximum sag duration (in MLTI increments). The feature is disabled if C100 = 0.
C101	Power Sag speed reduction	Speed reduction in percent during the voltage sag. This reduction is only applied if the motor is accelerating when the sag is detected.
C102	Power Sag voltage threshold	Voltage threshold triggering the power sag protection when input voltage M91 becomes smaller than C102. Value in [V/100].

The monitoring register M370 (the Power Sag Detection timer) is available to see when the feature starts to count and how long the sag lasts. This counter, in MLTI increments, is reset as soon as the power sag ends.

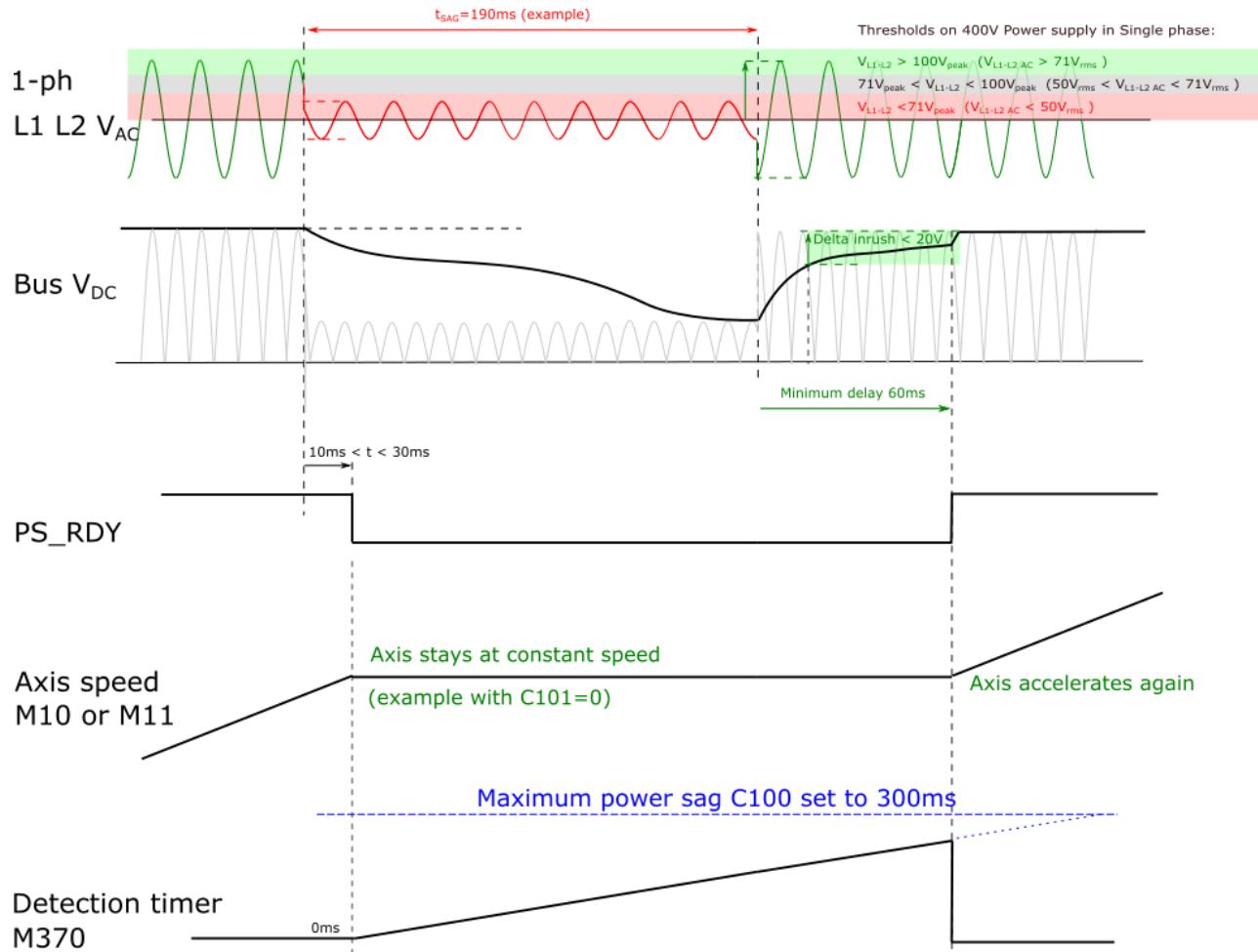
M	Name	Comment
M370	Power Sag Detection timer	The Power Sag Detection timer gives the value of the internal counter of the Power Sag feature.

Remark:

- If a command MVE or RMVE is sent during a power sag and the axis is not moving, the start of the motion is postponed as long as the power sag is present. Due to this delay, the command manager is on hold and it is not possible to send another command to the axis (except urgent commands).

- If a power sag occurs during the first 5 ms of a new motion, the acceleration will remain until the first 5 ms of the movement are elapsed.

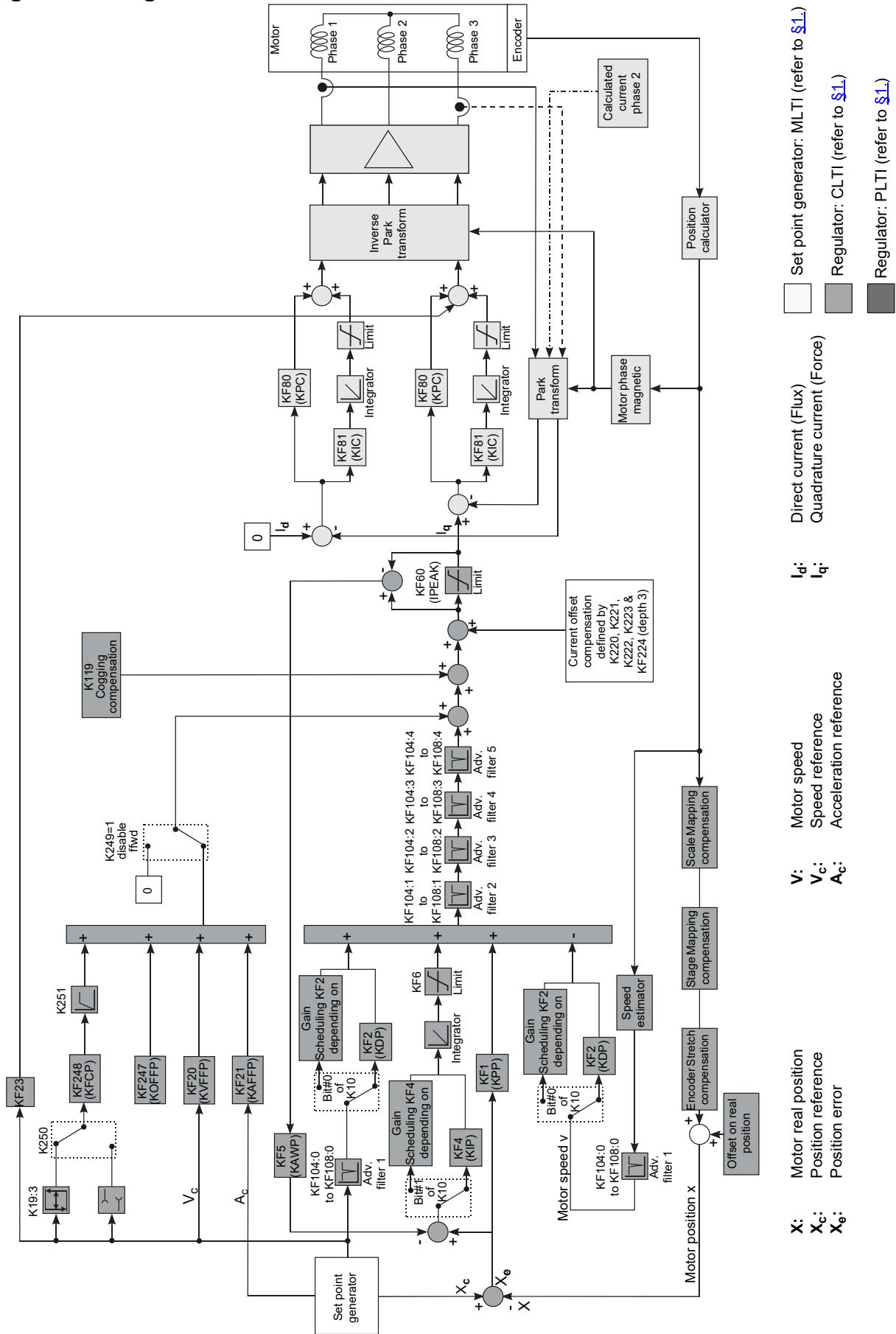
The figure below provides an example of a power sag when the Power Sag Detection is enabled. In this case, the sag is not long enough to trigger the **POWER SAG** error (M64 = 250).



8. Advanced functions (only for advanced users)

8.1 Regulators in details - advanced tuning

8.1.1 Regulators diagram



8.1.2 Regulators parameters

The position feedback is controlled by a **state regulator**, which can be approximated to a proportional-integral-derivative (PID) regulator. The **position state regulator's** parameters are:

8.1.2.1 Position regulator's gains and limits

K	Alias	Name	Comment	Units
KF1	KPP	Proportional gain	Gives the proportional gain of the position loop	-
KF2	KDP	Speed feedback gain	Gives the speed feedback gain of the position loop	-
KF4	KIP	Integrator gain	Gives the integrator gain of the position loop	-
KF5	KAWP	Anti-windup gain	Gives the anti-windup gain of the position loop	-
KF6	-	Integrator limit	Gives the integrator limit of the position loop	-
KF23	-	Back EMF compensation	Gives the Back EMF compensation	-
KF60	IPEAK	Maximum current limit	Gives the theoretical software current limit (regulator output)	[ci]

Remark: There is no back EMF compensation on the AccurET VHP controllers, then KF23 does not exist.

Parameters **KF1**, **KF2** and **KF4** define the PID of the position regulator. The parameter KF1 is proportional to the motor position error x_e . It is used to make the position regulator more reactive. The integral gain (parameter KF4) will reduce the oscillations and will suppress a permanent error on the position. The parameter KF2 is proportional to the speed (and not the speed error), thus it acts as a pseudo-derivative gain. The value of KF1, KF2 and KF4 can be monitored by monitorings MF1, MF2 and MF34 respectively (refer to [§8.1.3](#)).

The parameter **KF5** (anti-windup) works with the parameter KF60 (the current reference limitation).

The parameter **KF60** is automatically set to protect the motor against a too high current reference (I_r) coming from the state regulator. For an optimal operation, the parameter KF60 should never be reached, or should limit I_r only for a very short time. If the parameter KF60 is often reached and for a long time, it means that the motor is under-sized compared to its load. In that case, the parameter KF5 should be used to compensate the state regulator saturation. When the parameter KF60 is active, the low motor's current needs a too long time to reach a far position. During that time, the integrator reaches a high value. When the position is reached, I_r should invert its direction, but it is impossible because the integrator is full. To avoid the problem, the parameter KF5 will subtract a value from the integrator's input (parameter KF4), but only when the parameter KF60 is active (state regulator saturated).

The parameter **KF6** is the integrator's maximum and minimum values and the parameter **KF23** allows the compensation of the back EMF. The formula to calculate the value of KF23 is as follows (with M91 given in volt and not in increment):

$$KF23.<\text{axis}> = \frac{Ku \times \sqrt{\frac{2}{3}}}{M91.<\text{axis}>}$$

8.1.2.2 Gain Scheduling

In some applications, it is sometimes useful to modify the regulator's gains according to the state of the system. There are 3 types of Gain Scheduling available in AccurET controllers:

- Modification of the speed feedback gain KF2 depending on the real current
- Modification of the integrator gain KF4 depending on the motor speed
- The advanced Gain Scheduling for a flexible modification of PID gains in function of any register.

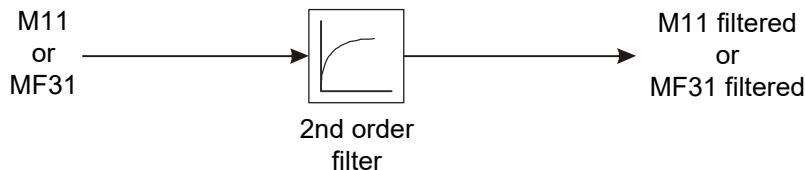
The Gain Scheduling is activated by the parameter **K10**.

K	Name	Comment
K10	Gain Scheduling activation	Gain Scheduling activation. Bit#0 for KF2 Gain Scheduling, bit#1 for KF4 Gain Scheduling, bit#2 for advanced Gain Scheduling. Now by default, if K10 = 0, 1, 2 or 3 after the reboot of the controller, it is not possible to use the advanced Gain Scheduling. If a user wants to use the advanced Gain Scheduling, he must save with K10=4.

A second order filter can be applied on the Gain Scheduling. This filter is set by the parameter **KF12**.

K	Name	Comment
KF12	Filter for Gain Scheduling	Gives the filter for the Gain Scheduling. Depth 0: filter on current level (MF31) for KF2 Gain Scheduling, depth 1: filter on speed level (M11) for KF4 Gain Scheduling

The filter design is as follows:



with

$$Ir = \frac{20000}{2\pi} \times \ln\left(\frac{2^{32}}{KF12}\right)$$

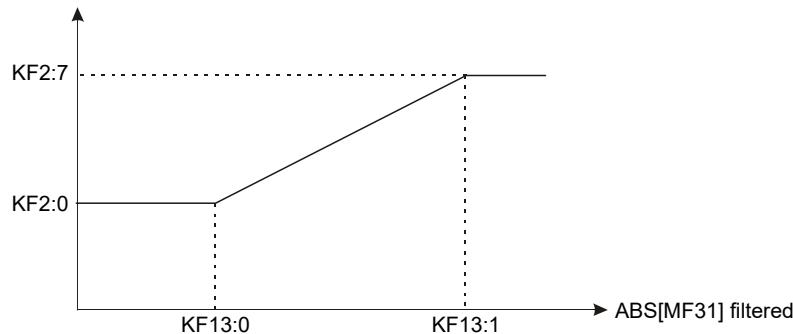
- **Gain Scheduling KF2**

Due to the non-linear behavior of the current loop at high-frequency related to the limited bus voltage, the stability of the system can be improved by increasing the speed feedback gain for high currents.

The Gain Scheduling KF2 is modified according to the value of the parameter **KF13**:

K	Name	Comment
KF13	Current level for KF2 Gain Scheduling	Gives the current level (MF31) from which KF2 Gain Scheduling is modified

The value of KF2 can be monitored by monitoring MF2 (refer to [§8.1.3](#)).



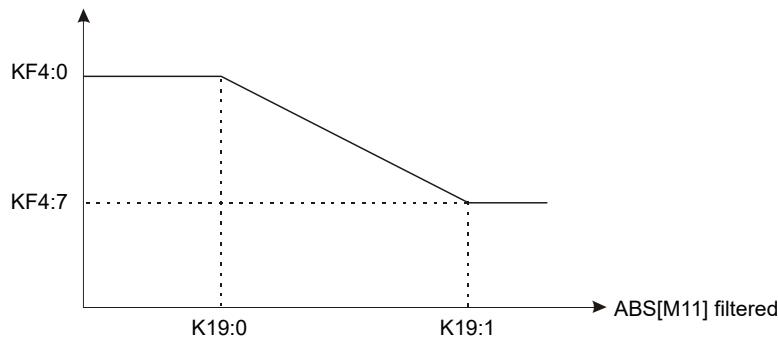
- **Gain Scheduling KF4**

At very low speed, the mechanical friction has a non-linear behavior at low-frequency that can be compensated by increasing the integrator gain. In this way, the settling time can be reduced without compromising the stability of the system during motion.

The Gain Scheduling KF4 is modified according to the value of the parameter **K19**:

K	Name	Comment
K19	Gain Scheduling KF4, speed level for I2t motor,...	Depth 0 and 1 for Gain Scheduling KF4

The value of KF4 can be monitored by monitoring MF34 (refer to [§8.1.3](#)).



- **Advanced Gain Scheduling**

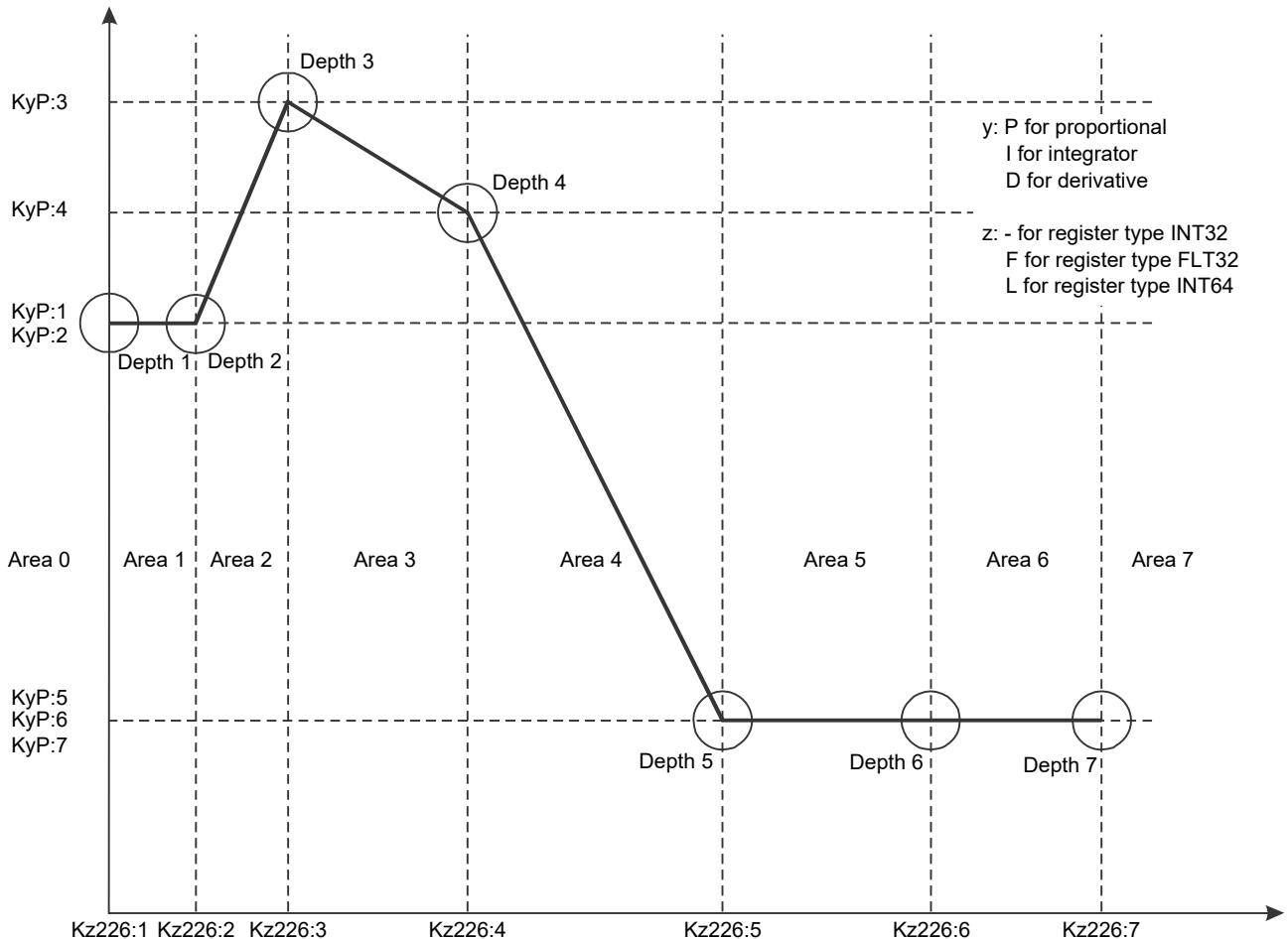
The points of the advanced Gain Scheduling are defined by the parameter **K226**.

K	Name	Comment
K226 KF226 KL226	Advance gain scheduling points	defines the points (maximum 7) of the advanced Gain Scheduling with depths 1 to 7.

The points (of the abscissa) can be positive or negative and must be defined from the lowest value in depth 1 to the highest in depth 7. This parameter is in increment without ISO conversion. If not all area are used, it is necessary to set the others with the same value as the last used area.

Remark: In gantry level 1 or 2, parameters K226, KF226 and KL226 could be different between the two axes.

In gantry level 2, it is not possible to change the value of parameters K226, KF226 and KL226.



The parameter Kz226, representing the register type of the abscissa, is defined by:

- **depth 4 of parameter K220 (refer also to §8.2.1) to select the type of the register**
 Kz226 is a 32-bit integer register if K220=1, 2, 3 or 13
 Kz226 is a 32-bit float register if K220=33, 34, 35 or 45
 Kz226 is a 64-bit integer register if K220=65, 66, 67 or 77
- **depth 4 of parameter K221 (refer also to §8.2.1) to select the index of the register**
- **depth 4 of parameter K222 (refer also to §8.2.1) to select the depth of the register**
 (for X, XF and XL, only depth 1 is available and for M, MF and ML, no need to set a depth)
- **depth 4 of parameter K223 (refer also to §8.2.1) to select an absolute value or not of the register**
 K223:4.<axis>=0, the register is taken as an absolute value (ABS)
 K223:4.<axis>!0, the register is taken as a signed format

Remark: If the setting of the advanced Gain Scheduling (Kz226) is incorrect (depths 1 to 7 are not in ascending order), the **ERR GAIN SCHEDUL** error (M64=37) will occur.
 It is possible to change parameters K220, K221 and K222 during the movement.
 In gantry level 1 or 2, parameters K220, K221 and K222 must be the same between the two axes.
 In gantry level 2, it is possible to change the value of parameters K220, K221 and K222.

With the 7 depths, 8 different areas (see the above drawing) can be defined:

Area 0: when the abscissa < Kz226:1 (in area 0, the value of the gains are the ones of depth 1 (KyP:1))
 Area 1: when Kz226:1 ≤ abscissa < Kz226:2
 Area 2: when Kz226:2 ≤ abscissa < Kz226:3
 Area 3: when Kz226:3 ≤ abscissa < Kz226:4
 Area 4: when Kz226:4 ≤ abscissa < Kz226:5
 Area 5: when Kz226:5 ≤ abscissa < Kz226:6
 Area 6: when Kz226:6 ≤ abscissa < Kz226:7
 Area 7: when Kz226:7 ≤ abscissa (in area 7, the value of the gains are the ones of depth 7 (KyP:7))

Remark: The area management is done on the MLTI side and the gains calculation is done in the PLTI.
 Inside an area, a linear interpolation is done to calculate the gains.

The 'Advanced Gain Scheduling' function is activated when bit#2 of parameter K10 is set. If bit#2 is set, bit#0 (used for Gain Scheduling KF2) and bit#1 (used for Gain Scheduling KF4) are set to 0 without generating any error. When this functionality is not activated, depth 0 of the 3 gains (KPP, KDP and KIP) is used.

Remark: In gantry level 1 or 2, parameter K10, KF1, KF2 and KF4 could be different between the two axes.
 In gantry level 2, it is not possible to change the value of parameter K10, KF1, KF2 and KF4.
 It is possible to change the value of parameter K10 during the motion.

8.1.2.3 Feedforward

The parameter **K249** allows the user to disable all feedforward.

K	Name	Value	Comment
K249	Disable all feedforward	0 1	All feedforward are enabled (default value) all feedforward are disabled

- **Friction feedforward**

The parameter **K250** allows the user to choose the configuration for the friction feedforward.

K	Name	Value	Comment
K250	Configuration of dry friction feedforward	0 1	Dry friction with hysteresis (default configuration) Dry friction without hysteresis (discontinuity at speed 0)

The parameter **K251** allows the user to filter the feedforward response to smooth the step at speed 0 or at the speed given by K19:3. The depth 3 of the parameter **K19** (K19:3) is the threshold speed used for the relay functionality from which the friction compensation is added depending on the direction. The parameter **KF248** is the feedforward gain used to compensate the dry friction.

K	Name	Comment	Unit
K251	Filter for dry friction feedforward	The authorized values are 'power of 2' (0, 1, 2, 4, 8, 16, 32 or 64 PLTI). For all the other values, the filter is deactivated.	-
K19	Speed level for I2t motor, Gain Scheduling KF4,...	Friction feedforward given by depth 3	[dsi]
KF248	Current friction feedforward gain	Gives the force/torque friction feedforward gain	[cur]

Here is the effect when K250=0 or 1

Description when parameter K250.<axis>=0:

If the speed reference (M109.<axis>) is greater than K19:3.<axis> => KF248.<axis>*1.0 is added.
 If the speed reference (M109.<axis>) is equal to 0 after a positive movement => KF248.<axis>*1.0 is added.
 If the speed reference (M109.<axis>) is lower than -K19:3.<axis> => KF248.<axis>*-1.0 is added.
 If the speed reference (M109.<axis>) is equal to 0 after a negative movement => KF248.<axis>*-1.0 is added.
 After PWR.<axis>=1, AUT.<axis> or INI.<axis> => KF248.<axis>*0.0 is added.

Description when parameter K250.<axis>=1:

If the speed reference (M109.<axis>) is greater than 0 => KF248.<axis>*1.0 is added.
 If the speed reference (M109.<axis>) is lower than 0 => KF248.<axis>*-1.0 is added.
 If the speed reference (M109.<axis>) is equal to 0 => KF248.<axis>*0.0 is added.

Remark: A 'Friction compensation' tool is present in our sizing software. Refer to the '**ComET Communication Manual**' for more information.

- **Current offset feedforward**

K	Alias	Name	Comment
KF247	KOFP	Current offset feedforward	Gives the current offset feedforward gain

- **Speed and acceleration feedforward**

K	Alias	Name	Comment
KF20	KVFFP	Speed feedforward	Gives the speed feedforward gain of the position loop
KF21	KAFFP	Acceleration feedforward	Gives the acceleration feedforward gain of the position loop

During a movement, a permanent error may appear between the position reference and the motor's real position. This drag can be due to mechanical friction. It is possible to compensate it with feedforward parameters. These parameters will increase the speed and acceleration command inputs to the state regulator. The **speed feedforward**, given by the parameter **KF20**, will compensate the drag's constant part and the **acceleration feedforward**, given by the parameter **KF21**, will compensate the undershoot and overshoot remaining after the speed feedforward compensation.

Remark: Parameters KF20 and KF21 can be used with all reference modes (parameter K61).

The parameter **K24** allows the user to add a filter (in the PLTI) to linearize the acceleration and to avoid doing a jump in the feedforward acceleration given by the parameter KF21. The second effect of this filter is to shift the movement of 1 MLTI which means the movement starts 1 MLTI later.

K	Name	Comment
K24	Filter for acceleration feedforward	Filter of the theoretical acceleration for the acceleration feedforward (0: disable, 1: active)

Remark: This parameter can be change only when the axis is not moving.

8.1.2.4 Regulators filters

Parameters **KF104**, **KF105**, **KF106**, **KF107** and **KF108** are used to define advanced filters (second order filter). There are five different depths for those parameters:

- depth 0 is used to filter the speed feedback (or pseudo-derivative gain) to the state regulator (parameter KF2). Monitorings M111 and M11 allow the user to know the real speed before and after the advanced filter respectively.
- depth 0 is also used to filter the speed reference to the state regulator (parameter KF2). Monitorings M109 and M10 allow the user to know the real speed before and after the advanced filter respectively.
- depth 1 to 4 are used to filter the output of the position's regulator (current reference).

K	Name	Comment
KF104	Advanced filter coefficient	Advanced filter coefficient for term yk_1
KF105	Advanced filter coefficient	Advanced filter coefficient for term yk_2
KF106	Advanced filter coefficient	Advanced filter coefficient for term xk
KF107	Advanced filter coefficient	Advanced filter coefficient for term xk_1
KF108	Advanced filter coefficient	Advanced filter coefficient for term xk_2

Remark: The default values of the those parameters make the advanced filters inactive. To make it active use the 'Filter design' tool of the ComET software.

The **SAF** command (Set Advanced Filters) allows the user to set in once a complete advanced filter.

Command	<P>	Comment
SAF.<axis>=<P1>,<PF2>,<PF3>,<PF4>,<PF5>,<PF6>	<P1> <PF2> <PF3> <PF4> <PF5> <PF6>	Depth of the advanced filters Value of KF104 Value of KF105 Value of KF106 Value of KF107 Value of KF108

Remark: The **BAD CMD PARAM** error (M64=70) is generated if the command does not have the right number of parameters and if the types of parameters are wrong.

8.1.2.5 Current regulator's gains

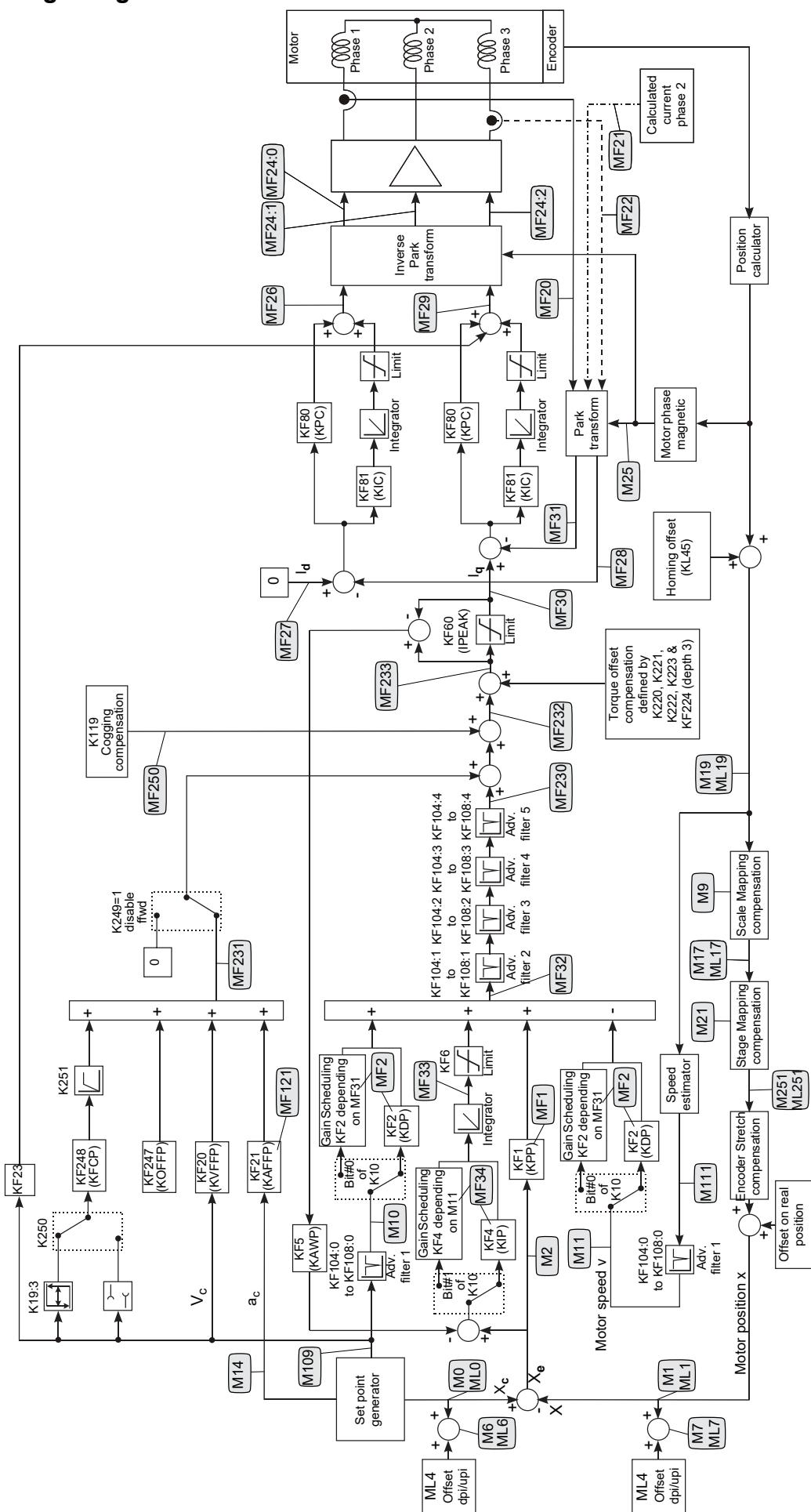
The **current reference generator** (with the motor's look-up table) calculates the input current into each motor phase from the current reference Ir delivered by the position loop and the real motor position. The current loop itself makes sure that this reference is reached.

K	Name	Comment
KF80	Current loop proportional gain	Proportional gain of the current loop.
KF81	Current loop integrator gain	Integrator gain of the current loop.

To do that, a classical PI regulator is used. The parameter **KF80** is proportional to the current reference Ir. It is used to make the current regulator more reactive. The integral gain given by the parameter **KF81** will reduce oscillations and will suppress a permanent error on the current.

Remark: With a direct drive, the quality of the current loop has to be very good, otherwise the noise due to the transistor commutation or the current bad measurement due to a ripple, will not be removed by the position loop. A special algorithm has been developed in the controller to reduce the noise.

8.1.3 Monitorings diagram



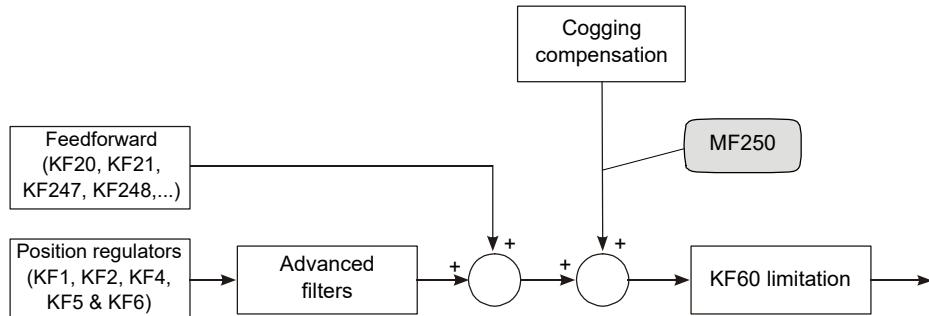
8.1.3.1 Monitorings description

M	Alias	Name	Comment	Units
M0 ML0	-	Theoretical position	Theoretical position Xc. It does not take care about the SET command. The monitoring M0 corresponds to the LSL part of the monitoring ML0.	[dpi] [rdpi]
M1 ML1	-	Real position	Real position X. Takes the mapping corrections into account, but does not take care about SET command. The monitoring M1 corresponds to the LSL part of the monitoring ML1.	[dpi] [rdpi]
MF1		Position loop proportional gain value	Position loop proportional gain value KF1 modified by Gain Scheduling	-
M2	-	Position control error	Tracking position control error Xe. This is the difference between monitorings M0 and M1	[dpi] [rdpi]
MF2		Position loop speed feedback gain value	Position loop speed feedback gain value KF2 modified by Gain Scheduling	-
M6 ML6	-	Theoretical position in user scale	Theoretical position Xc in user scale. Takes SET command into account. The monitoring M6 corresponds to the LSL part of the monitoring ML6.	[upi] [rupi]
M7 ML7	-	Real position in user scale	Real position X in user scale. Takes SET command and the mapping correction into account. The monitoring M7 corresponds to the LSL part of the monitoring ML7.	[upi] [rupi]
M9 ML9	-	Scale Mapping compensation value	Scale Mapping compensation value. The monitoring M9 corresponds to the LSL part of the monitoring ML9.	[dpi], [rdpi]
M10	-	Theoretical speed	Theoretical speed Vc after advanced filter	[dsi], [rdsi]
M11	-	Real speed	Real speed V after advanced filter at depth 0	[dsi], [rdsi]
M13 ML13	-	Secondary encoder position	Position given by the secondary encoder. The monitoring M13 corresponds to the LSL part of the monitoring ML13.	[dipi2], [rdpi2]
M14	-	Theoretical acceleration	Theoretical acceleration Ac	[dai], [rdai]
M15 ML15	-	Auxiliary encoder position	Position given by the auxiliary encoder. The monitoring M15 corresponds to the LSL part of the monitoring ML15.	[dpi], [rdpi]
M17 ML17	-	Real position after Scale Mapping	Real position after Scale Mapping, before Stage Mapping. The monitoring M17 corresponds to the LSL part of the monitoring ML17.	[dpi], [rdpi]
M19 ML19	-	Real position coming from the encoder	Real position coming from the encoder, before Scale Mapping (dpi). The monitoring M19 corresponds to the LSL part of the monitoring ML19.	[dpi], [rdpi]
MF20	RCUR1	Real current in phase 1	Real current in phase 1	[ci], [A]
M21	-	Stage Mapping compensation value	Stage Mapping compensation value	[dpi], [rdpi]
MF21	RCUR2	Real current in phase 2	Real current in phase 2	[ci], [A]
MF22	RCUR3	Real current in phase 3	Real current in phase 3	[ci], [A]
MF24	-	Phase PWM value	Phase PWM value. Depth0: phase 1, depth1: phase 2 and depth2: phase 3	Incr. [1]*
M25	-	Phase angle	Phase angle	Incr. [1]*
MF26	UD	Ud vector control part	Ud vector control part	Incr. [1]*
MF27	TID	Theoretical current Iq reference	Theoretical current Iq reference	[ci], [A]
MF28	RID	Real current Id measured	Real current Id measured	[ci], [A]
MF29	UQ	Uq vector control part	Uq vector control part	Incr. [1]*
MF30	TIQ	Theoretical current	Theoretical current Iq for current loop after KF60 limitation	[ci], [A]
MF31	RIQ	Real current	Real current F	[ci], [A]
MF32	-	Theoretical current	Theoretical current Ir before advanced filter	[ci], [A]
MF33	-	Integrator value	Position loop integrator output value	[ci], [A]
MF34	-	Integrator factor	Position loop integrator gain value KF4 modified by Gain Scheduling	-
M109	-	Theoretical speed	Theoretical speed before advanced filter	[dsi], [rdsi]
M111	-	Real speed	Real speed before advanced filter at depth 0	[dsi], [rdsi]
MF116	-	Current reference value	Current reference value defined by parameters K220, K221, K222, K223 & KF224	[ci], [A]
M117 ML117	-	Position reference value	Position reference value defined by parameters K220, K221, K222, K223 & KF224. The monitoring M117 corresponds to the LSL part of the monitoring ML117.	[dpi] [rdpi]
M118 ML118	-	Speed reference value	Position reference value defined by parameters K220, K221, K222, K223 & KF224. The monitoring M118 corresponds to the LSL part of the monitoring ML118.	[dsi] [rdsi]
M119	-	Position reference value after trajectory filter	Position reference value after trajectory filter.	[dpi] [rdpi]
M251 ML251	-	Real position after Stage Mapping	Real position after Stage Mapping, before Encoder Stretch. The monitoring M251 corresponds to the LSL part of the monitoring ML251.	[dpi] [rdpi]

[1]*: ISO relative voltage value are given in a range of [-1 to +1] or [-100 to 100%]. To get ISO value in [V], this value must be multiplied by Ubus/2 (M91 / 200).

8.1.4 Cogging compensation

The cogging compensation is done according to the measured position before Scale Mapping (ML19). This compensation is added to the current reference after the advanced filters.



The cogging compensation is done through a look-up table (LD:2.<axis>) of correction with a maximum of 8192 points which can be programmed and saved in the controller (refer to [§8.3.2](#) for more information about the look-up tables). This look-up table contains the correction value in current increment [ci] to be applied on the measured position of the axis.

The parameters **KL370**, **KL371** and **K372** are needed to define this function:

K	Name	Comment	Unit
KL370	Cogging compensation start	Gives the position where the cogging compensation starts from. it is defined by the point 0 of the look-up table	[dpi], [rdpi]
KL371	Active cogging compensation length	Gives the length where the cogging compensation is active	[dpi], [rdpi]
K372	Cogging compensation point number	Gives the number of points for the cogging compensation	-

To enable the correction, a homing has to be done and the parameter **K119** must be properly set.

K	Name	Value	Comment
K119	Cogging compensation	0 1 2	Cogging compensation disabled Compensation for linear axis enabled Compensation for rotary axis enabled

By default, there is no correction (K119=0). If K119 is equal to 1 or 2, the correction will be immediately activated. The setting of the parameters KL370, KL371 and K372 will be done only after the end of a movement.

Remark: When the real position is out of the correction zone (defined by KL370 and KL371), the applied correction corresponds to the last value of the table.

Refer to the '**ComET Communication Manual**' for more information about the 'Cogging Compensation' tool.

The monitoring **MF250** allows the user to monitor, on the 'Scope' tool of ComET, the value of the correction.

M	Name	Comment	Unit
MF250	Cogging compensation value	Gives the cogging compensation value	[ci], [A]

8.2 Advanced reference modes (K61≠1 or K63=1)

The parameters **K61** and **K63** selects the type of input reference given to the controller. When K61≠1 or K63=1, an advanced reference mode is selected.

K	Name	Value	Comment
K61	Position reference mode	1	Standard mode with set point generator movement profiles (refer to S7.6).
		3	Pseudo-speed reference defined by parameters K220, K221, K222, K223 and KF224 depth 0.
		4	Position reference defined by parameters K220, K221, K223 and KF224 depth 0.
		36	Like K61=4, but the actual position is kept as a reference when the controller is switched on.

K	Name	Value	Comment
K63	Current reference mode	0	Current reference is coming from position regulator
		1	Current reference is defined by registers K220, K221, K222, K223 and KF224 depth 0.

8.2.1 External reference modes (K61=3, 4 or 36 and K63=1)

The **external reference** mode includes all the cases where the reference is not given by the set point generator but by an external source which can be a K, M or X register.

In all cases the external source can be used as position (K61=4 or 36), speed (K61=3) or current (K63=1) reference. Parameters used with the external reference mode are given in the following table:

K	Name	Value	Comment
K220	External reference type	1	Reference type is a user's variable X
		2	Reference type is a parameter K
		3	Reference type is a monitoring M
		13	Reference type is a common parameter C
		33	Reference type is a user's variable XF
		34	Reference type is a parameter KF
		35	Reference type is a monitoring MF
		45	Reference type is a common parameter CF
		65	Reference type is a user's variable XL
		66	Reference type is a parameter KL
		67	Reference type is a monitoring ML
		77	Reference type is a common parameter CL
K221	External reference index	-	External reference index
K222	External reference depth	-	External reference depth
K223	External reference offset	-	External reference offset correction
KF224	External reference gain	-	External reference gain correction
K225	External mean filter	- 1 2 to 16	External mean filter on the position offset compensation. Valid only at the depth 1 and 6. No filtering Value * MLTI = length of the mean filter

Remark: The external position reference must be given in user position increments [upi], the external speed reference in user speed increments [usi] and the external current reference in current increments [ci]. In these modes, the SET command is disabled. When one of the external reference modes is used, the user must set **first** the parameters K220, K221, K222, K223, KF224 and K225 **before** the parameter K61 or K63.

Parameters **K220**, **K221** and **K222** determine which register will be used as an external reference.

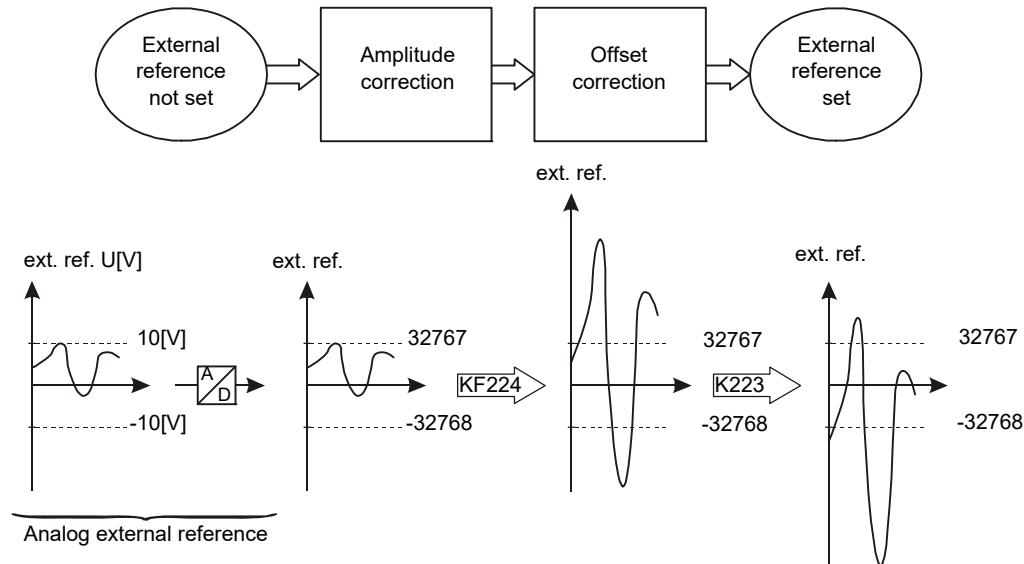
Parameters **K223** and **KF224** are used to configure the external reference. The external reference formula taken from the 'unset' external reference (value of the register given by parameters K220, K221 and K222) is:

$$\text{external reference set} = (\text{'unset' external reference} \times KF224) - K223$$

The external reference is multiplied by the value of the parameter KF224 to adjust the amplitude of the reference. After that, the value contained in the parameter K223 is subtracted to the previous result. This **offset** moves the origin according to the user's needs.

The parameter **K225** allows the user to linearly interpolate the values of the position offset compensation. If K225:1=16, the result corresponds to $16 \times 400 \mu\text{s} = 6.4 \text{ ms}$. The parameter K225 is valid only at the depth 1 and 6.

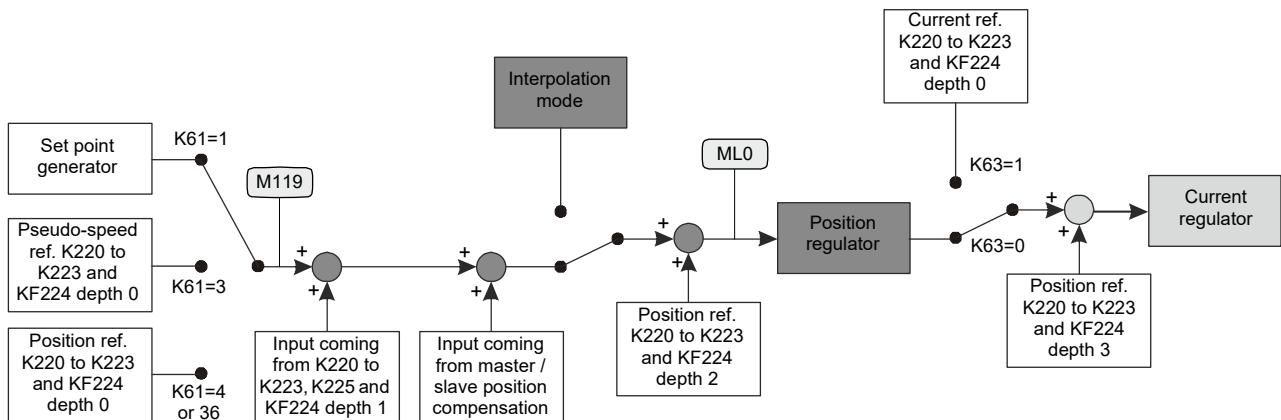
The following diagram shows the different steps. They are the same for all external reference modes but the limits can be different depending on the modes. In the following example (with K63=1), a signal sent to the analog input of the optional board has been chosen for the external reference.



Remark: Refer to [§22](#) for more information about the units of the external reference.

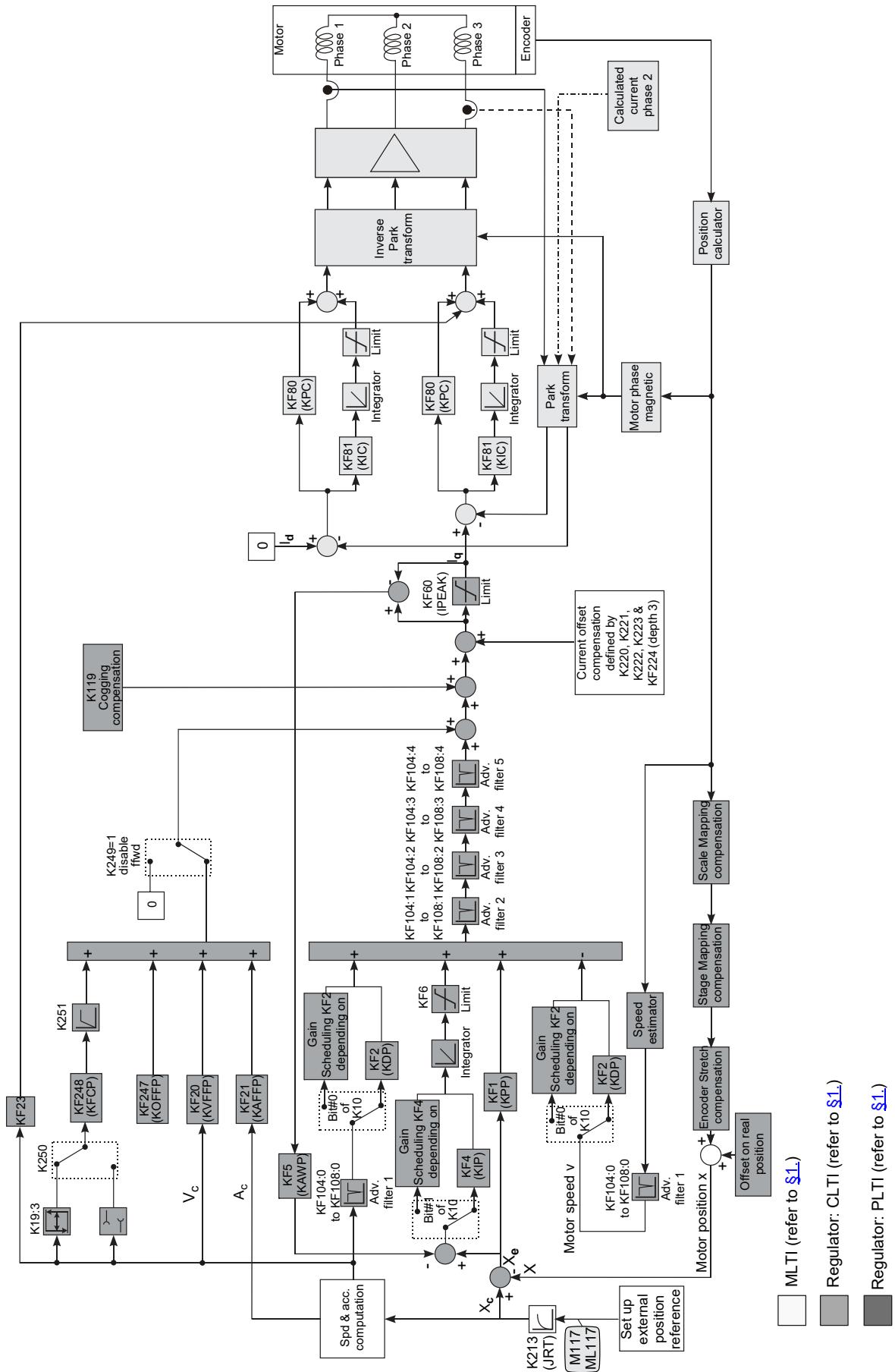
Whatever the value of K61 and the interpolated mode, it is possible to add a correction value to the position reference at three different places. To do so, the parameters K220 to K225 have different depths:

- Depth 0: dedicated to the external references modes described above
- Depth1: possibility for the user to add an offset on the position at the set point generator level. In that case it is possible to set a mean filter of the correction value with the parameter K225:1
- Depth2: possibility for the user to add an offset on the position after the interpolation mode input
- Depth3: possibility for the user to add an offset on the current after the position regulator
- Depth 5: master/slave position compensation (refer to [§8.2.2](#))
- Depth 6: compensation on real position (refer to [§8.2.3](#))



8.2.1.1 Position reference mode (K61=4 and 36)

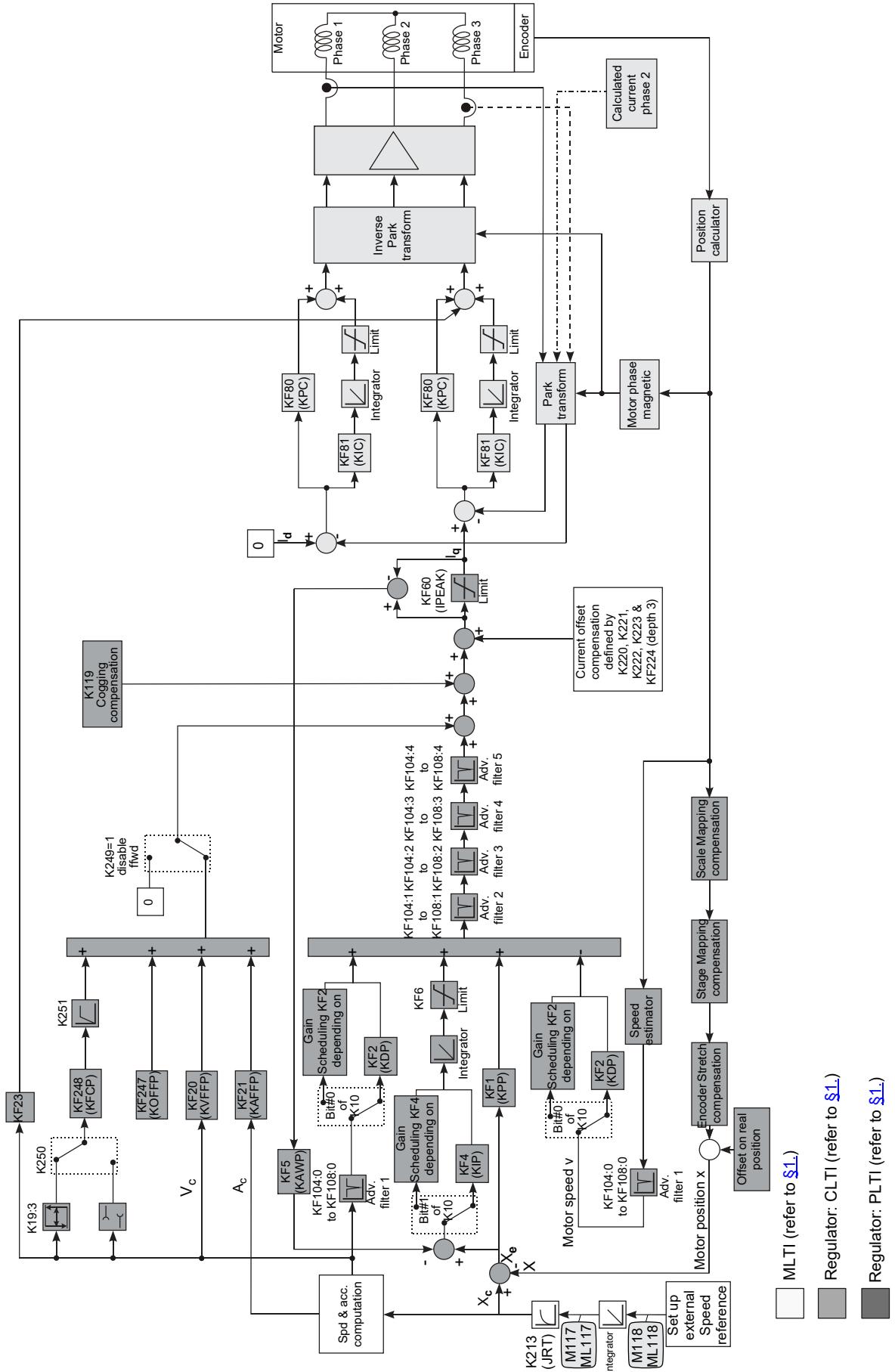
K61=36 is identical to K61=4, except that the actual position is kept as a reference when the controller is switched on. The graph below shows the regulation loop of the position reference mode:



Remark: The external position reference is given in [upi] and [rudi]. Refer to [§22](#), for more information.

8.2.1.2 Speed reference mode (K61=3)

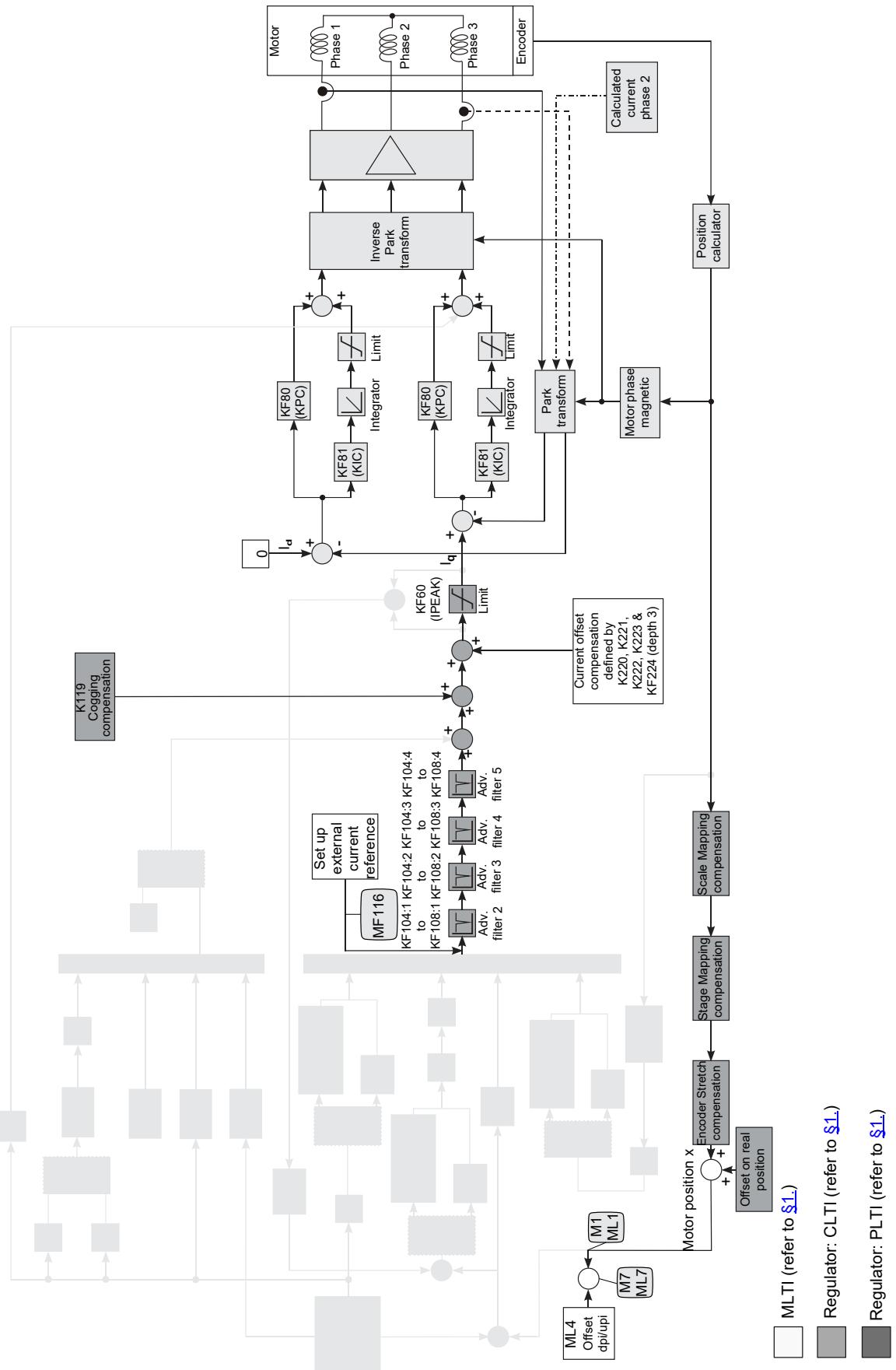
The external reference is used as a speed reference. An integrator calculates the position reference from this reference. The graph below shows the regulation loop of the speed reference mode:



Remark: The external speed reference is given in [us] and [rusi]. Refer to [§22](#) for more information.

8.2.1.3 Current reference mode (K63=1)

The external reference gives directly the motor reference current I_r without going through the position loop regulator. The limit given by the parameter KF60 is used and limits the current reference value after being set. The graph below shows the regulation loop of the current reference mode.



Remark: The external current reference is given in [ci] and [ci]. Refer to [S22](#) for more information.

8.2.2 Master/slave position compensation

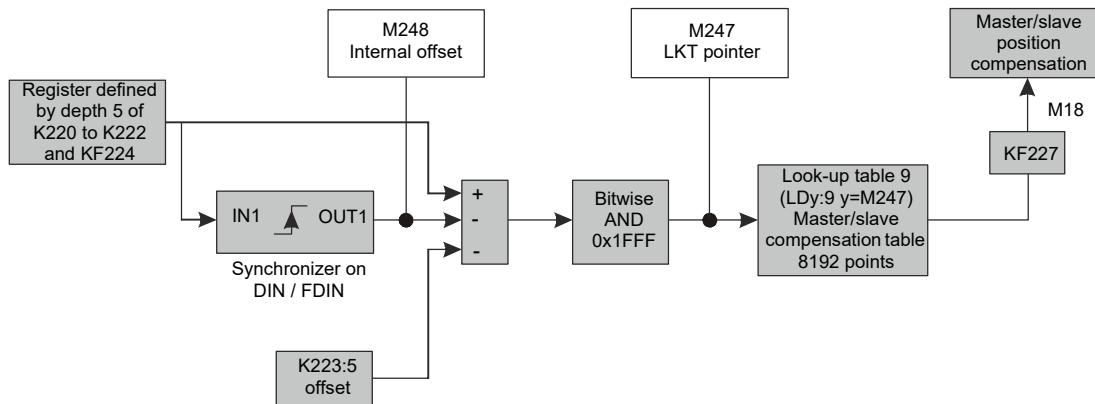
The master/slave position compensation feature allows the user to define any register as a pointer, to extract a value from a look-up table (at the MLTI rate) and add it to the position reference as 'position compensation'.

Remark: This feature is only available on AccurET Modular Rev2.
The position compensation must be lower than 32bits.

The master/slave position compensation, given by the monitoring **M18**, is stored in the look-up table (LDx:9 with 8192 points max.). The user can define any register to be used to compute a pointer on this look-up table defined by the monitoring **M247**. This register is defined through the depth 5 of K220 to K222 and KF224 parameters (same principle as the other features using the other depths described in [§8.2.1](#)). It is also possible to reset externally the pointer through DIN/FDIN.

The functionality is disabled when K220:5.<axis> and K221:5.<axis> point to register K0 (K220:5.<axis>=2 and K221:5.<axis>=0) and KF224:5.<axis>=0.0F.

When K220:5.<axis> and K221:5.<axis> point to another register than K0, simply by modifying KF224:5.<axis>, the user can enable the functionality (KF224:4.<axis>!=0.0F) or disable the functionality (KF224:5.<axis>=0.0F). When enabling or disabling this functionality, the correction is directly applied or removed (no smooth ramp). With those parameters, it is possible to determine the external reference for the pointer given by M247.



The following parameters are also needed to set the function:

K	Name	Comment
KF227:0	Master/slave compensation gain	Applies a gain (-100 to +100) to the value extracted from the LDx:9 look-up table.
K223:5	External reference offset	Subtracts a basic offset to the register value defined through depth 5 of parameters K220 to K222 and KF224.
KL306:0	DIN edge mask for starting master/slave compensation	Defines the mask of the digital inputs (rising & falling edge) to start again the scanning of the look-up table from the beginning (start again with pointer at 0): Bit#0-9: DIN.<1st axis> rising edge Bit#10-19: DIN.<2nd axis> rising edge Bit#32-41: DIN.<1st axis> falling edge Bit#42-51: DIN.<2nd axis> falling edge
KL306:1	FDIN edge mask for starting master/slave compensation	Defines the mask of the fast digital inputs (rising & falling edge) to start again the scanning of the look-up table from the beginning (start again with pointer at 0): Bit#0-31: FDIN rising edge Bit#32-63: FDIN falling edge

For the parameter KL306, it is possible to have more than one condition and to detect the rising and falling edge on the same DIN/FDIN or on a different one. The detection is done at the MLTI interrupt. The pulse must be present at least during two MLTI to be sure to detect the edge.

The following monitorings are also needed to monitor the function:

M	Name	Comment
M18	Master/slave position compensation	Master/slave position compensation added to the reference position [dpi]
M247	Master/slave compensation pointer	Show the pointer value determined by the external reference for the LKT pointer
M248	Master/slave compensation pointer offset	Show the pointer offset determined by the edge detection defined by KL306

M248.<axis> = (defined register x KF224:5). It is computed each time the occurrence of the external reset (DIN/FDIN) occurs.

M247.<axis> = ((defined register × KF224:5) – M248 – K223:5) & 0x1FFF

This feature is also supported in gantry mode (C245).

- **Gantry level 1**

In gantry level 1 (C245.<axis>=1), the feature is the same as in single axis mode.

The parameters K220:5 to K223:5 and KF224:5 must be at the same value on the master and on the slave (same principle as for the other depth of the registers). The parameters KF227 and KL306 do not need to be at the same value.

- **Gantry level 2**

In gantry level 2 (C245.<axis>=2), the master/slave compensation works in the same manner than the compensation coming from depth 1 to depth 3 of K220 to K223 and KF224.

The parameters K220:5 to K223:5 and KF224:5 shall be at the same value on the master and on the slave (same principle as for the other depth of the registers). For these parameters, the management mode is 'Master executes for himself and for the slave; Slave enters into error' when trying to affect them. The pointer defined by K220:5<master> to K223:5<master> and KF224:5<master> is defined for both axes (master and slave). The parameter KF227 shall be at the same value on the master and on the slave. For this parameter, the management is 'Master executes for himself and for the slave; Slave enters into error' when trying to affect them.

The parameter KL306 does not need to be at the same value on the master and on the slave. For this parameter, the management is 'Master executes for itself; Slave enters into error' when trying to affect them. LDx:9.<master> is for the master and for the slave. DIN/FDIN for the resynchronization could come from the master or the slave but must be defined in KL306.<master>.

8.2.3 Compensation on real position

The compensation is added after the Encoder Stretch compensation. K225:6 allows the user to determine if the compensation is updated every 400 µsec, 100 µsec or 50 µsec. On the other hand, a filter is applied to smooth the jump between two new values. The filter length is of 2 for variable updated every 100 µsec and of 8 for variable updated every 400 µsec.

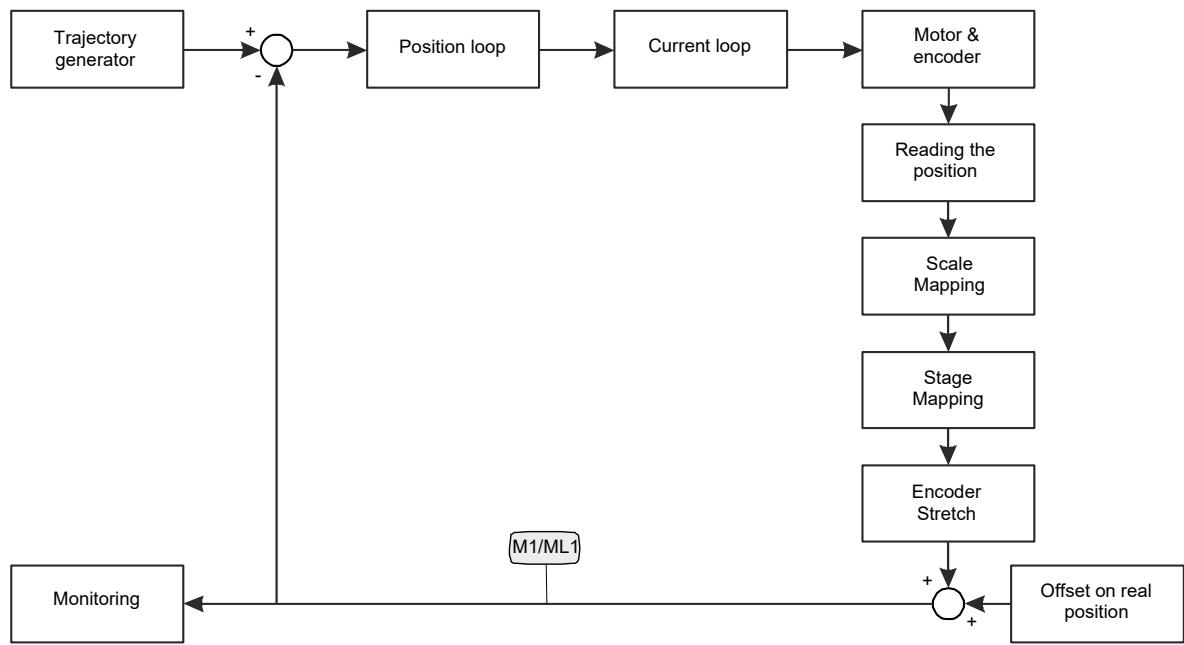
- When K225:6.<axis>=1, a filter of 8 PLTI is applied (for compensation updated at the MLTI).
- When K225:6.<axis>=2, a filter of 1 PLTI is applied (for compensation updated every PLTI).
- When K225:6.<axis>=3, a filter of 2 PLTI is applied (for compensation updated every 100 µsec).

For all other value of K225:6.<axis>, a filter of 8 PLTI is applied and a new value is calculated every 400 µsec.

The compensation value is added every 50 µsec in the PLTI side. The compensation value is calculated on 32 bits. The register where the compensation value must be read is defined with K220 to K223 and KF224 depth 6. In gantry level 2 (C245.<axis>=2), the compensation calculated on the master is also applied on the slave axis.

The monitoring **M22** allows the user to display the value of the added compensation.

M	Name	Comment	Unit
M22	Compensation value on real position	Gives the compensation value on the real position added to the measured position	[dpi]



The functionality is disabled when K220:6.<axis> and K221:6.<axis> point to K0 parameter (K220:6.<axis>=2 and K221:6.<axis>=0) and KF224:6.<axis>=0.0F.

When K220:6.<axis> and K221:6.<axis> point to another register than K0.<axis>, simply by modifying KF224:6.<axis>, the user can enable the functionality (KF224:6.<axis>!=0.0F) or disable the functionality (KF224:6.<axis>=0.0F). When enabling or disabling this functionality, the correction is directly applied or removed (no smooth ramp).

8.3 Advanced movements

Here are the advanced movements usable in standard reference mode (K61=1).

8.3.1 Movements types

Advanced movement types are:

- Look-up table movement (refer to [§8.3.2](#))
- Infinite rotary movement (refer to [§8.3.3](#))
- Movement predefined profile (refer to [§8.3.4](#))

Remark: S-curve and rotary S-curve movements have been explained in [§7.12](#), as basic movements.

8.3.1.1 Movements definition

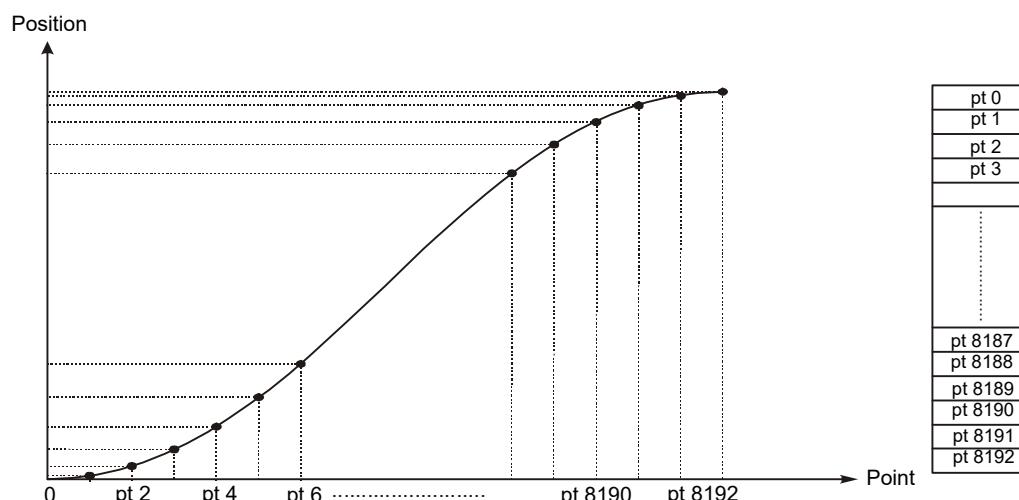
All parameters allowing the definition of advanced movements are mentioned below. Detailed explanations are given from paragraph [§8.3.3](#). Some of the following parameters are also available via alias.

K	Alias	Name	Value	Comment	Units
K202	MMD	Movement type	1 3 10 17 19 24 26	Linear S-curve (jerk time) movement Predefined linear movement according to K230 value and executes the trajectory in a time given by K229 Look-up table linear movement Rotary S-curve (jerk time) movement Predefined rotary movement according to K230 value and executes the trajectory in a time given by K229 Infinite rotary movement Look-up table rotary movement	-
K201	MMC	Concatenated movements selection	0 1 2 3	Concatenated movements disabled Concatenated movements for MMD=1, 17 & 24 Continuous back and forth movements enabled for MMD = 3, 10, 19 and 26 movement (can be stopped by MVE, BRK, HLT, HLB and HLO commands) One back and forth movements enabled for MMD = 3, 10, 19 and 26 (can be stopped by BRK, HLT, HLB and HLO commands)	-
KF205	CAM	Cam value	-	Cam value (in percent). Stretches the time scale	-
KL206	BRKDEC	Quick stop dec. pre-programmed	-	Brake deceleration (depth 0: with BRK and HLB command; depth1: brake ramp with the stage protection)	[uai] [ruai]

Remark: The **MMD** (Movement MoDe), **MMC** (Movement Mode Concatenated), **CAM** and **BRKDEC** (BRaKe DECeleration) are alias of the above-mentioned parameters and use the same syntax (for example: MMD.<axis>=<P1> and CAM.<axis>=<P1>).

8.3.2 Look-up table movements

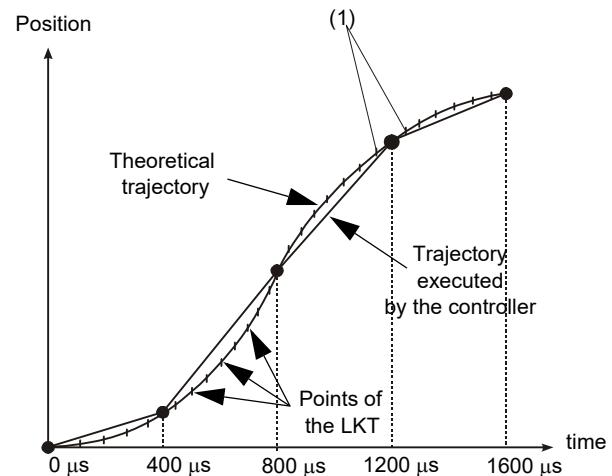
The **look-up table movements** (set with the parameter **K202**) are movements whose trajectories are freely set by the user. It is possible to save in the controller up to 10 different trajectories (in the depths). The **look-up table movement's trajectory** is kept in a table of **8192 points** memorized in a controller, so its name.



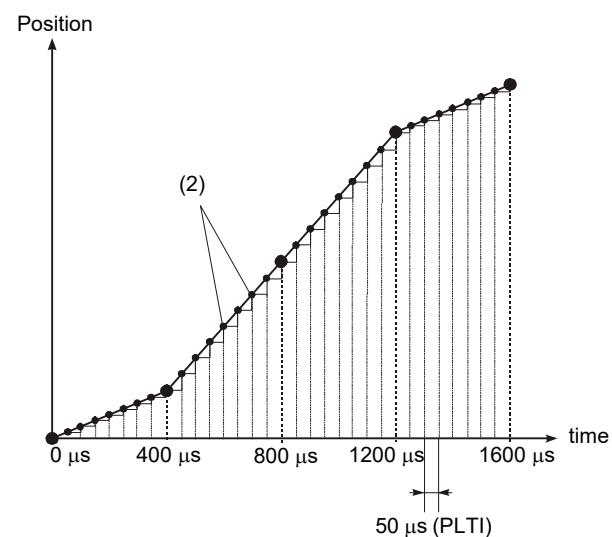
For specific applications, the user himself may create the movement trajectory. It can be a complex movement, with back and forth movements, where the controller is used as an electronic cam.

When a look-up table movement is required, only select the table with the requested movement and the total time of the movement $t_{movement}$ with LTN and LTI commands respectively. Then the MVE command selects the trajectory final point position x_{final} and starts the movement. If a long movement is requested during a very short lapse of time, speed and acceleration may attain very high values, that can even exceed the capacity of the system.

Each MLTI (Manager Loop Time Interrupt at 400 µs), the controller takes a point of the look-up table. The point depends on the value of the execution time (LTI) of the look-up table. If this value should fall between two points of the table, it will be linearly interpolated between the two adjacent points of the table (1).



Then the points read by the controller in the table are interpolated a second time (2) every PLTI (Position Loop Time Interrupt) at 50 µs. The trajectory is thus made of segments of a MLTI.



8.3.2.1 Start and end at the same point, max. stroke

It is possible with the look-up tables to begin and finish a movement at the same point (when K207=1). Refer to the drawing below.

K	Alias	Name	Value	Comment	Units
K207	-	LKT mode selection	0 1	LKT movement running to target defined by MVE command LKT movement with same starting and end point. Parameter KL208 defines the amplitude of the movement	-

In that case, the 'position to reach' (MVE.<axis>=<PL1>) is not used any more, as the end is the same as the start position (LD0:x=LD8191:x) and it **only** starts the LKT movement.

The **movement's maximum stroke** is defined by the parameter **KL208**. This parameter defines the furthest point (with respect to the start/end point) reached during the LKT movement.

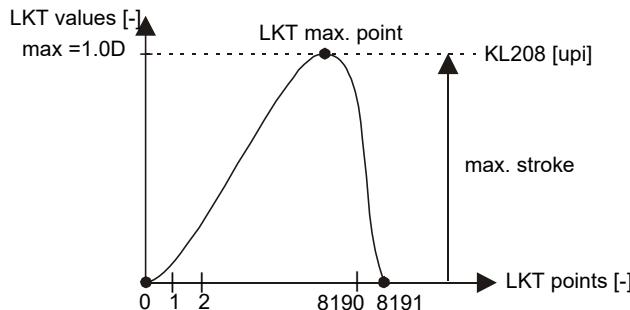
K	Alias	Name	Comment	Units
KL208	-	Max. stroke for LKT	Maximum stroke for LKT movement with K207=1 and for MMD=10 & 26	[upi]

8.3.2.2 Look-up table points definition

There are 8192 points in the LKT (0 to 8191 stored in the register LD) which must be not signed. The LKT points maximum possible value is 1.0D. Ideally, the point corresponding to the movement's maximum stroke should correspond to this value to have the highest possible resolution with the LKT movement. Position during the movement is given by the formula:

$$\text{Position[upi]} = \text{LKTvalues} \cdot \text{KL208[upi]}$$

The LKT maximum value for the maximum stroke will use the whole LKT resolution:



8.3.2.3 LTN and LTI commands

To realize a look-up table movement, the movement must first be selected with the MMD command. Then the requested table is selected with the **LTN** command (Look-up Table Number) and the total movement duration $t_{movement}$ with the **LTI** command (Look-up table TIme). Finally the movement is executed with the MVE command which also determines the final position x_{final} to reach.

K	Alias	Name	<P1>	Comment	Units
K203	LTN	LKT number movement	0-9	Number of the look-up table movement selected.	-
K204	LTI	LKT time	-	Time to execute a look-up table movement. Time = K204 x 400μs	[MLTI]

Remark: LTI and LTN alias use the same syntax than the corresponding parameter K. Refer to [§2](#) for more information about the syntax and the possible operators.

For technical reasons, the LTI value is limited to 625000. This means that it is not possible to realize look-up table movements of more than $400\mu s \cdot 625000 = 250$ s.

It is also possible to use the **CAM** command (refer to [§8.3.7](#)) to increase the duration of look-up table movements, but it is not possible to increase the time over 250s whatever the controller.

Caution: The reader must be aware that the shorter time $t_{movement}$ is, the bigger speed and acceleration become.

Once the LKT are programmed into the controller, you can use them as explained below:

```

LTN.0=<table number>; // Number of the look-up table to use.
LTI.0=<time>; // Execution time of the LKT in [MLTI].
K207.0=<LKT mode>; // 0 for a normal LKT and 1 for a LKT with same starting and end point.
KL208.0=<max. stroke>; // Maximum stroke for LKT movement when K207=1.
MVE.0=<start move>; // Starts the LKT movement (not taken into account when K207=0).
or
STA/STI.0=<P1, P2>; // Start a movement with <P1> is the value in a depth to start a movement and
// <P2> is a bits mask.
    
```

Remark: Refer to [§8.3.5](#) for more information about STA and STI command.

Example:

The following example shows how to realize a look-up table movement with a total movement of 200000 increments in 6s.

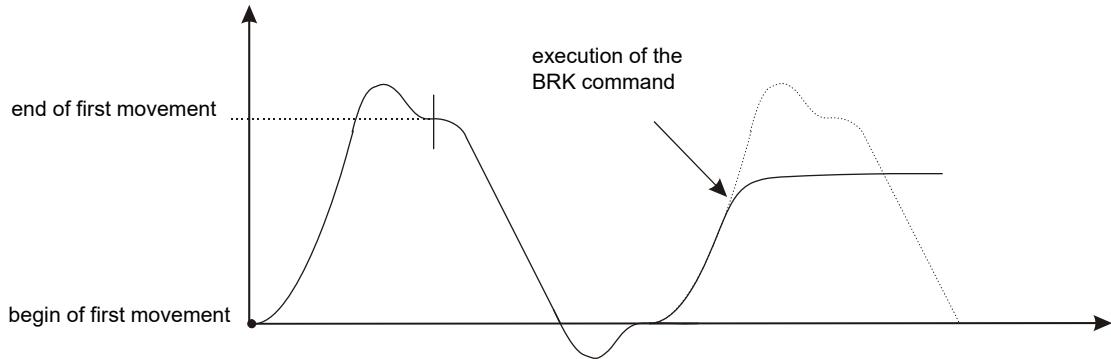
```

MMD.1=10 ;Select the look-up table movement.
LTN.1=3 ;Select the look-up table movement (LKT 3 is user-defined).
LTI.1=15000 ;The movement duration is: 15000 x 400μs = 6s.
MVE.1=200000L ;The motor moves by following the trajectory (defined in LKT 3) to the position
                  200000 and reaches this position 6s after leaving.
    
```

8.3.2.4 Concatenated look-up table movement

The look-up table movements may be concatenated with MMC=2 or 3.

Example:



Remark: The example above is also valid for MMD=10 (linear look-up table movement). To stop a concatenated look-up table movement, use a BRK, HLT, HLO or HLB command.

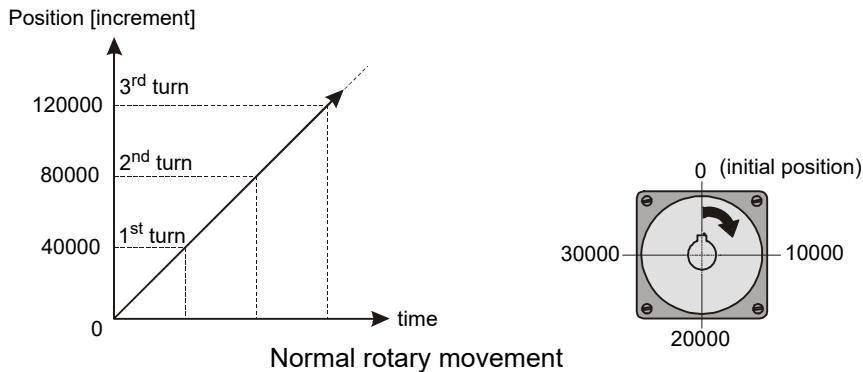
The final position before the return may not be reached if the movement is done in a short execution time (LTI). If it is absolutely required to reach the final position before the return, the user must carefully set the LKT, taking into account the MLTI duration, the LTI and the size of the LKT table.

8.3.3 Infinite rotary movement

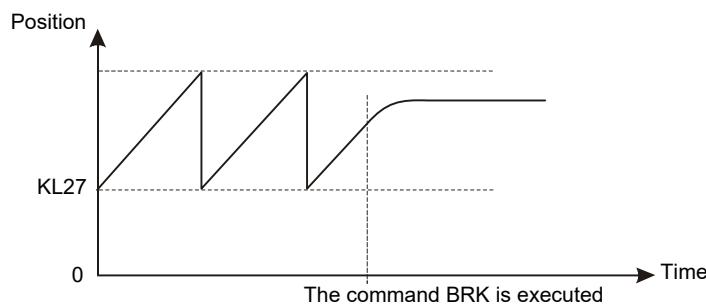
A rotary motor can make an infinite rotation in one or the other direction if there is no mechanical end stop limiting its stroke. This type of movement is selected with the value 24 of the MMD command or the parameter K202 (**MMD.<axis>=24** or **K202.<axis>=24**). To avoid the value of the motor's real position to become too important and exceed the limit of the controller's position counter, it must be reset at regular intervals with the parameter **KL27**. The motor turns indefinitely in the way defined by the parameter K209 (refer to [§7.12.4.1](#)). Its position is always included between 0 and the value of the parameter KL27.

8.3.3.1 Functioning principle

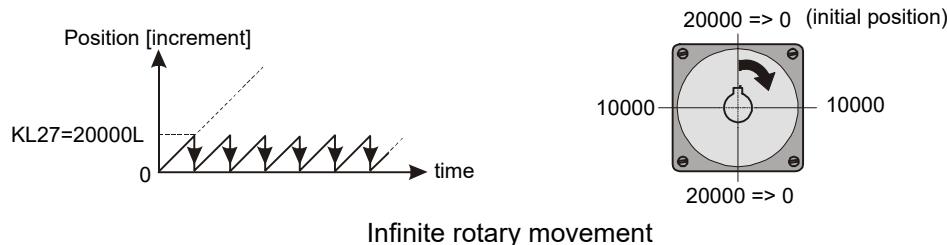
Supposing that a rotary motor revolves at constant speed, the measurement of the position versus time is:



The measurement of the position increases continuously and after a lapse of time, the value will be so big that the controller will not be able to handle it. To avoid it, as soon as the motor reaches the position programmed by the parameter KL27, the position is brought back to 0. Thus, the motor position is never bigger than the value of the parameter KL27. If the value 20000L is inserted in the parameter KL27 (half a turn), the measurement



of the motor position versus time will be as follows:



The movement undertaken before the measured distance has been reset is called a cycle. In the case above, a cycle corresponds to half a turn. The size of a cycle definition depends on the application.

Remark: In an infinite rotary movement, the acceleration can be modified only when the speed is constant (and whatever the value of MMC).

Example:

```
K209.1=0;           // Selects a positive movement
MMD.1=24;          // Selects an infinite rotary movement
KL27.1=20000L;     // Defines the value of the parameter KL27
ACC.1=300000L;     // Defines the acceleration of the movement
SPD.1=200000L;     // Defines the speed of the movement
MVE.1=1L;          // Starts an infinite movement whatever the value of the MVE.1 command
```

The motor starts revolving indefinitely in the positive direction with an acceleration of 300000 and a speed of 200000. It is a trapezoidal movement.

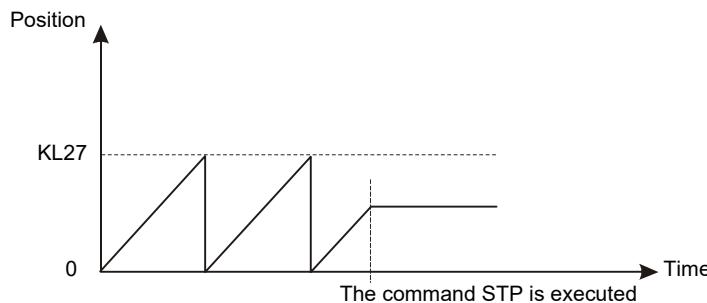
8.3.3.2 Stop of an infinite rotary movement

- Using the BRK command

The **BRK** command brakes the motor with the deceleration contained in the parameter KL206 and stops. The final position is not determined by the user. The use of the BRK command is generally recommended, as it is smoother for the machine.

- Use of the STP command

The **STP** command (**SToP**) behaves the same way as the BRK command but the deceleration is infinite. The braking is then very sudden. The use of the STP command is generally not recommended, as it is very sharp for the machine.

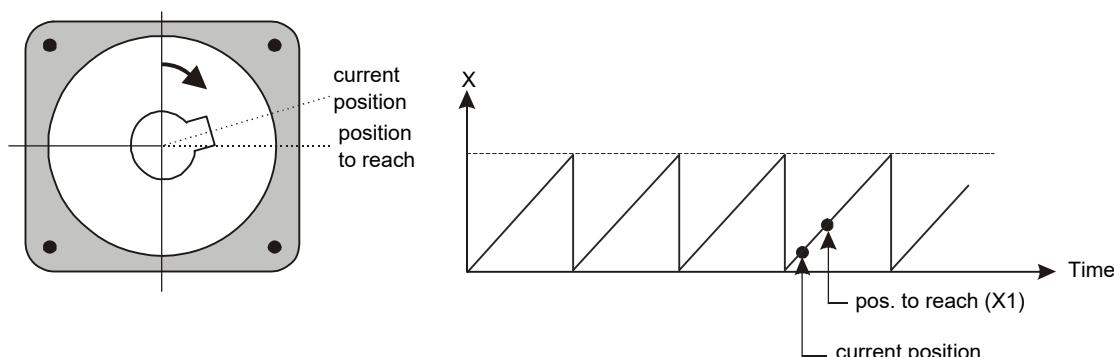


- **Use of the MVE command**

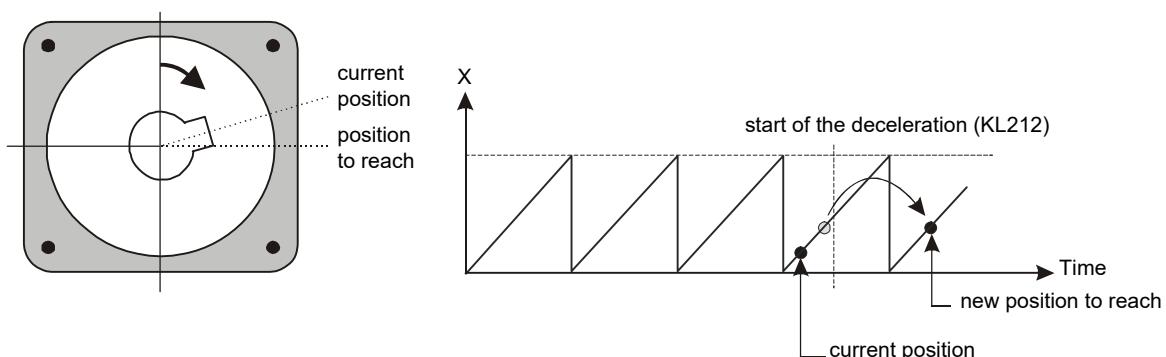
The MVE command stops the motor in a determined position with the acceleration defined by the parameter KL212.

The user can stop the motor in the position X1 by sending the command MVE.<axis>=X1;. If the distance between the current motor position and the position to reach (X1) is too small to stop the motor with the acceleration given in parameter KL212 (negative value = deceleration), the motor will make one more turn to stop correctly.

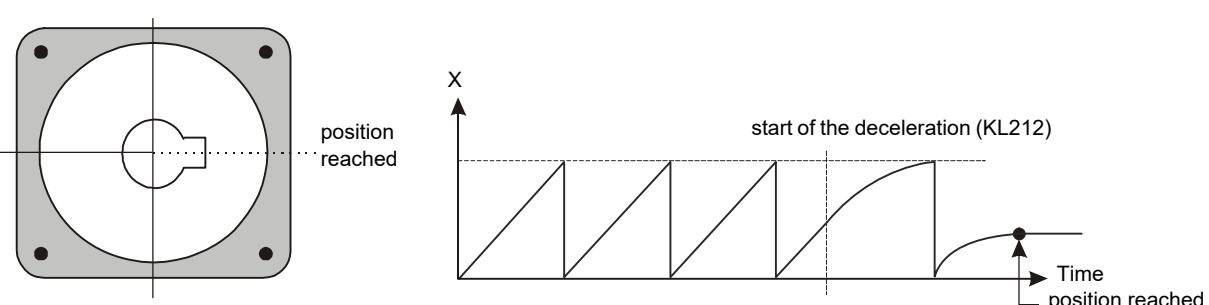
Step 1: MVE.<axis>=X1;



Step 2



Step 3



8.3.4 Movement with predefined profile

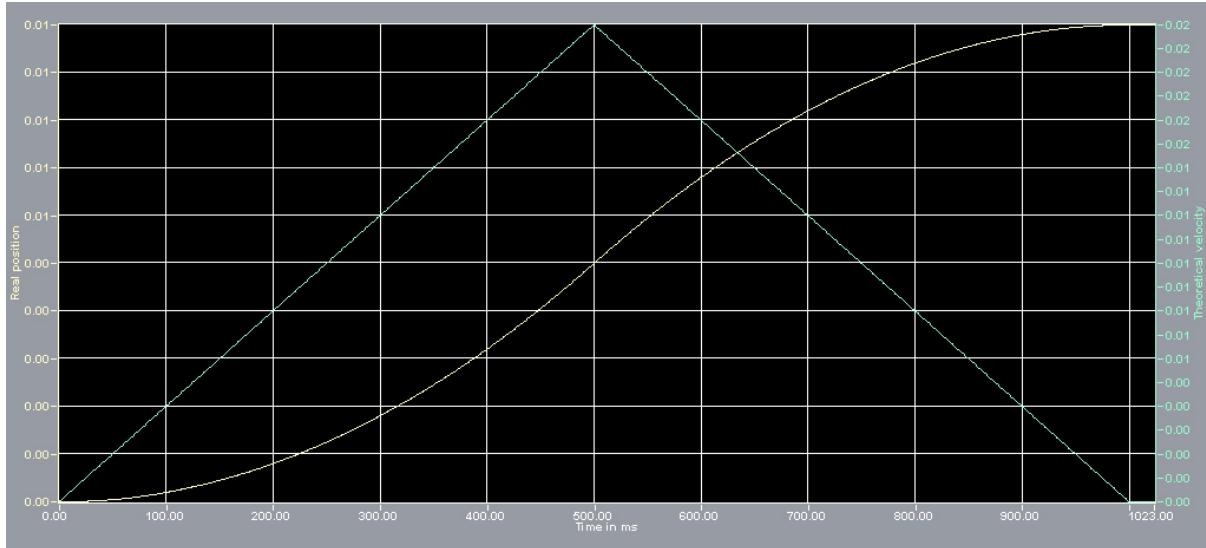
This type of movement is selected with the value 3 (for linear movement) or 19 (for rotary movement) of the MMD command or the parameter K202 (**MMD.<axis>=3** or **19** or **K202.<axis>=3** or **19**). The type of trajectory is defined by the parameter **K230** and the duration of the whole movement is defined by the parameter **K229**.

K	Name	Value	Comment
K229	Calculated movement time	-	Execution time for calculated movement profile selected by K230. Time = K229 x MLTI = K229 x 400µs
K230	Calculated movement profile	0 1 2 3	Triangular (speed) movement S-curve (full jerk) movement Sine modified movement Real sine movement

The target is defined by the parameter KL210 (refer to [§7.12.3](#)). There is no linear interpolation but at each MLTI a point is calculated (position versus time). For a rotary movement, the way of rotation is defined by the parameter K209 (refer to [§7.12.4](#)) and the limit of the position's counter by the parameter KL27 (refer to [§7.4.1](#)). The movement stops either when the target has been reached or when a command such as MVE, STP or BRK has been used. If such a command is used, the position is not lost and the movement can start again from this point.

Example:

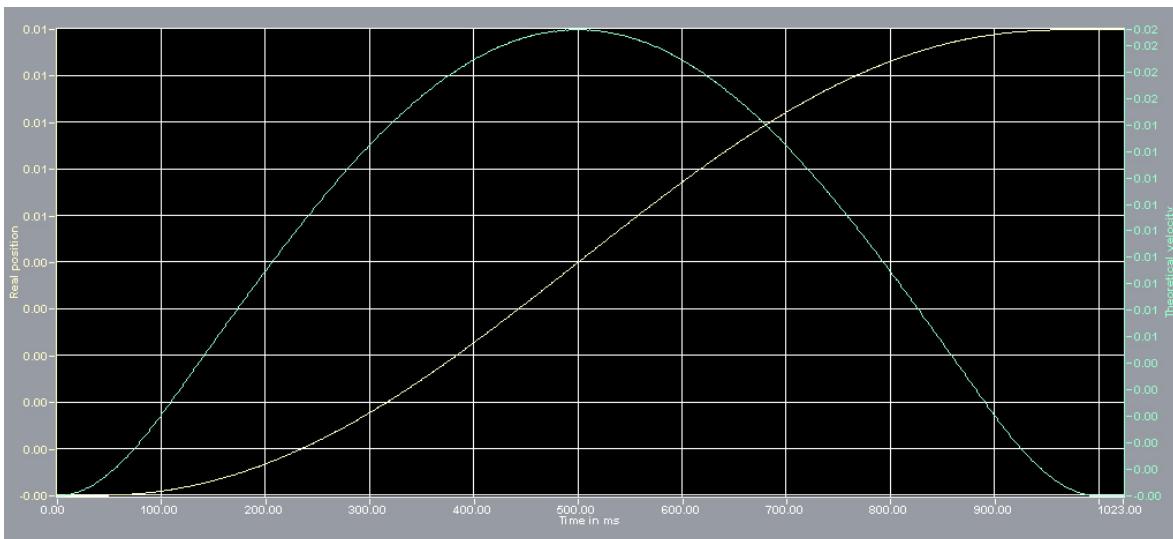
K230=0: Triangular movement (the graph represents the theoretical speed and position)



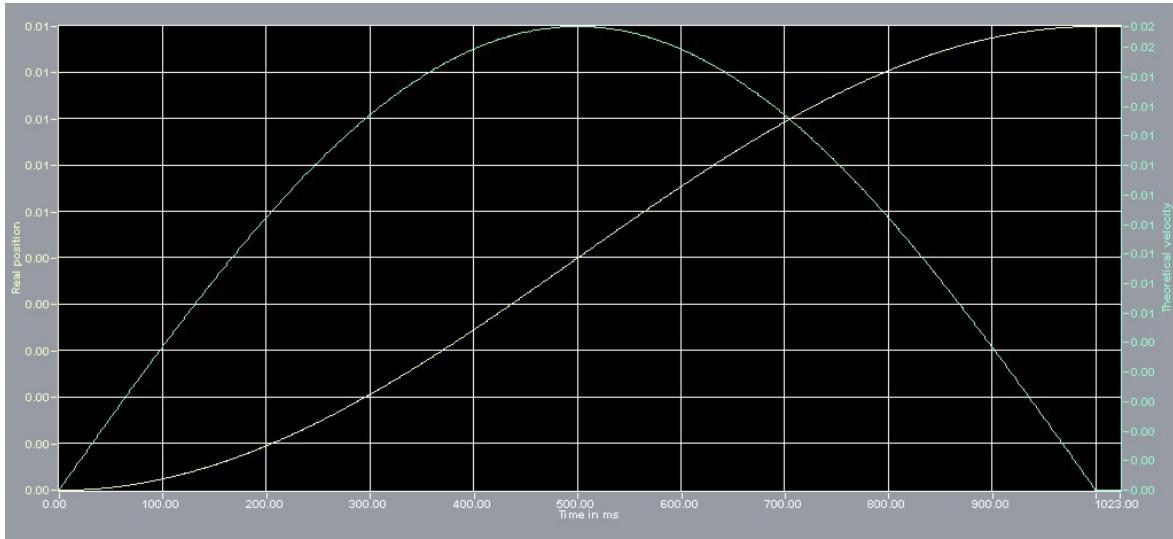
K230=1: S-curve movement (the graph represents the theoretical speed and position)



K230=2: Sine modified movement (the graph represents the theoretical speed and position)



K230=3: Real sine movement (the graph represents the theoretical speed and position)



8.3.5 Start movements: STA, STI and SMP commands

The **SMP** (Set Movement Parameter) command allows the user to set the movement parameters (position, speed, acceleration and jerk time) for the STA and STI command in only one command. **This command does not start a movement.**

Command	<P>	Comment
SMP.<axis>=<P1>,<PL2>[,<PL3>,<PL4>,<P5>]	<P1> <PL2> <PL3> <PL4> <P5>	Depth (1 to 7) Target position corresponding to KL210 Speed for the movement (optional parameter) corresponding to KL211 Acceleration for the movement (optional parameter) corresponding to KL212 Jerk time for the movement (optional parameter) corresponding to K213

Remark: The SMP command is not available for depth 0

Only integer 64-bit can be used for PL2, PL3 and PL4, and integer 32-bit for P1 and P5.

The **STA** (STArt movement) and **STI** (STart movement on Input) commands are used to initiate a movement using the values stored in a particular depth of the registers TARGET (KL210), SPD (KL211), ACC (KL212) and JRT (K213) as well as look-up table movements parameters stored at another depth than 0.

The STI command starts a movement according to the status of the digital input(s) defined by parameters K300 (refer to [§8.3.5.1](#)). As soon as the STI command is executed, the digital output(s) are set to the state defined by parameters K358 (refer to [§8.3.5.1](#)). A **SYNCHRO START** error (M64=63) is sent to the controller if the movement does not start within the time-out defined by the parameter K164 (time-out = K164 x 400µs).

Command	Comments on commands	<P1> values (depth)	<P2> values	<P2> bit#	Comments on bits mask <P2>
STA.<axis>=<P1>, <P2>	Starts a movement by using the movement parameters of the sub-indexes specified by <P1>, and <P2> is a mask of the movement parameter to be loaded (0 means all movements parameters are taken into account)	1 - 7	1	0	Gets target position (TARGET=KL210) from the specified index. Also possible during a trapezoïdal movement if MMC=1.
			2	1	Gets profile speed (SPD=KL211) from the specified index. Also possible during a trapezoïdal movement if MMC=1.
			4	2	Gets profile acceleration (ACC=KL212) from the specified index. Also possible during a trapezoïdal movement if MMC=1.
			8	3	Gets jerk filter time (JRT=K213) from the specified index.
			64	6	Gets profile type (MMD=K202) from the specified index.
			128	7	Gets user look-up table number (LTN=K203) and the look-up table number selection mode (K207) from the specified index
			256	8	Gets look-up table time (LTI=K204) from the specified index
			512	9	Gets user amplitude for look-up table (KL208) in mode K207=1
			1024	10	Gets rotary movement type selection (K209), only when the motor is not moving (bit#4 of SD1 at 0).
			2048	11	Gets the value of K230 from the specified index
			4096	12	Gets the value of K229 from the specified index

Remark: The depth where the movement characteristics are stored is defined in <P1>.

A mask of bits is used in <P2> to select which movement parameters are used among all parameters present in the depth defined in <P1>.

If **<P2> value = 0**, the controller considers that **all <P2> bits = 1** (bits# 0-12: 8191 decimal).

Do not use values 16 (bit#4) or 32 (bit#5) for <P2> of STA and STI commands.

STA and STI commands: bits mask description for values in <P2>				
Movement parameter in the mask	Value	bit#	Parameter function (refer to §7.12.3 & §8.3.1.1)	Behavior with S-curve concatenated movements (MMC=1)
KL210	1	0	Position to reach (alias is TARGET command)	Recaptured
KL211	2	1	Speed limit (alias is SPD command)	Recaptured
KL212	4	2	Acceleration limit (alias is ACC command)	Recaptured
K213	8	3	Jerk time (alias is JRT command)	Ignored
-	-	4	Reserved for future use	-
-	-	5	Reserved for future use	-
K202	64	6	Movement type (alias is MMD command)	Ignored
K203 / K207	128	7	LKT number (alias is LTN command) / LKT start and end positions	Ignored
K204	256	8	Execution time of the LKT (alias is LTI command)	Ignored
KL208	512	9	LKT maximum stroke	Ignored
K209	1024	10	Way of rotation movement	Ignored
K230	2048	11	Calculated movement (predefined profile) type	Ignored
K229	4096	12	Calculated movement (predefined profile) time	Ignored

The bits set to 1 mean that the values programmed at the defined depth will be kept to replace the values at the depth 0. The reserved bits must be set to 0.

Remark: When the whole **bits field is equal to 0**, the controller considers that **all bits are equal to 1!**

Example:

PWR.1=1, IND.1, WTM.1 commands are from now on omitted to avoid example overloading.

- Without SMP command

```

MMD.1=1;           // Selects the S-curve movement
ACC.1=500000L;     // Defines amax at the depth 0
SPD.1=200000L;     // Defines vmax at the depth 0
ACC:1.1=700000L;   // Defines amax at the depth 1
SPD:1.1=400000L;   // Defines vmax at the depth 1
TARGET:1.1=200000L; // Defines xfinal at the depth 1. The motor does not move.
ACC:2.1=600000L;   // Defines amax at the depth 2.
SPD:2.1=300000L;   // Defines vmax at the depth 2.
TARGET:2.1=800000L; // Defines xfinal at the depth 2. The motor does not move.
    
```

At this moment the motor has not moved yet but all the values are stored at different depths.

```

MVE.1=300000L,200000L,500000L; // The motor moves to the position 300000 with an acceleration
                                 // of 500000 and a speed of 200000 increments.
STA.1=2,7;                      // Starts a movement with the values contained at the depth 2.
                                 // The motor moves to the position 800000 with an acceleration
                                 // of 600000 and a speed of 300000 increments.
STA.1=1,7;                      // Starts a movement with the values contained at the depth 1.
                                 // The motor moves to the position 200000 with an acceleration of
                                 // 700000 and a speed of 400000 increments.
    
```

- With SMP command

```

MMD.1=1;           // Selects the S-curve movement
ACC.1=500000L;     // Defines amax at the depth 0
SPD.1=200000L;     // Defines vmax at the depth 0
SMP.1=1,200000L,400000L,700000L; // Defines xfinal, vmax and amax at the depth 1
SMP.1=2,800000L,300000L,600000L; // Defines xfinal, vmax and amax at the depth 2
    
```

At this moment the motor has not moved yet but all the values are stored at different depths.

```
MVE.1=300000L,200000L,500000L; // The motor moves to the position 300000 with an acceleration
                                  // of 500000 and a speed of 200000 increments.

STA.1=2,7;                      // Starts a movement with the values contained at the depth 2.
                                  // The motor moves to the position 800000 with an acceleration
                                  // of 600000 and a speed of 300000 increments.

STA.1=1,7;                      // Starts a movement with the values contained at the depth 1.
                                  // The motor moves to the position 200000 with an acceleration
                                  // of 700000 and a speed of 400000 increments.
```

Remark: STA and STI commands copy at the depth 0, the TARGET, SPD, ACC and JRT values contained at the depth where the movement will be executed. Therefore each STA or STI execution will replace the values stored at the **depth 0**. In the above example, after executing the first STA command (STA.1=2,7), the acceleration of 500000, the speed of 200000 which were initially at the depth 0 are lost. However, all the others at the depths other than 0 remain unchanged.

If an axis cannot execute the movement started by the **STI** command within the time defined by the parameter K164 (because another movement is being realized, eg.), the axis switch to error mode and the **SYNCHRO START** error (M64=63) is displayed on the ComET software. This will never happen with the STA command, where each available axis executes immediately the movement and the others do it as soon as they can.

8.3.5.1 DIN and DOUT with the STI command

- **DIN mask and status**

With the command STI, the synchronized start of the movements on one or several axes depends on the status (0 or 1) of one or several digital inputs (DIN). They are defined by the parameter **K300**.

K	Name	Comment
K300	STI input state mask	Defines a mask and the status on the digital input which will be tested to execute a STI command. bit#0-15: state.1 bit#16-31: state 0. Depth 0: DIN, depth 1: FDIN

- **DOUT mask and status**

When a synchronized movement starts on one or several axes with the command STI, one or several DOUT(s) are set or reset (0 or 1). They are defined by the parameters **K358**.

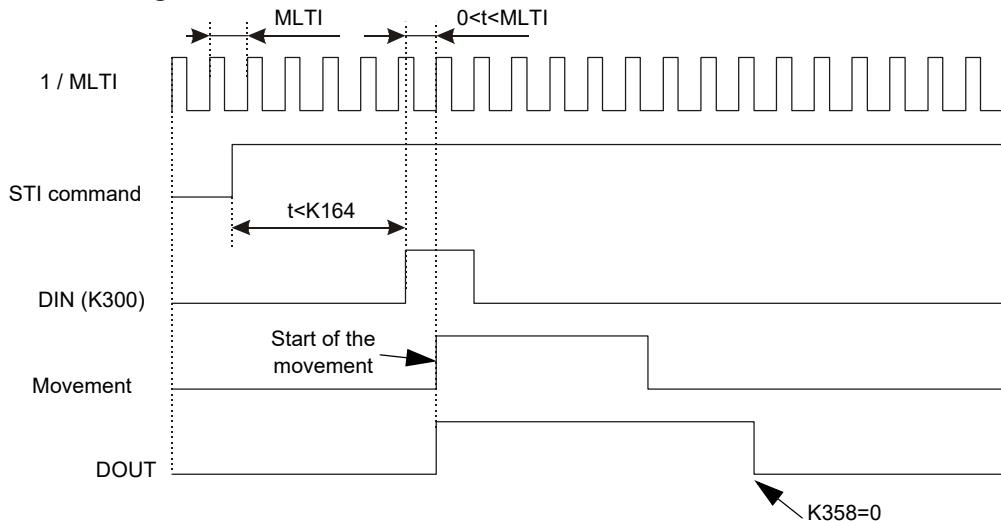
K	Name	Comment
K358	STI output state mask	Defines with a mask and the status of the digital output which will be modified when the movement starts. bit#0-15: state.1 bit#16-31: state 0. Depth 0: DOUT, depth 1: FDOUT

- **Time limit**

The parameter **K164** allows the user to define the time limit for the start of the STI command.

K	Name	Comment	Units
K164	STI time-out limit	Defines the time limit between the moment when the STI command is sent and the moment when the DIN are set to the required values. If this time is exceeded, a SYNCHRO START error (M64=63) appears on the controller.	[MLTI]

The ISO time is K164 x 400μs. Thus, K164 maximum value is 240 seconds.

STI command timing**Example:**

The user wants to start the movement when **DIN1=1** and **DIN10=0** and at the same time to set **DOUT2** and to reset **DOUT1**.

```

K300.1=0x02000001; // 0x02000001 means that the inputs DIN1(=1) and DIN10 (=0) will be tested to
// execute a STI command.
K358.1=0x00010002; // 0x00010002 means that the outputs DOUT2(=1) and DOUT1(=0) will be
// modified when a movement starts.
K164=6000; // time-out of 2.4 sec
.....
STI.1=1,1;
.....

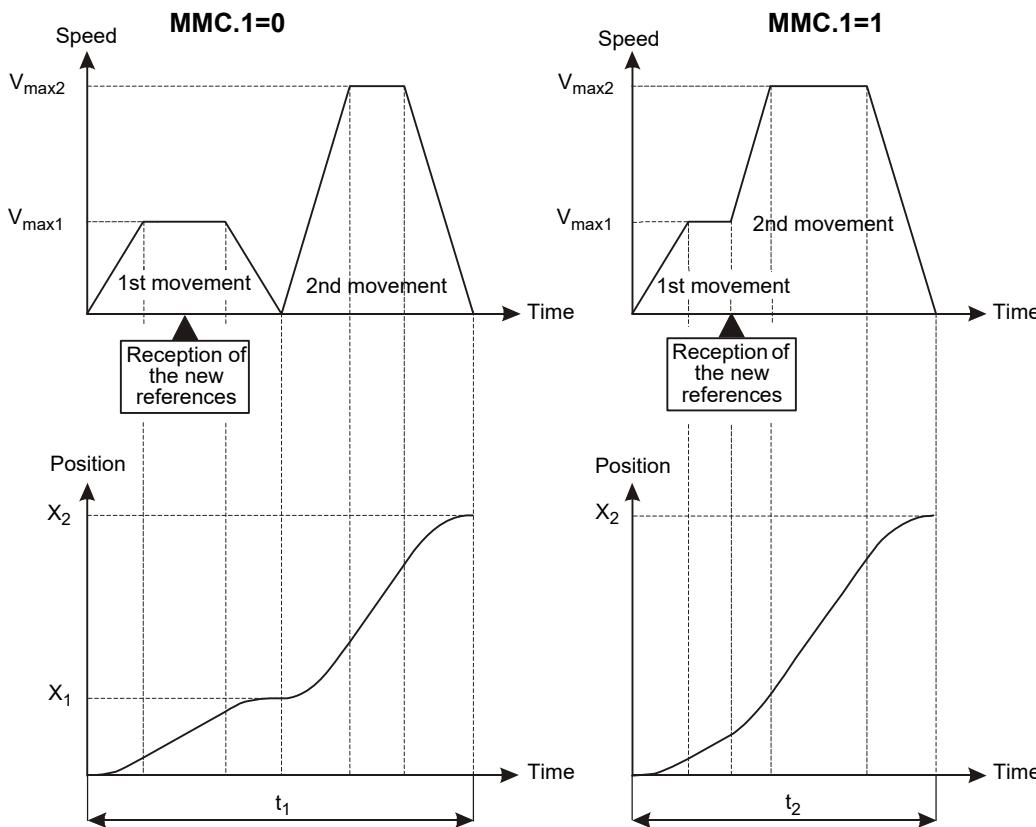
```

8.3.6 Concatenated movements: MMC command

If two MVE commands follow each other, the movements are made one after the other, so that the first must be finished before the second starts. However, some applications need to change the final position (x_{final}) to reach during a movement. The **MMC** command (**Movement Mode Concatenated**), which is an alias of the parameter **K201** allows the user to realize this function and even more, since also speed and acceleration can be changed during the movement.

K	Alias	Name	Value	Comment
K201	MMC	Concatenated movements selection	0 1 2 3	Concatenated movements disabled Concatenated movements for MMD=1, 17 & 24 Continuous back and forth movements enabled for MMD = 3, 10, 19 and 26 movement (can be stopped by MVE, BRK, HLT, HLB and HLO commands) One back and forth movements enabled for MMD = 3, 10, 19 and 26 (can be stopped by BRK, HLT, HLB and HLO commands)

When a concatenated movements mode is selected (**MMC.1=1**), and a second movement starts, the controller immediately calculates the new speed trajectory and the motor switches from one movement to the other one without going through the 0 speed. The first movement does not end and the controller immediately considers the new position, the new maximum speed and the new maximum acceleration.



Remark: For the same final position, times t_1 and t_2 are not identical; time t_2 is shorter.
If the MMC command does not appear in the sequence, the movements without concatenation mode are used by default.

Example:

Concatenated movements are simple to execute. Just select the concatenated movements mode and execute the memorized movements at different depths.

```

MMD.1=1;                                // Selects the S-curve movement.
MMC.1=1;                                // Selects the concatenated movements mode.
ACC.1=500000L;                            // Defines  $a_{max}$  at the depth 0.
SPD.1=200000L;                            // Defines  $v_{max}$  at the depth 0.
SMP.1=1,200000L,400000L,700000L;        // Defines  $X_{final}$ ,  $v_{max}$  and  $a_{max}$  at the depth 1
MVE.1=100000L;                            // The motor moves to the position 100000
STA.1=1,...                                // The memorized movement at the depth 1 is started. If the first
                                            // movement is not finished, it is interrupted. The motor will
                                            // increase its speed because the movement speed at the depth
                                            // 1 is higher than that of the previous movement and it will finally
                                            // stop at the position 200000.

```

8.3.7 CAM command

The **CAM** command (which is an alias of the parameter **KF205**) decreases by a certain percentage the **kinematic quantities** of a movement such as speed, acceleration and jerk time (but not the position to reach). Similarly, **time quantities** such as time, lapse of time between two movements with the WTT command, execution time of a look-up table movement, etc., are increased in the same proportion. The formulae which calculates the new values of these quantities according to the number included in the CAM command are given below.

For kinematic quantities the formula is as follows:

- speed: $\frac{\text{Kinematic quantity} \cdot \text{CAM}}{100}$

- acceleration: $\frac{\text{Kinematic quantity} \cdot \text{CAM}^2}{100^2}$

For time quantities the formula is: $\frac{\text{Time quantity} \cdot 100}{\text{CAM}}$

K	Alias	Name	Comments	Units
KF205	CAM	CAM value	Came value (in percent). Decreases kinematic quantities (speed, acceleration) and increases time quantities	[%]

Remark: The CAM alias uses the same syntax as the parameter KF205.
 The value programmed with the CAM command remains always active. In order to come back to the basic values, the command CAM.1=100.0F must be entered.
 The CAM command does not have any effect on the Fast Trigger function.

After the CAM calculation, the maximum speed as well as acceleration and deceleration can be monitored with **ML211** and **ML212**.

M	Name	Comment	Units
M211 ML211	Absolute maximum speed	Absolute maximum speed after CAM calculation. The monitoring M211 corresponds to the LSL part of the monitoring ML211	[usi], [rusi]
M212 ML212	Absolute max. acc./deceleration	Absolute max. acc./deceleration after CAM calculation. The monitoring M212 corresponds to the LSL part of the monitoring ML212	[uai], [ruai]

Example:

```

MMD.1=1;           // S-curve movement selection
ACC.1=500000L;     // Definition of amax.
SPD.1=200000L;     // Definition of vmax.
MVE.1=300000L;     // The motor moves to the position 300000 with an acceleration of 500000 and a
                   // speed of 200000.
CAM.1=20;          // Global diminution to 20%.
MVE.1=5000L;       // The motor moves to the position 5000 with a speed and an acceleration decreased
                   // of 80%. Only 20% of these values are kept. The new speed will be:
                   
$$\frac{200000 \cdot 20}{100} = 40000 \text{ [usi]}$$
 and the new acceleration will be:
                   
$$\frac{500000 \cdot 20^2}{100^2} = 8000 \text{ [uai].}$$


```

8.3.8 STE command

The **STE** command (**STE**p) enables the motor to make a jump in the movement. It is used by the ComET program to find out the right values of the regulator different gains.

Remark: This command must be used with the **caution** as the movement speed is very high.

Command	<P1> or <PL1>	Comment	Units
STE_ADD.<axis>=<P1> or <PL1>	-2 ⁴⁷ to (2 ⁴⁷ -1)	Performs a position step (infinite acceleration) of the specified size in the positive direction	[upi] [rupi]
STE_SUB.<axis>=<P1> or <PL1>	-2 ⁴⁷ to (2 ⁴⁷ -1)	Performs a position step (infinite acceleration) of the specified size in the negative direction	[upi] [rupi]
STE_ABS.<axis>=<P1> or <PL1>	-2 ⁴⁷ to (2 ⁴⁷ -1)	Performs a position step (infinite acceleration) to the specified (absolute) position	[upi] [rupi]

Although the STE command is not an alias, the syntax STE.1+=1000 or STE.1-=1000 are allowed to make **relative** jump movements, because an absolute jump movement can be dangerous if the motor is far away from the position to reach.

Example:

```
STE_SUB.1=1000; or STE.1-=1000; // Relative jump movement. The motor realizes a jump movement of
// 1000 increments in the negative direction.
```

```
STE_ADD.1=1000; or STE.1+=1000; // Relative jump movement. The motor realizes a jump movement of
// 1000 increments in the positive direction.
```

```
STE_ABS.1=1000; or STE.1=1000; // Absolute jump movement. The motor makes a jump movement
// from its current position to the position of 1000 increments.
```

Remark: In the compiled sequence, only the following syntax can be used: STE_ADD.<axis>=<P1> or <PL1>, STE_SUB.<axis>=<P1> or <PL1> and STE_ABS.<axis>=<P1> or <PL1>.

8.3.9 BRK and STP commands

The **BRK** command (**BRaKe**) stops a movement with the deceleration programmed in the parameter KL206 but does not stop the sequence. It is also used with an infinite rotary movement (refer to [§8.3.3](#)).

The **STP** command (**SToP**) behaves in the same way as the BRK command but the deceleration is infinite. The braking is then very sudden. This command must be used only in case of emergency.

Caution: The sudden deceleration set with the STP command can damage the mechanical parts of the system.

Remark: BRK and STP commands do not stop the sequence; thus, the movements defined in a sequence will be executed.
Those commands have no effect in the ITP mode.

Command	Comment
BRK.<axis>	Stops the movement using the deceleration programmed in parameter KL206 but it does not stop the sequence. Stops an infinite rotary movement.
STP.<axis>	Stops the movement using an infinite deceleration but it does not stop the sequence.

When one of the conditions defined by the parameters **K298** is true, a BRK command is activated. It has the same function as a BRK.<axis> command.

K	Name	Comment
K298	Input edge mask for BRK	Defines the mask of the digital inputs (rising & falling edge) to stop the movement like a BRK command. bit#0-15: rising edge. bit#16-31: falling edge. Depth 0: DIN, depth 1: FDIN

For the STP command, the deceleration is in fact not necessarily infinite as it depends on the speed and the jerk time of the current movement. This value can be calculated as follows:

$$\text{Deceleration } [\text{m/s}^2] = \frac{\text{M10 } [\text{m/s}](\text{at the command execution})}{(\text{MLTI } [\text{s}] \times \text{K213 } [\text{Inc}])}$$

If K213=0 Inc, the controller will use the same formula with K213=1 Inc.

The monitoring ML3 provides permanently the braking distance if we execute a BRK command. This distance is updated at the MLTI frequency taking into account KL206:0 for the deceleration.

M	Name	Comment	Unit
ML3	Braking distance	Gives the braking distance if BRK command is executed	[dpi]

This information cannot be compared with ML0 / M0. Indeed ML0 / M0 is updated after passing through the filters of the PLTI.

Restrictions: ML3 is not available for external reference modes (K61! = 1) as well as for interpolated mode (ITP).

Warning: If you wait until you are at a distance of ML3 from the target and you start the execution of the BRK command, the final position will be farther than the expected target due to the time of the communication!

If you want to reach a position with the command BRK and ML3, you have to consider the communication time when reading the monitoring ML3 as well as when sending the BRK command.

8.4 Trajectory filter

The trajectory filter is used to remove the unwanted oscillations. To do so, up to 4 filters (then 4 frequencies) corresponding to depths 0 to 3 can be activated. Only two parameters, **K291** and **K292**, are needed to set the trajectory filter. The **TFTD** (Trajectory Filter Time Delay) and **TFAR** (Trajectory Filter Amplitude Ratio) commands are alias of the parameters **K291** and **K292** respectively and use the same syntax.

K	Alias	Name	Comment
K291	TFTD	Time delay	Time delay of the trajectory filter (depth 0 to 3)
K292	TFAR	Amplitude ratio	Amplitude ratio of the trajectory filter (depth 0 to 3)

- The time delay (K291) is defined as follows:

$$K291 = \frac{1}{2} \times \frac{f_{MLTI}}{f_{osc}} = \frac{1}{2} \times \frac{T_p}{T_{MLTI}} \quad [\text{Units: MLTI}]$$

With

f_{MLTI} = Frequency of the manager loop time interrupt = 2500 Hz.

f_{osc} = Oscillation frequency [Hz]

T_p = Oscillation time period [s]

T_d = Time delay [s] and $T_d = T_p / 2$

When K291=0 (min. value), the filter is inactive

When K291=4095 (max. value), the minimal frequency is 0.305 Hz.

- The amplitude ratio (K292)

When K292=0 (min. value), the filter is inactive

When K292=2048, the filter absorbs the complete frequency (notch filter).

Remark:

The trajectory filter can be used only with K61=1, 3, 4 and 36.

The time delay induces a delay in the movement.

The trajectory filter is inactive during an IND and SLS commands as well as the initialization time of the filter.

To improve the immunity of the trajectory filter (in case of variation of the oscillation frequency), it is possible to set 2 trajectory filters: the first one at the minimum frequency of the variation and the second one at the maximum frequency of the variation. For example, an oscillation at 10Hz \pm 1Hz will induce the 1st frequency at 9Hz and the 2nd one at 11Hz.

The trajectory filter features has no impact on the regulation but only on the trajectory.

The filter process takes some processor time. It is important to reduce, at the strict minimum, the number of filter used in the application (to avoid MLTI interrupt time overflow).

8.4.1 Setting phases

- Phase 1: frequency determination**

- Determine, without the 'trajectory filter' feature, the movement size which introduces the biggest oscillation with the typical acceleration, speed and jerk time.
- Analyses the tracking error with the 'Scope' tool of ComET and determine the time period and/or the frequency period of this oscillation

- Phase 2: trajectory filter setting**

- Compute the time delay for the oscillation to be removed

- Set the value in depth 0 of the parameter K291
- **Phase 3: testing of reduction of the oscillation**
 - Analyses the tracking error with the 'Scope' tool
 - Adjust the parameter K292 to completely remove the oscillation
- **Phase 4: Jerk time adjustment**
 - Test the typical movement and try to reduce as much as possible the jerk time

Then, do the same at the other depths to remove the other oscillations.

8.5 Digital inputs / outputs

8.5.1 Digital inputs of the controller

The monitorings **M50**, **M54** and **M52** allow the user to read the status of the standard digital inputs of one axis or two axes and the fast digital inputs of the controller respectively. The DIN, CDIN and FDIN commands are alias of these monitorings and use the same syntax.

M	Alias	Name	Comment
M50	DIN	DIN status	Gives the status of the digital inputs of the controller per motor: DIN1 (bit#0),..., to DIN10 (bit# 9).
M54	CDIN	CDIN status	Gives the status of the digital inputs of the controller for both motors (combination of M50 from both axes)
M52	FDIN	FDIN status	Gives the status of the fast digital inputs of the controller: FDIN1 (bit# 0) to FDIN6 (bit# 5).

M50					
Bit#9	Bit#8	Bit#7 to 3	Bit#2	Bit#1	Bit#0
DIN10 of the axis	DIN9 of the axis	All 0	DIN3 of the axis	DIN2 of the axis	DIN1 of the axis

M54												
Bit#31 to 20	Bit#19	Bit#18	Bit#17 to 13	Bit#12	Bit#11	Bit#10	Bit#9	Bit#8	Bit#7 to 3	Bit#2	Bit#1	Bit#0
All 0	DIN10 of axis 1	DIN9 of axis 1	All 0	DIN3 of axis 1	DIN2 of axis 1	DIN1 of axis 1	DIN10 of axis 0	DIN9 of axis 0	All 0	DIN3 of axis 0	DIN2 of axis 0	DIN1 of axis 0

Remark: The fast digital inputs are dedicated to fast position capture, pulse direction and differential input. When it is not connected, the value is 1.

A bit equal to 1 means that the corresponding digital input is activated and equal to 0 means that the corresponding digital input is deactivated. Refer to the corresponding '**Hardware Manual**' to know the number and location of digital inputs of your controller.

Here is explained the use of the digital inputs bits values of the controllers:

DIN # on AccurET 400 / 600 VHP 48 / 100	10	9	-	-	-	-	-	-	3	2	1
DIN # on AccurET 48 / 300	10	9	-	-	-	-	-	-	-	2	1
FDIN # on AccurET 400 / 600	-	-	-	-	-	-	-	4	3	2	1
FDIN # on AccurET 48 / 300 VHP 48 / 100	-	-	-	-	6	5	4	3	2	1	0
bit#	9	8	-	-	5	4	3	2	1	0	
Decimal value	512	256	-	-	32	16	8	4	2	1	
Hexadecimal value	200	100	-	-	20	10	8	4	2	1	

A simple conversion in binary of the value shown by DIN (in hexadecimal) is enough to know which digital inputs are activated and deactivated.

Example:

DIN values		DIN10	DIN9	-	-	-	-	-	DIN3	DIN2	DIN1
		Bit#9	Bit#8	-	-	-	-	-	Bit#2	Bit#1	Bit#0
3	3	0	0	-	-	-	-	-	0	1	1
769	301	1	1	-	-	-	-	-	0	0	1
Decimal	Hexa.	Binary									

In the first line of the above-mentioned example, DIN1 and DIN2 are activated.

DIN.1; // Reads the digital inputs state. The controller gives the value 3 in hexa, because 0x3 in binary is // 0000000011 and each bit represents one of the digital inputs, from right to left.

In the second line of the above-mentioned example DIN1, DIN9 and DIN10 are activated.

DIN.1; // Reads the digital inputs state. The controller gives the value 301 in hexa, because 0x301 in // binary is 1100000001 and each bit represents one of the digital inputs, from right to left.

Remark: Some digital inputs can also be used for external limit switches and external home switch, if these switches are present in the application. Refer to [§7.9](#) for more information.
It is possible to reset the controller (hardware reset) by using a digital input like with a RSD command. Refer to [§7.5.1](#) for more information.

8.5.2 Digital inputs of the optional board

The monitoring **M55** allows the user to read the status of the digital inputs of the optional board. The XDIN commands is an alias of this monitorings and uses the same syntax. The optional board cannot be used on the VHP controllers.

M	Alias	Name	Comment
M55	XDIN	XDIN status	Gives the status of the digital inputs of the optional board: XDIN1 (bit#0) to XDIN8 (bit# 1).

The optional board has 8 digital inputs (refer to the '**User's Manual**' of the AccurET's I/O optional board for more information).

Remark: The 8 digital inputs are available for both axes.

A bit equal to 1 means that the corresponding digital input is activated and equal to 0 means that the corresponding digital input is deactivated. The digital inputs bits values are as follows:

XDIN #	8	7	6	5	4	3	2	1
bit#	7	6	5	4	3	2	1	0
Decimal value	128	64	32	16	8	4	2	1
Hexadecimal value	80	40	20	10	8	4	2	1

Example:

A simple conversion in binary of the value shown by the monitoring M55 (in hexadecimal) is enough to know which digital inputs are activated and deactivated.

XDIN values		XDIN8 Bit#7	XDIN7 Bit#6	XDIN6 Bit#5	XDIN5 Bit#4	XDIN4 Bit#3	XDIN3 Bit#2	XDIN2 Bit#1	XDIN1 Bit#0		
16	10	0	0	0	1	0	0	0	0		
255	FF	1	1	1	1	1	1	1	1		
Decimal	Hexa.	Binary									

In the first line of the above-mentioned example, XDIN5 is activated.

M55.1; // Reads the digital inputs state. The controller gives the value 0x10 (in hexadecimal), because

```
// 0x10 in binary is 00010000 and each bit represents one of the eight digital inputs, from right
// to left.
```

In the second line of the above-mentioned example, XDIN1 to XDIN8 are activated.

```
M55.1; // Reads the auxiliary digital inputs state. The controller gives the value 0xFF (in hexadecimal),
// because 0xFF in binary is 11111111 and each bit represents one of the eight digital inputs,
// from right to left.
```

8.5.3 Digital outputs of the controller

The parameter **K171** and the common parameter **C5** allow the user to activate or deactivate the standard digital outputs and the fast digital outputs of the controller. The DOUT and FDOUT are alias of these parameters and use the same syntax.

K / C	Alias	Name	Comment
K171	DOUT	DOUT value	Activates / deactivates the digital outputs of the controller per motor: DOUT1 (bit#0) and DOUT2 (bit# 1).
C5	FDOUT	FDOUT value	Activates / deactivates the fast digital outputs of the controller: FDOUT1 (bit#0) to FDOUT4 (bit#3). The FDOUT are common to both axes.

A bit equal to 1 means that the corresponding digital output is activated and equal to 0 means that the corresponding digital output is deactivated. Refer to the corresponding '**Hardware Manual**' to know the number and location of digital outputs of your controller.

The status of the DOUT and FDOUT can be read with the monitorings **M171** and **M172**:

M	Name	Comment
M171	DOUT status	Gives the status of the digital outputs of the controller per motor at the beginning of the PLTI. Takes DOUT (K171) into account
M172	FDOUT status	Gives the status of the fast digital outputs of the controller at the beginning of the PLTI.

- For version older than 2.04A, the monitorings **M171** and **M172** represent only the parameters **K171** and **C5** respectively and do NOT represent the real value of the DOUTs according to their functions (Fast Trigger,...).
- For version from 2.04A, the monitorings **M171** and **M172** represent the DOUT / FDOUT state at the beginning of each PLTI and take into account the parameters **K171** and **C5** respectively and the associated functions (Fast Trigger,...).

Here is explained the use of the digital outputs bits values of the controllers:

DOUT #	-	-	2	1
FDOUT #	4	3	2	1
bit#	3	2	1	0
Decimal and hexadecimal value	8	4	2	1

A simple conversion in binary of the value shown by the monitoring M171 (in hexadecimal) is enough to know which digital inputs are activated and deactivated.

Example:

DOUT values		DOUT2 Bit#1	DOUT1 Bit#0
3	3	1	1
2	2	1	0
Decimal		Binary	

In the first line of the above-mentioned example, DOUT1 and 2 are activated.

```
DOUT.0=3; // Activates DOUT1 and 2.
```

In the second line of the above-mentioned example, DOUT2 is activated and DOUT1 is deactivated.

```
DOUT.0=2; // Activates DOUT2 and deactivates DOUT1.
```

In addition, the DOUT and FDOUT can be used to output several other signals: 1Vpp sine and cosine signals (not interpolated), 1Vpp index signal, the TransnET cycles, the MLTI or PLTI cycles, and finally signals used in the Fast Triggers feature. To do so, the following parameters must be used:

K	Name	Comment
K350	Mask for SIN	FDOUT mask for 1 Vpp SIN signal (available at depth 1; bit#0-15: non-inverted, bit#16-31: inverted)
K351	Mask for COS	FDOUT mask for 1 Vpp COS signal (available at depth 1; bit#0-15: non-inverted, bit#16-31: inverted)
K352	Mask for index	FDOUT mask for 1 Vpp index signal (available at depth 1; bit#0-15: non-inverted, bit#16-31: inverted)
C353	Mask for TransnET	FDOUT mask for TransnET cycle (available at depth 1, bit#0-15: non-inverted, bit#16-31: inverted)
C354	Mask for MLTI	FDOUT mask for MLTI cycle (available at depth 1, bit#0-15: non-inverted, bit#16-31: inverted)
C355	Mask for PLTI	FDOUT mask for PLTI cycle (available at depth 1, bit#0-15: non-inverted, bit#16-31: inverted)
K359	DOUT for Fast Trigger	DOUT mask for Fast Trigger signal (bit#0-15: non-inverted, bit#16-31: inverted)
C359	FDOUT for Fast Trigger	FDOUT mask for Fast Trigger signal (available at depth 1, bit#0-15: non-inverted, bit#16-31: inverted)

Remark: In case of multiple usage of one DOUT, the **BAD DOUT SETTING** error (M64=45) is raised.
 In case of multiple usage of one FDOUT, the **BAD FDOUT SETTING** error (M64=42) is raised.
 When a DOUT/FDOUT is reserved for the Fast Triggers, it is possible to directly set or reset its state thanks to the command TRR (refer to [§8.9.6.1](#)).

8.5.4 Digital outputs of the optional board

The common parameter **C6** allows the user to activate or deactivate the digital outputs of the optional board. The XDOUT command is an alias of this parameter and uses the same syntax.

C	Alias	Name	Comment
C6	XDOUT	XDOUT value	Activates / deactivates the digital outputs of the optional board XDOUT1 (bit#0) to XDOUT8 (bit#1). The XDOUT are common to both axes.

The optional board has 8 digital outputs (refer to the '**User's Manual**' of the AccurET's I/O optional board for more information).

Remark: The 8 digital outputs are available for both axes.

The status of the XDOUT can be read with the monitoring **M170**:

M	Name	Comment
M170	XDOUT status	Gives the status of the digital outputs of the optional board. Takes XDOUT (C6) into account

A bit equal to 1 means that the corresponding digital output is activated and equal to 0 means that the corresponding digital output is deactivated. The digital outputs bits values are as follows:

XDOUT #	8	7	6	5	4	3	2	1
bit#	7	6	5	4	3	2	1	0
Decimal value	128	64	32	16	8	4	2	1
Hexadecimal value	80	40	20	10	8	4	2	1

Example:

XDOUT value		XDOUT8 Bit#7	XDOUT7 Bit#6	XDOUT6 Bit#5	XDOUT5 Bit#4	XDOUT4 Bit#3	XDOUT3 Bit#2	XDOUT2 Bit#1	XDOUT1 Bit#0
FF	255	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	1
AA	170	1	0	1	0	1	0	1	0
Hexa.	Decimal	Binary							

In the first line of the above-mentioned example, XDOUT1 to 8 are activated.

```
C6.1=255; // Activates XDOUT1 to 8.
```

In the second line of the above-mentioned example, XDOUT1 is activated and XDOUT2 to 8 are deactivated.

```
C6.1=1; // Activates XDOUT1 and deactivates XDOUT2 to 8.
```

In the third line of the above-mentioned example, XDOUT2, 4, 6 and 8 are activated and XDOUT1, 3, 5 and 7 are deactivated.

```
C6.1=170; // Activates XDOUT2, 4, 6 and 8 and deactivates XDOUT1, 3, 5 and 7.
```

8.5.5 Digital outputs triggered by RTV

To set a digital output from an application running on a host PC, the basic way is to write the corresponding value in registers K171 for a DOUT, C5 for a FDOUT or C6 for a XDOUT. For demanding applications, a faster alternative is available to quickly set or reset a digital output using an RTV (refer to [§10](#) for more details about RTV). To enable this feature, the user must first reserve the DOUT, FDOUT or XDOUT to be managed with an RTV. This operation is done with register C358.

C	Name	Comment
C358	Mask for output managed over RTV slots	Bit mask of DOUT, FDOUT or XDOUT triggered by a RTV slot.

The bit mask of register C358 is the following one:

C358																
Bit#23	Bit#22	Bit#21	Bit#20	Bit#19	Bit#18	Bit#17	Bit#16	Bit#11	Bit#10	Bit#9	Bit#8	Bit#3	Bit#2	Bit#1	Bit#0	
XDOUT 8	XDOUT 7	XDOUT 6	XDOUT 5	XDOUT 4	XDOUT 3	XDOUT 2	XDOUT 1	FDOUT 4	FDOUT 3	FDOUT 2	FDOUT 1	DOUT 2 of axis 1	DOUT 1 of axis 1	DOUT 2 of axis 0	DOUT 1 of axis 0	

Remark: In case of multiple usage of a digital output, an error is raised depending on the output type:
ERR CFG DOUT error (M64=45) is raised if a DOUT is reserved for more than one feature.
ERR CFG FDOUT error (M64=42) is raised if a FDOUT is reserved for more than one feature.
ERR CFG XDOUT error (M64=39) is raised if a XDOUT is reserved for more than one feature.

The digital outputs are finally linked to the RTV slot with the command RTVDOUT. This command must be sent to first axis only. If it is sent to axis 1, an error is raised.

Command	<P1>	Comment
RTVDOUT.<axis> = <P1>	Slot number	Assign a RTV slot to set the DOUT / FDOUT / XDOUT reserved with register C358. This command must be sent to first axis only. The link is broken when slot_number = -1.

The **BAD CMD PARAM** error (M64=70) is raised if:

- RTVDOUT is sent to the second axis of the controller.
- RTVDOUT is sent without parameter or with more than one parameter.

The **BAD SLOT TRANSNET** error (M64=38) is raised if:

- RTVDOUT is sent with a slot_number greater than 511 (TransnET running at 100 µs).

8.6 Analog inputs / outputs

8.6.1 On AccurET modular controllers

No analog output is present on the position controllers. It is possible to install in the controller an optional board adding up to 4 analog inputs and 4 analog outputs (refer to the '**User's Manual**' of the AccurET's I/O optional board for more information).

8.6.1.1 Analog inputs on the optional board

The optional board has 4 analog inputs (16 bits ADC). Refer to the '**User's Manual**' of the AccurET's I/O optional board for more information.

Remark: The 4 analog inputs are available for both axes.

The offset and amplitude errors of the input signal can be removed with the common parameters **C17** and **CF18**. The XAIO and XAIA commands are an alias of these common parameters and use the same syntax.

C	Alias	Name	Comment
C17	XAIO[:depth].<axis>=<P1>	Analog Input Offset	Modifies the offset of the signal sent to the analog inputs (XAIN1 to XAIN4). The 4 depths correspond to the 4 analog inputs.
CF18	XAIA[:depth].<axis>=<P1>	Analog Input gAin	Modifies the amplitude of the signal sent to the analog inputs (XAIN1 to XAIN4). The 4 depths correspond to the 4 analog inputs.

The monitoring **M56** allow the user to read the status of the analog inputs of the optional board. The XAIN command is an alias of this monitoring and use the same syntax.

M	Alias	Name	Comment
M56	XAIN	XAIN status	Reads the real state of the analog inputs of the optional board. The 4 depths of the monitoring correspond to the 4 analog inputs (depth 0 for XAIN1 up to depth 3 for XAIN4).

The input voltage of the analog input must be included between +10V and -10V between XAIN+ and XAIN-. +10V corresponds to +32767 increments and -10V to -32768 increments. Here is the formula calculating the voltage from the value contained in the monitoring M56:

$$U[V] = \frac{M56}{32768} \cdot 10$$

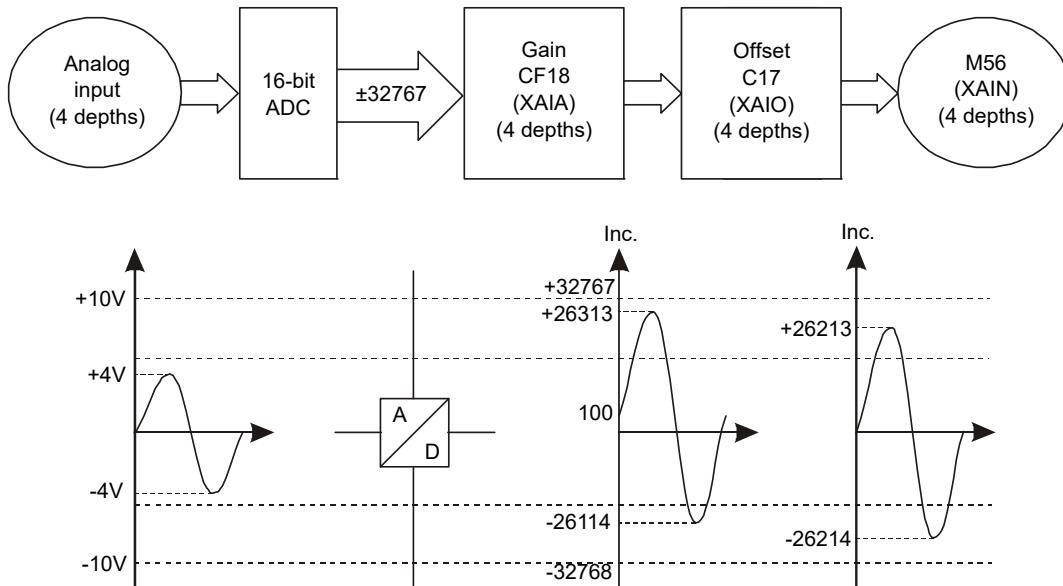
Example:

M56 : 2 . 1 Reads the voltage on the analog input 3 (XAIN3). For example, if the controller gives the value 5284 increments, use the above-mentioned formula to calculate the corresponding voltage measurable on XAIN3: here, 5284 increments corresponds to ~ +1,61V.

Here is the formula giving the values on the analog inputs after the amplitude and offset correction:

$$M56 = (\text{Input signal} \times CF18) + C17$$

Example with CF18=2.0F and C17=-100.



8.6.1.2 Analog outputs on the optional board

The optional board has 4 analog outputs (refer to the '**User's Manual**' of the AccurET's I/O optional board for more information).

Remark: The 4 analog outputs are available for both axes.

The analog outputs can be used from two different modes:

- A 'direct' mode
- A 'source register' mode
- **Analog outputs with 'direct' mode**

The common parameter **C7** allows the user to allocate a defined value (defined voltage) to the analog outputs (XAOUT1 to XAOUT4) of the optional board. To do so, the common parameter C30 must be equal to 0. The XAOUT command is an alias of this parameter and uses the same syntax.

C	Alias	Name	Comment
C7	XAOUT	Analog output value	Gives the value of the analog outputs (XAOUT1 to XAOUT4) of the optional board compared to the GND. The 4 depths of the parameter correspond to the 4 analog outputs.

As the common parameter is referenced to the GND, +10VDC corresponds to 32767 increments and -10VDC to -32768 increments.

Here is the formula calculating the voltage from the values contained in the common parameter C7:

$$U[V] = \frac{C7}{32768} \cdot 10$$

Example:

C7:2.1=9600; // 9600 is converted into +2.92VDC on XAOUT3. This can be measured between // XAOUT3+ and GND on the optional board.

Remark: If the measure is done between XAOUT3+ and XAOUT3-, the voltage will be multiplied by two $2.92 \times 2 = 5.84$ VDC (refer to the '**User's Manual**' of the AccurET's I/O optional board for more information).

The output voltage of the analog output is included between +20 VDC and –20 VDC between XAOUT+ and XAOUT-.

The digital analog converter (DAC) may induce a small offset and amplitude error. These errors can be removed with the common parameters **C27** and **CF28**. The XAOO and XAOA commands are an alias of these common parameters and use the same syntax.

C	Alias	Name	Comment
C27	XAOO[:depth].<axis>=<P1>	Analog Output Offset	Modifies the offset of the signal sent to the analog outputs (XAOUT1 to XAOUT4). The 4 depths correspond to the 4 analog outputs.
CF28	XAOA[:depth].<axis>=<P1>	Analog Output gAin	Modifies the amplitude of the signal sent to the analog outputs (XAOUT1 to XAOUT4). The 4 depths correspond to the 4 analog outputs.

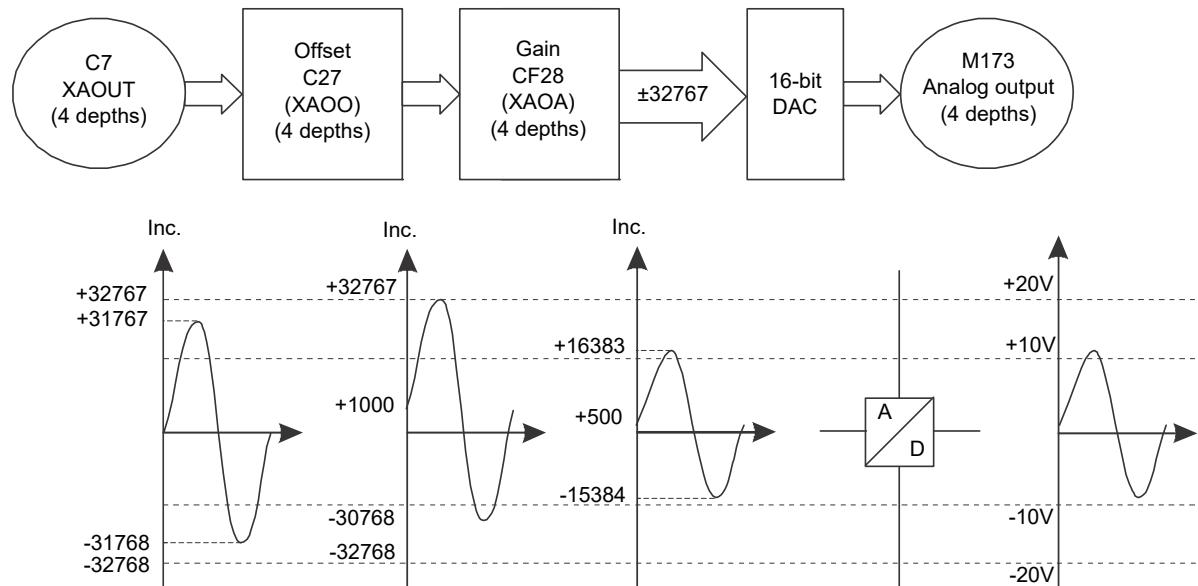
The monitorings **M173** allows the user to read the value of the 4 analog outputs by taking the common parameters C27 and CF28 into account:

M	Name	Comment
M173	Analog output status	Gives the value on the analog outputs. The 4 depths correspond to the 4 analog outputs.

Here is the formula giving the values on the analog outputs after the amplitude and offset correction:

$$M173 = (C7 + C27) \cdot CF28$$

Example:



- **Analog outputs with 'source register' mode**

The user can allocate to the 4 analog outputs of the optional board, the values of any register (X, K, C or M) at depth 0 present in the controller. To do so, common parameters **C30**, **C31** and **C32** must be used. The 4 depths of the common parameters correspond to the 4 analog outputs. The XSRT, XSRI and XSRA commands are an alias of these common parameters and use the same syntax.

C	Alias	Name	<P1>	Comment
C30	XSRT[:depth].<axis>=<P1>	Source Register Type	0 1 2 3 13 33 34 35 45 65 66 67 77	Immediate value defined by C7 Reference type is a user's variable X Reference type is a parameter K Reference type is a monitoring M Reference type is a C register Reference type is a user's variable XF Reference type is a parameter KF Reference type is a monitoring MF Reference type is a CF register Reference type is a user's variable XL Reference type is a parameter KL Reference type is a monitoring ML Reference type is a CL register
C31	XSRI[:depth].<axis>=<P1>	Source Register Index	0 - 512	Indicates the source register number. The 4 depths of the parameter correspond to the 4 analog outputs.
C32	XSRA[:depth].<axis>=<P1>	Source Register Axis	0 1	Indicates that the source register is taken from the first axis. Indicates that the source register is taken from the second axis The 4 depths of the parameter correspond to the 4 analog outputs.

The digital analog converter (DAC) may induce a small offset and amplitude error. These errors can be removed with the common parameters **C27** and **CF28**. The XAOO and XAOA commands are an alias of these common parameters and use the same syntax.

C	Alias	Name	Comment
C27	XAOO[:depth].<axis>=<P1>	Analog Output Offset	Modifies the offset of the signal sent to the analog outputs (XAOOUT1 to XAOOUT4). The 4 depths correspond to the 4 analog outputs.
CF28	XAOA[:depth].<axis>=<P1>	Analog Output gAin	Modifies the amplitude of the signal sent to the analog outputs (XAOOUT1 to XAOOUT4). The 4 depths correspond to the 4 analog outputs.

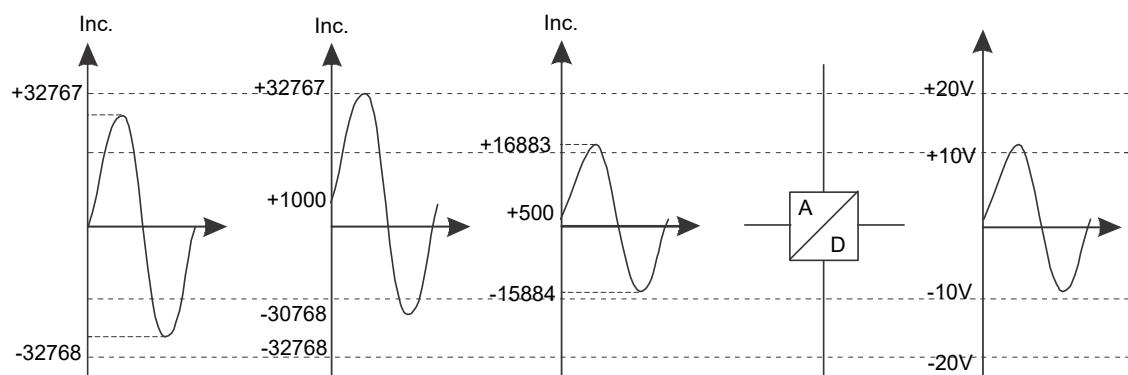
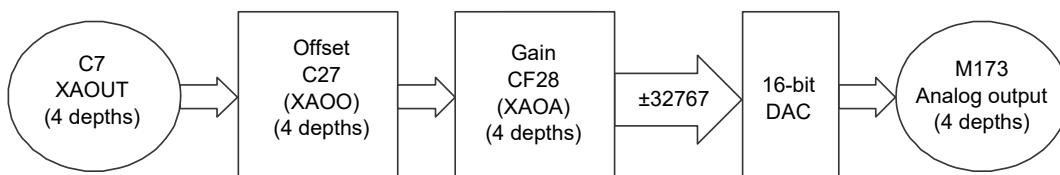
The monitorings **M173** allows the user to read the value of the 4 analog outputs by taking the common parameters C27 and CF28 into account:

M	Name	Comment
M173	Analog output status	Gives the value on the analog outputs. The 4 depths correspond to the 4 analog outputs.

Here is the formula giving the values on the analog outputs after the amplitude and offset correction:

$$M173 = ((\text{Result of C30, C31 and C32}) + C27) \cdot CF28$$

Example with C27=1000, CF28=0.5F.



8.6.2 On AccurET VHP controllers

8.6.2.1 Analog inputs

It is possible to use the 4 analog inputs (AIN). The management for these 4 AIN is common for the controller. M53 with 4 depths showing the AIN before the treatment (offset and gain) and MF51 with 4 depths showing the AIN after the treatment (offset and gain).

M	Alias	Name	Comment
MF51	AIN	Analog input after gain and offset compensation	Gives the analog input state after gain and offset compensation.
M53	-	Analog input before gain and offset compensation	Gives the analog input state before gain and offset compensation.

CF117 with 4 depths used for the offset correction and CF118 with 4 depths used for the gain correction. Refer to [§8.2](#) for more information.

C	Name	Comment
CF117	Analog input offset	Modifies the offset of the analog input signal. The 4 depths correspond to the 4 analog inputs.
CF118	Analog input gain	Modifies the gain of the analog input signal. The 4 depths correspond to the 4 analog inputs.

Here is the formula giving the values on the analog inputs after the amplitude and offset correction:

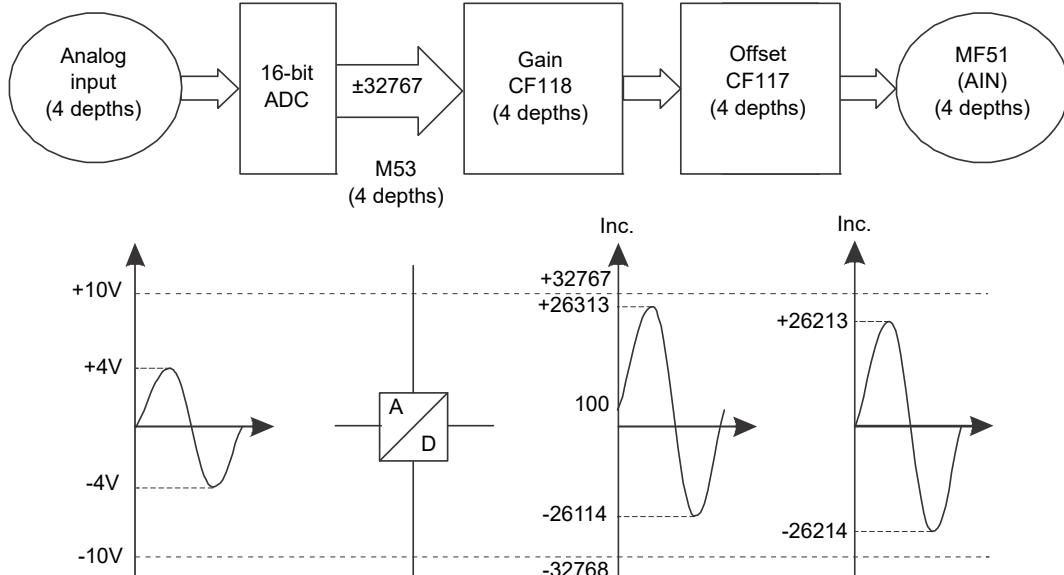
$$MF51.<\text{axis}> = (M53.<\text{axis}> \times CF118.<\text{axis}>) + CF117.<\text{axis}>$$

The input voltage of the analog input must be included between +10 V and -10 V between AIN+ and AIN-. +10V corresponds to +32767 increments and -10 V to -32768 increments.

Here is the formula calculating the voltage from the value contained in the monitoring MF51 and M53:

$$U[V] = \left(MF51 \text{ or } \frac{M53}{32768} \right) \times 10$$

Example with CF118=2.0F and CF117=-100.0F



8.6.2.2 Analog outputs

It is possible to use 4 analog outputs (AOUT). The management for these 4 AOUT is common for the controller.

The monitoring M174 with 4 depths shows the value given on the analog outputs

M	Name	Comment
M174	Real state of analog outputs	Gives the real state of the analog outputs

- 6 new common parameters are available (refer to [S8.2](#) for more information):
 - Parameter **CF107** with 4 depths for an immediate value
 - Parameter **CF127** with 4 depths for the offset correction
 - Parameter **CF128** with 4 depths for the gain correction
 - Parameter **C130** with 4 depths for the register source type
 - Parameter **C131** with 4 depths for the register source type index
 - Parameter **C132** with 4 depths for the register source type axis number

C	Alias	Name	Comment
CF107	AOUT	Analog output	Used to set the analog outputs value ($32767.0=20\text{ V}$ between AOUT+ and AOUT-, $-32768.0=-20\text{ V}$ between AOUT+ and AOUT-)
CF127	AOO	Analog outputs offset	Modifies the offset of the analog output signal. The 4 depths correspond to the 4 analog outputs.
CF128	AOA	Analog outputs gAin	Modifies the amplitude of the analog output signal. The 4 depths correspond to the 4 analog outputs.
C130	SRT	Source register type	Used to set the source register type
C131	SRI	Source register index	Used to set the source register index
C132	SRA	Source register axis	Used to set the source register axis. Choose the axis to be linked with the monitoring source

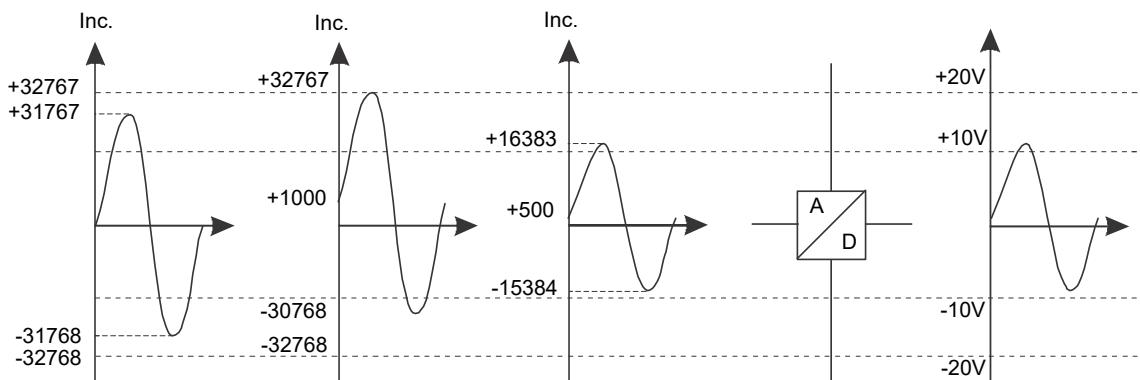
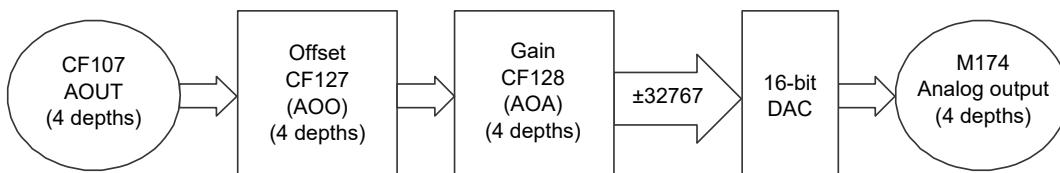
Here is the formula giving the values on the analog outputs after the amplitude and offset correction in 'direct' mode (this mode is available when C130 value is equal to 0):

$$\text{M174} = (\text{CF107} \times \text{CF127}) \times \text{CF128}$$

Here is the formula giving the values on the analog outputs after the amplitude and offset correction in 'source register' mode (this mode is available when C130 value different than 0):

$$\text{M174} = ((\text{Result of C130, C131 and C132}) + \text{CF127}) \times \text{CF128}$$

Example with CF127.0F=1000, CF128=0.5F



8.7 Position capture on digital inputs

The **POSCAPT** command allows a position capture on different position feedback.

Command	<P>	Comments
POSCAPT.<axis>=<P1>,<P2>	<P1> <P2>	Active or stop the position capture. 1: active one capture (same as K182=1) and 0: inactive Depth of the position feedback (bit0: main, bit1: secondary, bit2: auxiliary)

The parameter **K182** allows also the user to activate the position capture on the digital Inputs. However, this method is deprecated and should not be used.

K	Name	Comment
K182	Position capture on input (DIN & FDIN)	Enables position capture on input according to parameter K301. 1: enables the capture and 0 stops the process.

Remark: The parameter K182 is automatically equal to 0 when the controller is switched on. This parameter cannot be saved in the controller. Writing 1 in the parameter K182 starts the position capture process even if the value was already equal to 1. To start a new position capture, the value 1 must be rewritten in the parameter K182. This parameter K182 will remain at the set value even after the position capture.

Two consecutive position capture can be saved. The time resolution of the capture is 10 ns and the position is linearly interpolated between previous and current position (interval of PLTI=50 µs). When one of the conditions defined by the parameter **K301** is true, the position is captured and saved.

K	Name	Comment
K301	Input edge mask	Defines the mask of the digital inputs (rising & falling edge) for position capture. bit#0-15: rising edge. bit#16-31: falling edge. Depth 0: DIN, depth 1: FDIN.

K301		bit#0	bit#1	bit#2	bit#3	bit#4	bit#5	bit#16	bit#17	bit#18	bit#19	bit#20	bit#21
Depth 0: DIN	AccurET 400 / 600	DIN1	DIN2	DIN3	DIN9	DIN10	-	-	DIN1	DIN2	DIN3	DIN9	DIN10	-	-
	AccurET 48 / 300 VHP 48 / 100	DIN1	DIN2	-	DIN9	DIN10	-	-	DIN1	DIN2	-	DIN9	DIN10	-	-
Depth 1: FDIN	AccurET 400 / 600	FDIN1	FDIN2	FDIN3	FDIN4	-	-	-	FDIN1	FDIN2	FDIN3	FDIN4	-	-	-
	AccurET 48 / 300 HP 48 / 100	FDIN1	FDIN2	FDIN3	FDIN4	FDIN5	FDIN6	-	FDIN1	FDIN2	FDIN3	FDIN4	FDIN5	FDIN6	-
		Rising						Falling							

Here are the different monitorings allowing the user to manage the position capture.

M	Name	Comment
M58	Status position capture	Gives the status of the position capture. bit#0: capture 1 activated, bit#1: capture 2 activated, bit#2: capture 1 done, bit#3: capture 2 done, bit#4: capture 1 in process, bit#5: capture 2 in process, bit#6: capture on main encoder, bit#7: capture on secondary encoder, bit#8: capture on auxiliary encoder, bit#9: single capture, bit#10: multiple capture
M160	Captured position 1 conditions	Gives the conditions of the captured position 1. Depth 0 gives the first capture and depth 1 gives the second capture. Bit#0-4: DIN rising condition, bit#5-10: FDIN rising condition, bit#11-15: DIN falling condition, bit#16-21: FDIN falling conditions
M161	Captured position 2 conditions	Gives the conditions of the captured position 2 at depth 0. Bit#0-4: DIN rising condition, bit#5-10: FDIN rising condition, bit#11-15: DIN falling condition, bit#16-21: FDIN falling conditions. M161 is kept for compatibility reason because the second depth of M160 can be used to do the same thing.
ML165	Captured position 1 on main encoder	Gives the captured position 1 on main encoder. Depth 0 gives the first capture and depth 1 gives the second capture. Takes SET command and mapping correction into account. The ISO conversion is [upi] and is only related to the main encoder.
ML166	Captured position 2 on main encoder	Gives the captured position 2 on main encoder at depth 0. Takes SET command and mapping correction into account. ML166 is kept for compatibility reason because the second depth of ML165 can be used to do the same thing. The ISO conversion is [upi] and is only related to the main encoder.

M	Name	Comment												
ML167	Captured position on secondary encoder	Gives the captured position on secondary encoder. Depth 0 gives the first capture and depth 1 gives the second capture. Takes care of a offset compensation given by the command OFFSETSEC (ML33). The ISO conversion for this monitoring is [dpi2] and is only related to the secondary encoder.												
ML168	Captured position on auxiliary encoder	Gives the captured position on auxiliary encoder. Depth 0 gives the first capture and depth 1 gives the second capture. Takes care of a offset compensation given by the command OFFSETAUX (ML35) The ISO conversion for this monitoring is [dpi3] and is only related to the auxiliary encoder.												

M160 / M161 for AccurET400 / 600	bit#0	bit#4	bit#5	bit#8	bit#9 & 10	bit#11	bit#15	bit#16	bit#19	bit#20 & 21
	DIN1	DIN10	FDIN1	FDIN4	Not used	DIN1	DIN10	FDIN1	FDIN4	Not used
	Rising				Rising				Falling				Falling	

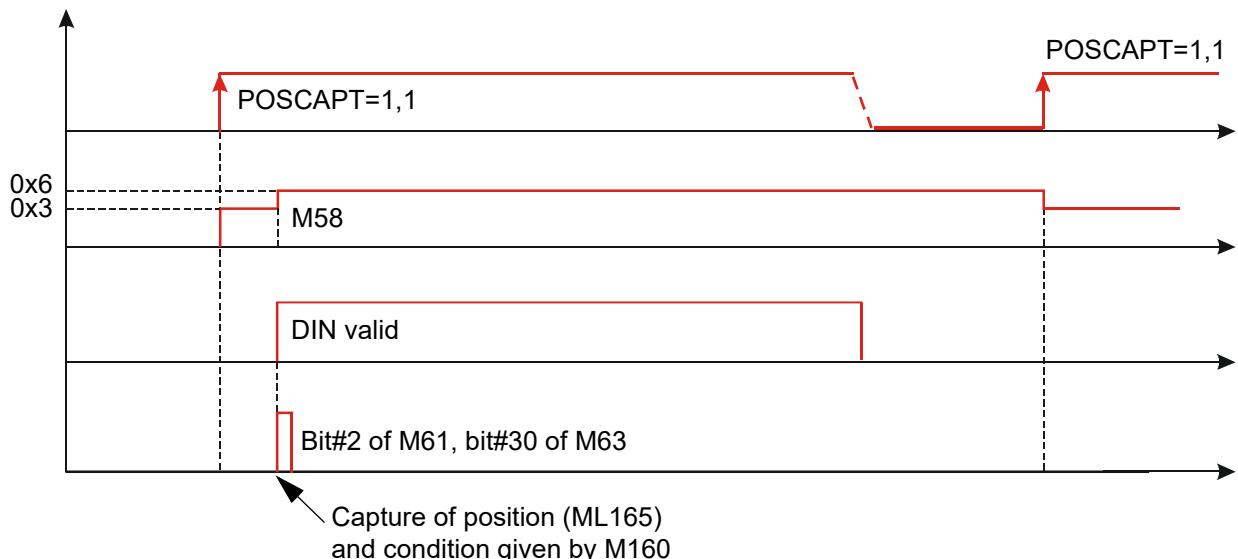
M160 / M161 for AccurET 48 / 300	bit#0	bit#4	bit#5	bit#10	bit#11	bit#15	bit#16	bit#21		
	DIN1	DIN10	FDIN1	FDIN6	DIN1	DIN10	FDIN1	FDIN6		
	Rising				Rising				Falling				Falling	

Remark: These monitorings are updated at the beginning of the MLTI only.

Bit#2 of the monitoring M61 is set for **one MLTI only** when a capture event occurs (same as bit# 30 of monitoring M63).

8.7.1 Description of the position capture process on the digital inputs

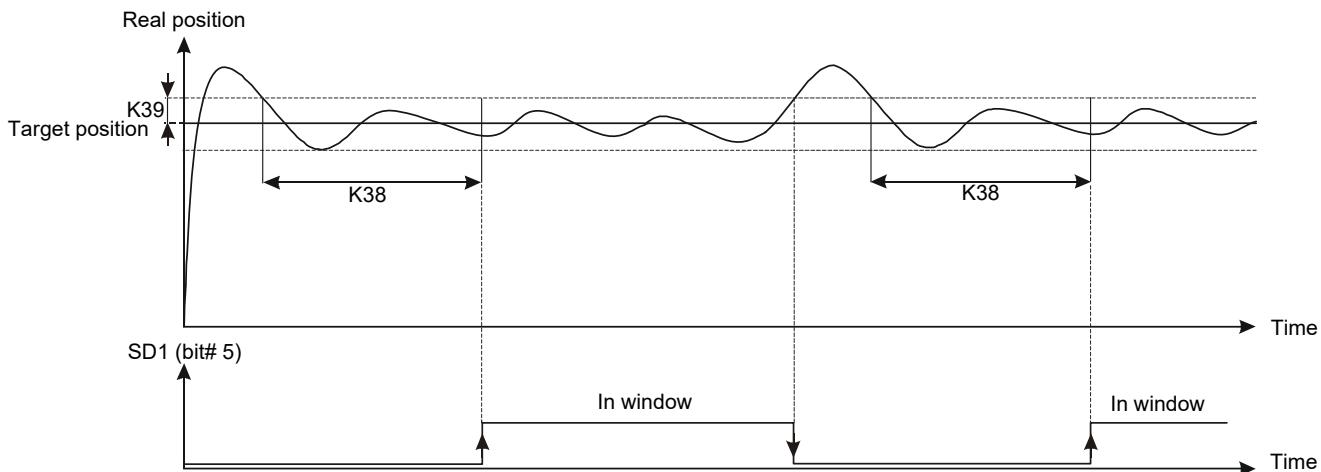
- Set the bits of the parameter K301 corresponding to the digital inputs which must be detected on a falling or rising edge (for example: K301:1.0=0x20001).
- To start a new position capture on the main encoder, the following three points must be re-executed.
 - Set the command POSCAPT=1,1 to execute a new position capture (M58 becomes: M58=0x3)
 - The position controller monitors the digital inputs according to the parameter K301. When one of the conditions is met, the position controller stores the position (upi) in the monitoring ML165/ML166, sets the condition of the trigger in the monitorings M160/M161 and sets the bit# 2 of monitoring M61 and the bit# 30 of monitoring M63 for one MLTI (400µs).
 - Up to two captures will be saved (first one to ML165/M160 and second one to ML166/M161).



8.8 In-window

It is possible to define a 'window' around the target position to reach when the standard reference mode is selected (K61=1) but not in interpolation mode (ITP). The controller scans the real position of the motor when the position is inside the 'window' and activates a status bit#SD1 (bit# 5 of monitoring M60). The user has the possibility to stop the programmed sequence until the motor arrives into the 'window' with the **WTW** command (refer to [S7.10.3](#) for more information about the WTW command).

The motor is considered to be positioned when its real position is between [target position - K39] and [target position + K39] during a minimum time given by the parameter K38 (in [MLTI]) without interruption. The **TIMEW** (**TIME Window**) and **POSW** (**POSition Window**) commands are alias of the parameters K38 and K39 respectively and use the same syntax.



K	Alias	Name	Comment	Units
K38	TIMEW	In-window time	Minimum time for the real position to stay in the position window	[MLTI]
K39	POSW	In-window position	Maximum (absolute) position error for the real position to stay in the position window	[upi], [rupi]

This **In-window bit** (bit#5 of monitoring M60 (SD1)) is set to 1 when the motor reaches the position window and stays within the tolerance during the time defined by the parameter K38. This bit is set to 0 during the next movement (with MVE, STA, STI or WTW.<axis>=2 command).

Remark: When K38=0, the In-window bit (bit#5 of monitoring M60) is set to 1 at the end of the theoretical trajectory whatever the value of the parameter K39.

When K39=0, the In-window bit (bit#5 of monitoring M60) is set to 1 at the end of the theoretical trajectory plus the value of the parameter K38.

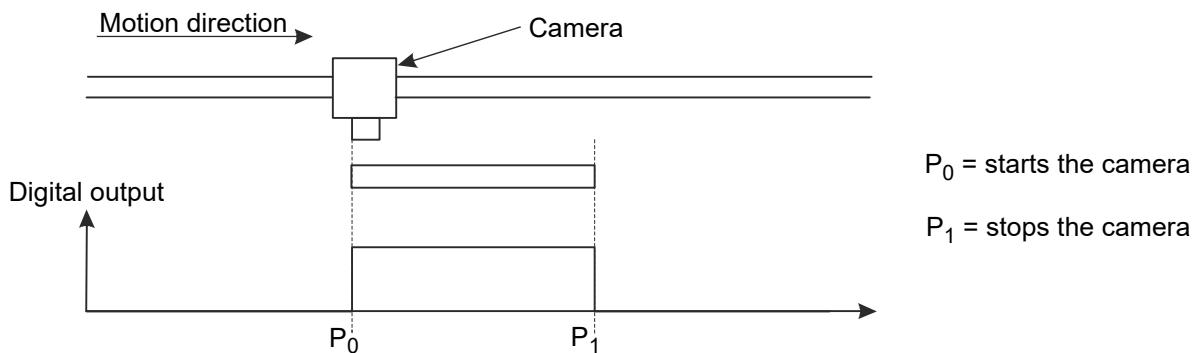
The monitoring **M38** allows the user to know the time between the WTW command and the setup of the bit# 21 of M63.

M	Name	Comment
M38	In-window time	Gives the time between the WTW command and the setup of the bit#5 of M60 (motor in the windows defined by K38 and K39)

Restrictions: The offset compensation (refer to [S8.2.1](#)) are not taken into account on the target position used by the WTW command.

8.9 Fast Triggers

ETEL's Fast Triggers feature "trigger" an "action" when reaching or crossing pre-set "position thresholds". It gives to the user a lot of freedom to program because it provides a very flexible configuration.



- "Position threshold" can be placed on 1D or 2D positions. (e.g. X, Theta or XY). It can be both positive and negative crossing of motion on the threshold. To find out all possible options, refer to General parameters ([§8.9.2](#)), 1D Fast Triggers ([§8.9.3](#)), 2D Fast Triggers ([§8.9.4](#)), Fast Triggers status ([§8.9.5](#)) or Additional feature ([§8.9.6](#)).
- "Action" is configured by user. For e.g., an event can be configured to set digital outputs, to start a pulse train or to set user bits. To find out all possible options, refer to Action induced by an event ([§8.9.2.3](#)).
- "Position threshold" and "action" must be programmed together into an event register (EL register) of the AccurET axis. Each AccurET axis has an EL register table. A single EL register table has a maximum of 512 EL registers available for the user. Refer to [§8.9.2.2](#) for more information.

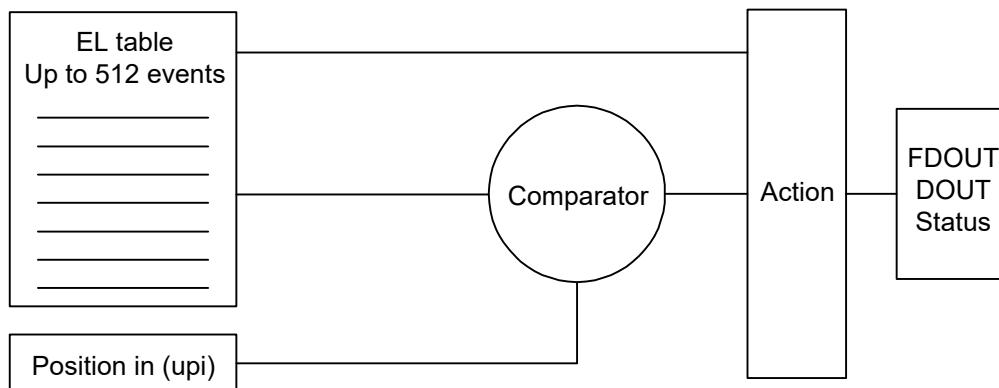
When the 2D Fast Triggers feature are used, the two EL register tables from two axes are required. Refer to [§8.9.4.3](#) for more information. Moreover, ETEL's Fast Triggers support advanced users with its unique features. They are optional features and can be combined with 1D or 2D Fast Triggers at user's discretion:

- When real position of motion trajectory is close, but misses the target position threshold, "Missed event" detection feature can detect and notify the user of such mishap. Refer to [§8.9.6.3.6](#) for more information.
- Position source for 1D or 2D Fast Triggers is generally the same axis where the EL register table is being downloaded. In addition, "Position source coming from any register" feature can expand the possible source of position triggering. Refer to [§8.9.6.3](#) for more information.
- With correct and careful configuration, the "Continuous Fast Triggers" feature allows user to setup virtually infinite number of Fast Triggers events. Refer to [§8.9.6.4](#) for more information.

In order to help the user better understand how to use ETEL's Fast Triggers feature, the How-to section provides examples from the simplest to the most advanced ones. The example sequence codes and pictures can be found in [§8.9.9](#).

8.9.1 Overview

Fast Triggers are programmed in tabular form. Every line represents an event describing the action induced by the controller when a defined position [upi] is reached. Once reached and executed, the Fast Triggers function moves to the next line in the table.



Remark: From firmware $\geq 3.10A$, the number of events contained in an EL table has increased to 512. These events can be stored in flash memory, however if downgrading to a firmware $< 3.10A$, the EL table will be restored to zero (default value). The user must then download the EL table (max 256 events) and save it again.

8.9.2 General parameters

8.9.2.1 Position type selection

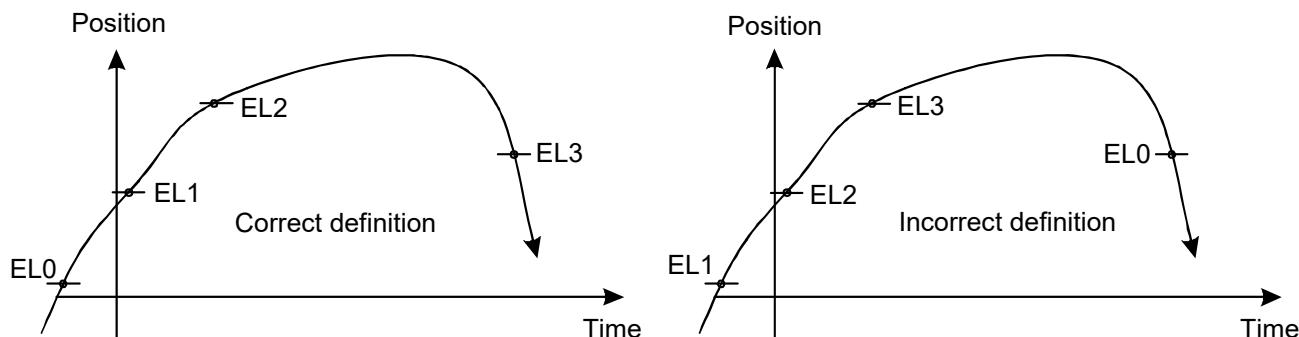
The parameter **K336** allows the user to choose if the fast output Triggers is based on real or theoretical position:

K	Name	Value	Comment
K336	Fast Triggers position type	0	Fast Triggers based on the real position (ML7) and real speed after scale and Stage Mapping (default value).
		1	Fast Triggers based on the theoretical position (ML6) and theoretical speed (M10).

Remark: It is also possible to get a position from an external register (refer to [§8.9.6.3](#) for more information).

8.9.2.2 EL registers

There is a maximum of 512 events per axis from EL0 to EL511. The events must be introduced in chronological order in the EL table according to their occurrence when executing the movement.



Remark: When an event is reached, the next event is the following in the table.
With firmware $< 3.10A$, there are only 256 events available.

The EL register has 6 depths of 64 bits. If one depth is not entered, its value will be equal to the previous one.

- **Target: position definition (depth0)**

Target defines the position to be reached or to be crossed (where the action will start from). This position is defined on 48 bits.

Remark: The position is defined in [upi] and will be compared to the position specified by K336.

The position must be entered in increment (inc.). ISO conversion is possible through ComET or EDI. However, check the conversion first to be sure to get an integer value.

- Type: Fast Triggers type (depth1)

Fast Triggers type defines the condition to reach the target:

Bits	Name	Comment
0 & 1	Threshold crossing (for corresponding axis)	0 = could be either positive or negative 1 = positive only 2 = negative only 3 = axis threshold is not taken into account

Remark: In case of an incremental mode, the delta position (ΔP) must be positive if the type of Fast Triggers is equal to 1 and negative if the type of Fast Triggers is equal to 2.

In case of 2D Fast Triggers there are more values available (refer to [§8.9.4.3.2](#) for more information).

- Combi: combination number (depth2)

Combi defines the combination number to be used by the event.

Combi 0 to 15: Available for standard mode Fast Triggers. Fast Triggers action (refer to [§8.9.2.3](#)) can be defined for each combi.

Combi 16: Reserved for incremental mode Fast Triggers. Fast Triggers action can be either Pulse Generator 1 or 2 (refer to [§8.9.3](#) in incremental mode for more information).

- PG: pulse generator (depth3)

PG defines if the selected pulse generator (1 or 2) is started or stopped. The different values are:

0 = No pulse generator is selected
1 = Starts the pulse generator 1
2 = Stops the pulse generator 1
3 = Starts the pulse generator 2
4 = Stops the pulse generator 2

To configure Pulse generation please refer to [§8.9.2.3](#).

- ΔP : delta position (depth4)

This parameter is only valid when using 1D Fast Triggers in incremental mode.

The delta position (ΔP) determines the position between two thresholds. The delta position is available only for the combination 16 and is defined on 48 bits.

EL0 defines the starting position and EL1 the stop position of the Fast Triggers (refer to [§8.9.3](#) in incremental mode for more information).

Remark: The position is defined in [upi] and corresponds to the reference or real position depending on the value of parameter K336.

- Not defined (depth5)

This parameter is not defined, but if you want to download EL registers through stream you need to specify it to 0.

8.9.2.3 Action induced by an event

8.9.2.3.1 Set/Reset/Toggle

A table of 16 combinations allow the user to specify a Set/Reset/Toggle on output (FDOUT/DOUT) or a status change.

K	Name	Comment
K320- K335	Fast Triggers combination	Fast Triggers combination X depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group (M351), depth 7: mask for users status

Remark: Refer to the corresponding “Hardware manual” to know the number of DOUT or FDOUT present.

K320 corresponds to combi1,... and K335 corresponds to combi16.

The depth 7 allows a modification of the status mask (bits#8-15 of monitoring M61 ('Status Drive' SD2) and bits#0-15 of monitoring M63). Refer to [§19](#). for more information about monitoring M61 and M63.

For all these masks, bits 0 to 15 are used to SET and bits 16 to 31 are used to RESET. If the same bit is used in the SET and RESET mask, the function is TOGGLE (TOGGLE function is only available on DOUT and FDOUT).

The depth 3 is used to reset the counter of group event when the event with a combination where depth3 is set to 1 is fired.

Example:

The user wants to set FDOUT1, to reset FDOUT2 and toggle FDOUT4

K320:1= | 0000 0000 0000 1010 | 0000 0000 0000 1001 |
 | Outputs to reset Outputs to set |

Remark: Only one event can be activated at the same time per motor and each event includes one of the 16 different combinations for a standard mode and only one for the incremental mode.

8.9.2.3.2 Pulse generator

There are 2 Pulse Generators (PG) available per controller (not per axis) named PG1 and PG2. If PG is configured for one axis, it is not allowed to configure it on the second axis (refer to [§8.9.7.1](#) for more information). These 2 pulse generators can be set with:

- Fix timing: the pulse is set with pre-programmed fixed duration of delay, pulse width and interval pulse.
- Modulated timing: the width or the period between 2 rising edges of pulses can be modulated from any source register (only available with firmware ≥ 3.10A).

The monitoring M345 allows the user to know the number of pulses already sent by the pulse generator PG1 or PG2.

M	Name	Comment
M345	Pulse count	Gives the PG pulse count (depth 0: PG1; depth 1: PG2)

This monitoring shows the number of pulses already sent, compared to the parameters K345. It is not linked to the incremental mode. Once all pulses generated, the monitoring M345 is automatically reset to 0. As the monitoring is updated at the frequency of the MLTI and depending to the pulses times and number, this monitoring can remain all the time at 0.

- **Fixed time**

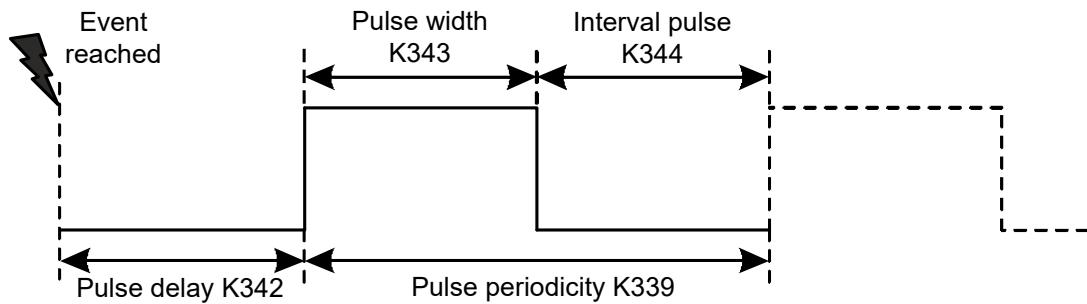
Here is the list of the parameters allowing the setting of pulse generators with fixed time.

K	Name	Comment
K339	Pulse Generator pulse periodicity	Depth 0: Pulse Generator 1 Depth 1: Pulse Generator 2
K340	Pulse Generator1 output mask	Depth 0: DOUT (bit#0 to 1) Depth 1: FDOUT (bit#0 to 3)

K	Name	Comment
K341	Pulse Generator2 output mask	Depth 0: DOUT (bit#0 to 1) Depth 1: FDOUT (bit#0 to 3)
K342	Pulse Generator delay	Depth 0: Pulse Generator 1 Depth 1: Pulse Generator 2)
K343	Pulse Generator pulse width	Depth 0: Pulse Generator 1 Depth 1: Pulse Generator 2
K344	Pulse Generator interval pulse	Depth 0: Pulse Generator 1 Depth 1: Pulse Generator 2
K345	Pulse Generator pulse count	Depth 0: Pulse Generator 1 Depth 1: Pulse Generator 2

Changes from the firmware $\geq 3.10A$:

- It is possible to change at any time all the registers K339, K342, K343 and K344. It means that it is possible to change for instance the pulse width while the PG is already outputting a train of pulses.
- It is possible to set the registers K342, K343 and K344 to 0. With firmware $< 3.10A$, the minimum value was 2, which meant 20ns minimum. From firmware $\geq 3.10A$, a '0' in these registers means that the register is not taken into account, allowing the KF registers to be taken into account.
- There are more protections to avoid bad parameters setting, especially with PG (refer to [§8.9.7.1](#) for more information).



Example:

- **The Pulse Generator 1 (PG1)** is configured by setting the following registers:
 DOUT K340:0.<axis> = mask of DOUT
 FDOUTK340:1.<axis> = mask of FDOUT
 Delay (*) K342:0.<axis> = time after the event before starting the pulse(s)
 Pulse width (*) K343:0.<axis> = time of the pulse
 Interval pulse (*) K344:0.<axis> = time between pulses
 Nbr pulse K345:0.<axis> = number of pulse(s) (0 corresponds to an infinite number)
 Periodicity K339:0.<axis> = pulse periodicity
- **The Pulse Generator 2 (PG2)** is configured by setting the following registers:
 DOUT K341:0.<axis> = mask of DOUT
 FDOUTK341:1.<axis> = mask of FDOUT
 Delay (*) K342:1.<axis> = time after the event before starting the pulse(s)
 Pulse width (*) K343:1.<axis> = time of the pulse
 Interval pulse (*) K344:1.<axis> = time between pulses
 Nbr pulse K345:1.<axis> = number of pulse(s) (0 corresponds to an infinite number)
 Periodicity K339:1.<axis> = pulse periodicity

(*): The delay, the pulse width (minimum 20 ns) as well as the pulse interval (minimum 20 ns) are a multiple of 10 ns (max. frequency 25MHz).

The setting of the PG Fixed Time can be done by writing all registers directly, or in order to make faster the setting, by using the **CFGPGFIX** command.

Command	<P>	Comments
CFGPGFIX.<axis> = <P1>, <P2>,<P3>,<P4>,<P5>, <P6>, <P7>,<P8>	<P1> <P2> <P3> <P4> <P5> <P6> <P7> <P8>	Define Trigger Pulse Generator (1 or 2) Trigger: Pulse Generator mask DOUT(K340 depth 0) or (K341 depth 0) Trigger: Pulse Generator mask FDOUT(K340 depth 1) or (K341 depth 1) Trigger: Pulse Generator pulse periodicity(K339) Trigger: Pulse Generator delay(K342) Trigger: Pulse Generator pulse width(K343) Trigger: Pulse Generator interval pulse(K344) Trigger: Pulse Generator pulse count(K345)

To reset the configuration of the Fixed Pulse Generator, the command must be used as follow:

CFGPGFIX.<axis> = <P1>,0

As PGs are common for both AccurET axes, sending the CFGPGFIX command on one axis may have consequence on the other one. E.g. if the command is sent to axis 0, the following changes occur:

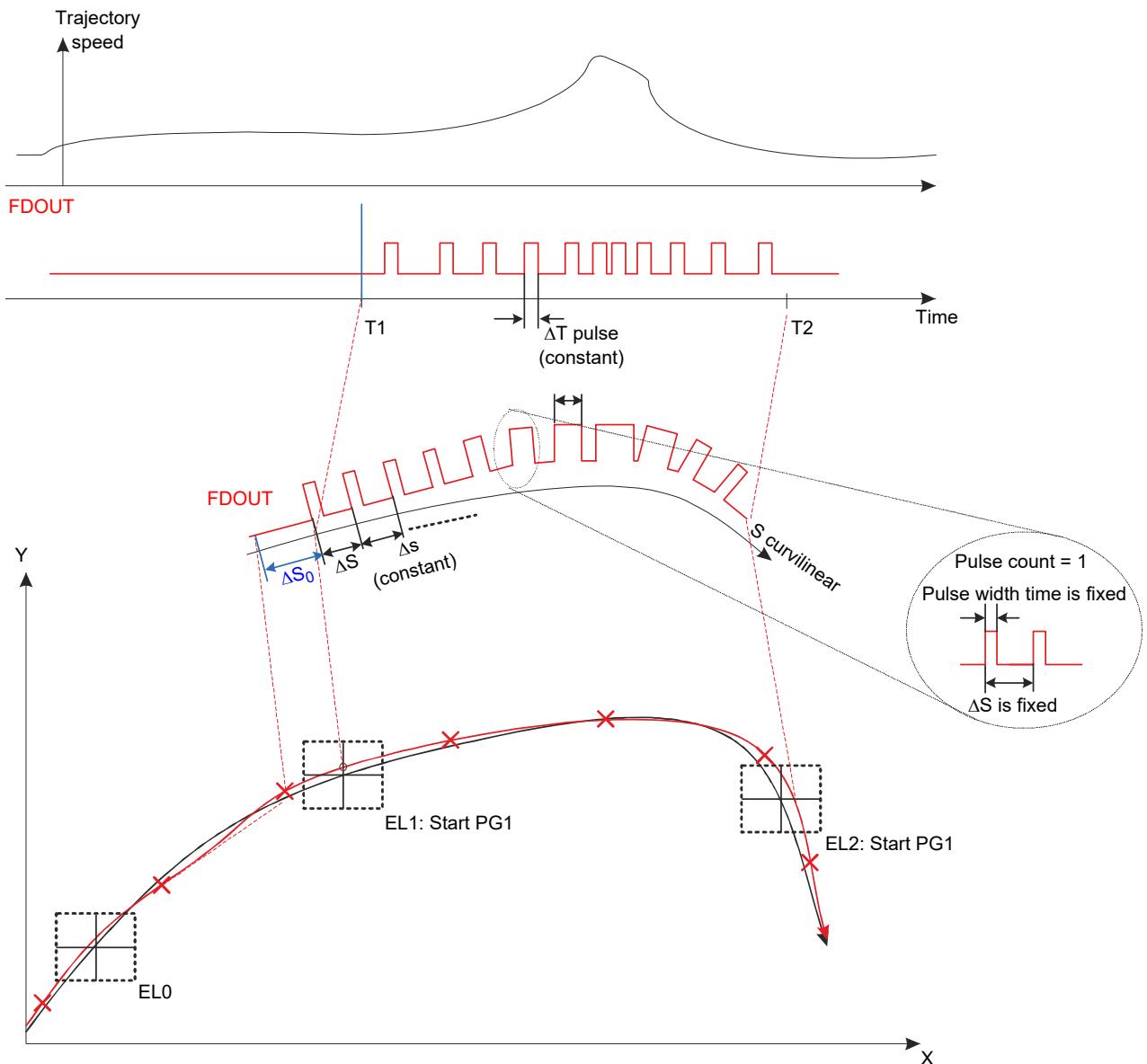
- All registers regarding the PG Fixed Time on axis 1 are cleared
- All registers regarding the PG Modulated Time on axis 0 are cleared (but not the ones from axis 1, as they have no influence)

Therefore, it is not required to clear the PG before using it with another axis.

Remark: An error is raised if the number of parameter is different than 2 or 8.
 An error is raised if a parameter is not an INT32.
 An error is raised if <P1> is not equal to 1 or 2.

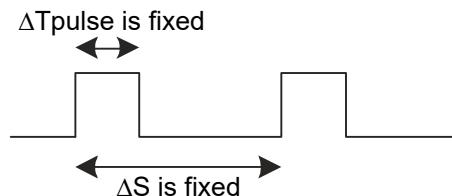
• **Modulated timing**

The aim of this feature is to generate a pulse every fixed distance over a curvilinear axis. It means the time between 2 rising edges of pulses depends on the speed. A typical use case with 2D Fast Triggers is to generate a pulse every fixed distance (ΔS) (Incremental mode is not allowed).

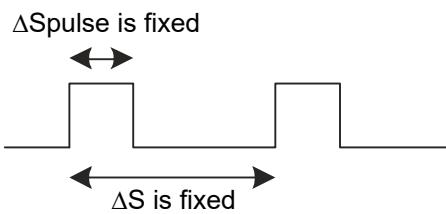


A curvilinear speed (calculated by UltimET *M/*MF680) is given through the TransnET. AccurET will use this speed from TransnET through register M450...M465 as the source of modulation for the period of PG. There are two options:

- Only the period between 2 rising edges of the pulses is modulated by the speed, in order to have a fixed distance (ΔS) and a fixed pulse time duration (ΔT_{pulse}).



- Both periods between 2 rising edges of pulse and the pulses duration are modulated by the speed, in order to have a fixed distance (ΔS) and a fixed distance (ΔS_{pulse})



Here is the list of the parameters allowing the setting of modulated time delays, pulse width and interval:

K	Name	Comment
KF339 (*)	Fast Triggers Pulse Generator pulse period modulation coefficient	Pulse period modulation coefficient (period is the time between two rising edges of pulses). This parameter can be used only if K339 and K344 are equal to 0 (depth 0: Fast Triggers Pulse Generator 1; depth 1: Fast Triggers Pulse Generator 2)
KF342 (*)	Fast Triggers Pulse Generator delay modulation coefficient	Delay modulation coefficient. This parameter can be used only if K342 is equal to 0 (depth 0: Fast Triggers Pulse Generator 1; depth 1: Fast Triggers Pulse Generator 2)
KF343 (*)	Fast Triggers Pulse Generator pulse width modulation coefficient	Pulse width modulation coefficient. This parameter can be used only if K343 is equal to 0 (depth 0: Fast Triggers Pulse Generator 1; depth 1: Fast Triggers Pulse Generator 2)
K353 (*)	Fast Triggers external register reference type	Fast Triggers external register reference type (depth 0: Fast Triggers Pulse Generator 1 (only M, MF, X and XF are accepted); depth 1: Fast Triggers Pulse Generator 2 (only M, MF, X and XF are accepted). Value 1 for X, value 3 for M, value 33 for XF and value 35 for MF are accepted.
K354 (*)	Fast Triggers external register reference index	Fast Triggers external register reference index (depth 0: Fast Triggers Pulse Generator 1; depth 1: Fast Triggers Pulse Generator 2)
K355 (*)	Fast Triggers external register reference depth	Fast Triggers external register reference depth (depth 0: Fast Triggers Pulse Generator 1; depth 1: Fast Triggers Pulse Generator 2)
K356 (*)	Fast Triggers external register reference axis	Fast Triggers external register reference axis (depth 0: Fast Triggers Pulse Generator 1; depth 1: Fast Triggers Pulse Generator 2)

(*): available from firmware ≥ 3.10A.

Remark: It is possible to change at any time all the KF339 to KF343 registers. It means that it will be possible to change for instance the pulse width while the PG is already outputting a train of pulses.

The setting of the PG Modulated Time can be done by writing all registers directly, or in order to make faster the setting, by using the **CFGPGMOD** command.

Command	<P>	Comments
CFGPGMOD.<axis> = <P1>, <P2>, <P3>, <PF4>, <P5>, <PF6>, <P7>, <PF8>, <P9>, <P10>	<P1> <P2> <P3> <PF4> <P5> <PF6> <P7> <PF8> <P9> <P10>	Define Trigger Pulse Generator (1 or 2) Trigger: Pulse Generator mask DOUT(K340 depth 0) or (K341 depth 0) Trigger: Pulse Generator mask FDOUT(K340 depth 1) or (K341 depth 1) Trigger: Pulse Generator pulse period modulation (KF339) Trigger: Pulse Generator pulse periodicity (K339) Trigger: Pulse Generator delay modulation coefficient (KF342) Trigger: Pulse Generator delay (K342) Trigger: Pulse Generator pulse width modulation coefficient (KF343) Trigger: Pulse Generator pulse width (K343) Trigger: Pulse Generator interval pulse (K344)

To reset the configuration of the Modulated Time Pulse Generator, the command must be used as follow:

CFGPGMOD.<axis> = <P1>,0

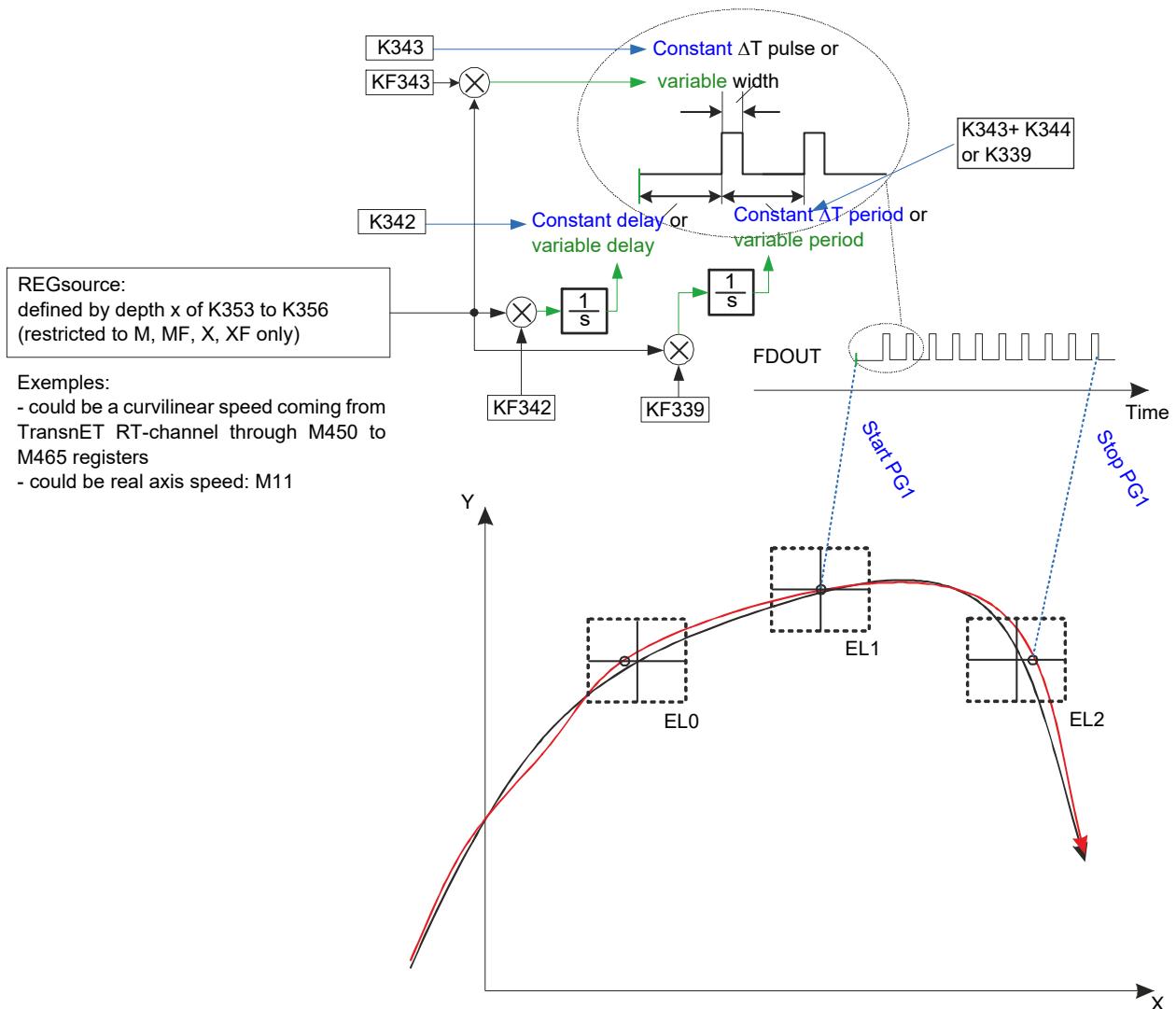
As PGs are common for both AccurET axes, sending the CFGPGMOD command on one axis may have consequence on the other one. E.g. if the command is sent to axis 0, the following changes occur:

- All registers regarding the PG Modulated Time on axis 1 are cleared
- All registers regarding the PG Fixed Time on axis 0 are cleared (but not the ones from axis 1, as they have no influence)

Therefore, it is not required to clear the PG before using it with another axis.

Remark: An error is raised if the number of parameter is different than 2 or 10.
An error is raised if all parameters are not in INT32, except <PF4> <PF6> and <PF8> which must be a FLT32.
An error is raised if <P1> is not equal to 1 or 2.

As an option to the registers K339, K342, K343 and K344, the registers KF339 to KF343 add the possibility to modulate the times (delay, pulse width and period) by an external source register. Here is the explanation including all possibilities:



Remark: In practice we have usually only one or two variables depending on the external source register (delay before first pulse, pulse width, or period between 2 rising edges of pulses), and the others are set to a fixed time with 342, K343, K344 or K339 (in that case these parameters must be different from zero and the corresponding KF must be set to zero).

To use a variable duration modulated by the REGsource, KF342, KF343 or KF339 parameters must be different of zero and the corresponding K must be set to zero.

Modulation coefficient registers: KF342, KF343 or KF339

These registers are defined as float 32-bit precision. They multiply the input REGsource to modulate the corresponding timing.

Min=0.0f // time modulation is OFF
Max= $3.4 \times 10^{38} f$

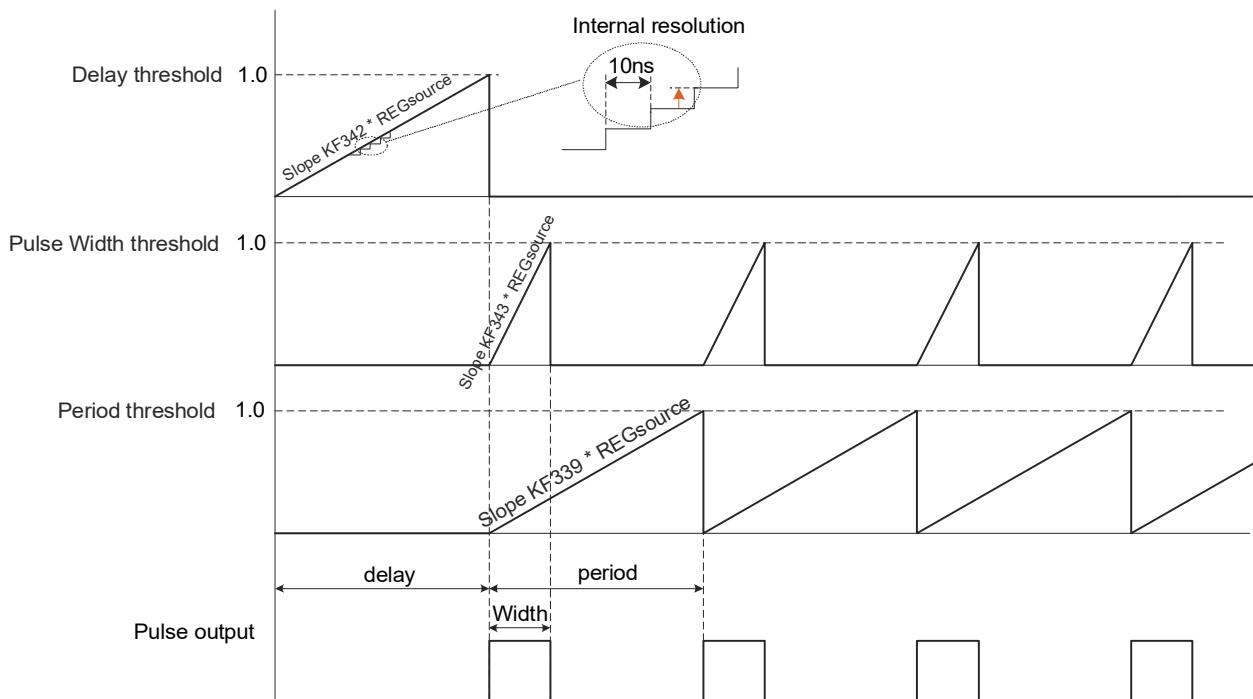
$$\text{Delay before first pulse [s]} = \frac{1}{KF342 \cdot REG_{Source}}$$

$$\text{Pulse width [s]} = \frac{1}{KF343 \cdot REG_{Source}}$$

$$\text{Period between rising edges [s]} = \frac{1}{KF339 \cdot REG_{Source}}$$

Here is an example showing a situation where all durations are modulated by an external source register and

multiplied by the three corresponding coefficients: KF339, KF342 and KF343. The 'REG_source * KF3xx' represents a slope. The fixed threshold of 1.0 is reached in a time that depends on the slope.



Remark: When using the modulated PG, the user should calculate the accuracy of the 'delay', 'pulse', and 'period' that he could expect. For instance, it is important in the case where the user wants to generate a fixed distance between two pulses using the curvilinear speed as the source of modulation. The maximum relative error is directly seen on the distance between pulses and can be calculated with the formula:

$$\text{Relative error} = \Delta T \cdot \frac{10^8}{2^{32}} = \frac{\Delta P}{V} \cdot \frac{10^8}{2^{32}}$$

Where: V is the real speed in [m/s] or [turn/s]
 ΔT is the period (time between two rising edges of pulse) in [s]
 ΔP is the distance between two rising edges of pulse in [m] or [turn]

Example: the user needs a pulse every 1mm and the motor is moving at 0.5m/s:

$$\text{Relative error} = \frac{0.001}{0.5} \cdot \frac{10^8}{2^{32}} = 46.6ppm$$

Here is an example of setting generated pulses every fixed distance with the following requirements:

- The curvilinear speed is given over the TransnET
- There is no initial delay (before the first pulse)
- The pulse width has a fixed duration of 5 μ s
- The distance between 2 rising edges of pulse is 5 μ m

Calculation example of KF339 to fit to the right distance. We assume that M450 contains a curvilinear speed in User Friendly Speed Increments of UltimET [UFSI] where *K524=-6 meaning that 1 USFI=1 μ m/s.

To have a fixed distance between 2 rising edges ($\Delta P = 5 \mu m$), we must then calculate:

$$KF339 = \frac{1}{\Delta P} \cdot \frac{1}{\text{speed_conv}}$$

Where speed_conv is the number of increment of speed in the source register for 1 m/s. In our example:

`speed_conv = 10-K524 = 106 UFSI / (m/s):`

$$KF339 = \frac{1}{5 \cdot 10^{-6}} \cdot \frac{1}{10^6} = 0.2f$$

8.9.2.4 Monitorings

Several monitorings allow the user to know how many events are reached.

M	Name	Comment
M346	Number of treated events	Gives the number of the treated events of the Fast Triggers
M351	Number of treated events per group	Same information as M346. However this counter could be reset when an event is reached with a combination K320-K335:3=1
ML340	Published position for Fast Triggers	Published position for Fast Triggers (upi)
M340	Published position for Fast Triggers	Published position for Fast Triggers (dpi). LSL part of ML340
M341	Published extrapolated step Fast Triggers (dpi)	Published extrapolated step Fast Triggers (dpi)

Remark: M346 and M351 are both limited to 2^{16} and are only reset when the Fast Triggers are re-enabled.
M340/ML340 or M341 are updated only when the Fast Triggers are enabled.
M340/ML340 could show a real or a theoretical position depending on K336. More details can be found in [§8.9.6.3](#).

8.9.2.5 Fast Triggers output mask

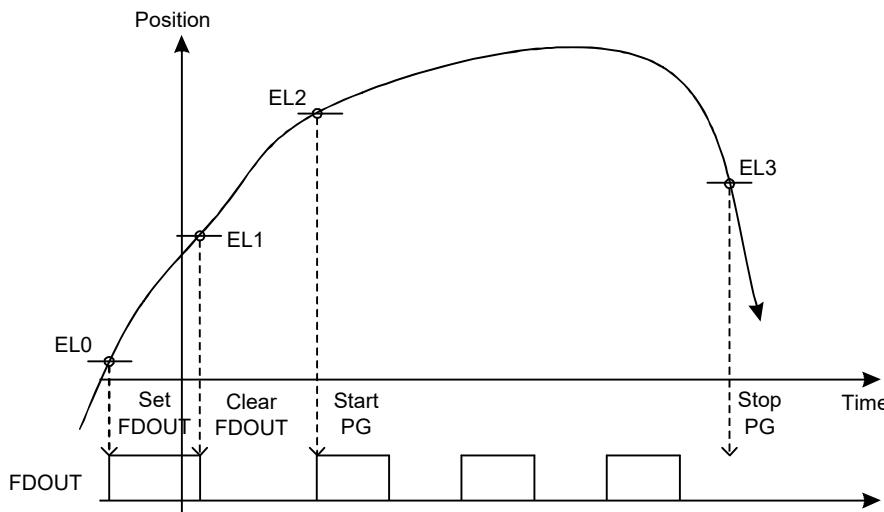
The parameters K359 and C359 are used to reserve the DOUT and/or FDOUT for the Fast Triggers feature (refer to [§8.5.3](#) for more information).

8.9.3 1D Fast Triggers

8.9.3.1 Description

The 1D Fast Triggers can be used in 2 modes:

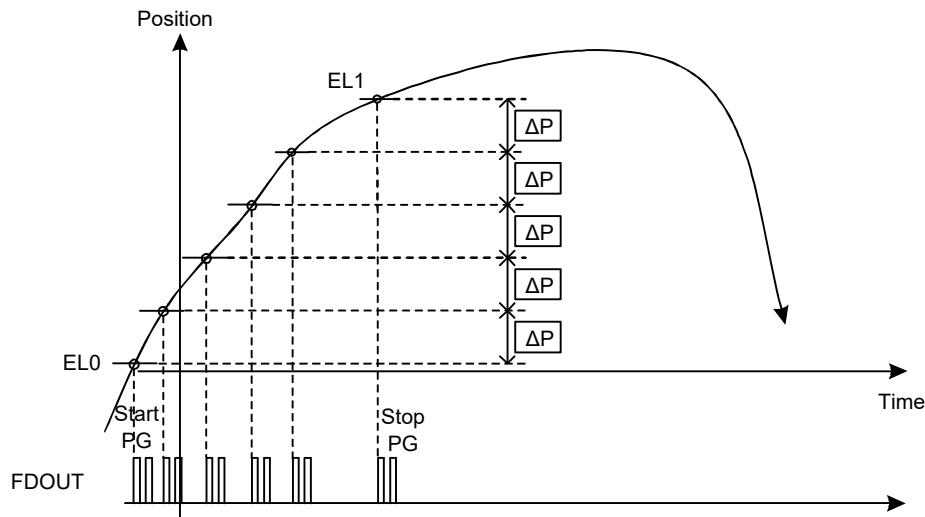
- **Standard mode:** One event correspond to one line in the EL table
 $EL0.<\text{axis}>=\text{Target, type, combi, PG, 0L, 0L}$



- **Incremental mode:** Events are defined with a start position, a Δ Position and a stop position. After each event reached, the next event to reach is obtained by adding the ΔP to the previous event until reaching

the last position.

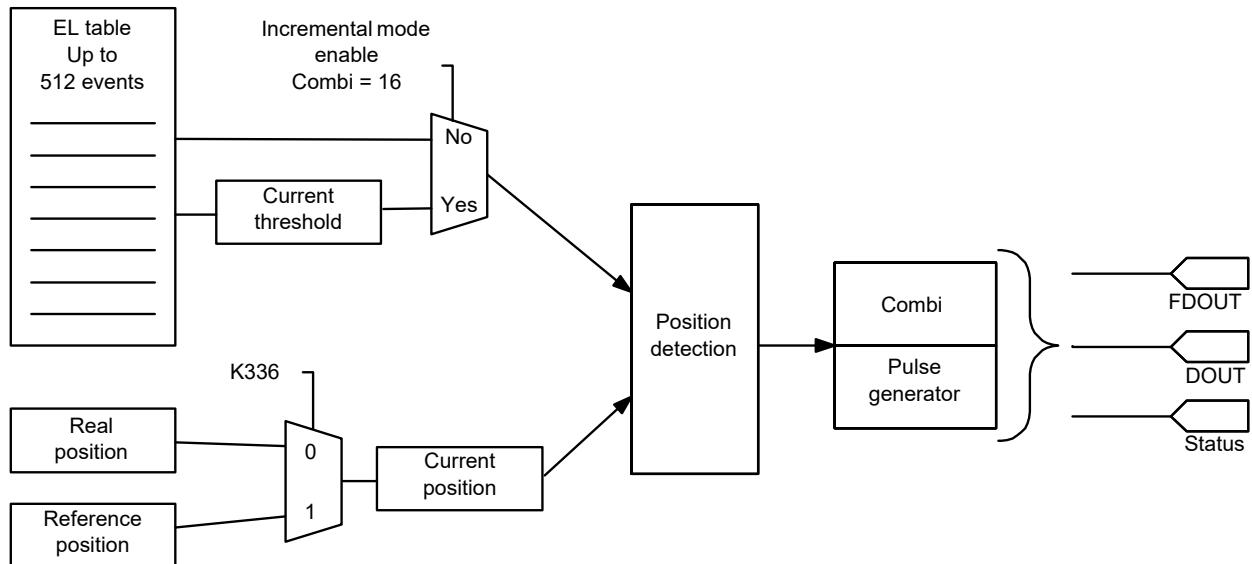
EL0.<axis>=Start_pos, type, combi, PG, ΔP, 0L
EL1.<axis>=Stop_pos, type, combi, PG, 0L, 0L



Remark: In case of the incremental mode, the delta position (ΔP) must be positive if the type of Fast Triggers is equal to 1 and negative if the type of Fast Triggers is equal to 2.
Only one “incremental mode” is possible by activation.
There is one “incremental mode” per axis.
M346 is incremented by 2 only when stop position is reached.
The action induced by reaching a position is a start of pulse generator.

8.9.3.2 Implementation drawing

The mechanisms of 1D Fast Triggers of each axis of the AccurET are shown in the following drawing:



8.9.3.3 Enabling Fast Triggers

The TRE, TRM commands enable or disable the 1D Fast Triggers.

Command	Comment
TRE.<axis>=<P1>,<P2>	Enable/disable Fast Triggers function when axis is not moving: <P1>=table starting line (If -1, Fast Triggers are disabled), <P2>=table line number.
TRM.<axis>=<P1>,<P2>	Enable/disable Fast Triggers function when axis is moving: <P1>=table starting line (If -1, Fast Triggers are disabled) <P2>=table line number.

Remark: When the last event is reached, the Fast Triggers are disabled.
 The TRE, TRM commands need to be sent to re-enable events.
 TRE, TRM commands cannot be used at the same time.
 1D Fast Triggers cannot be used at the same time as the 2D Fast Triggers or Fast Triggers status.
 TRE, TRM commands cannot be restarted for the same axis until the whole enabled events are not reached.
 TRE command waits until the motion is finished before being executed.
 It is not allowed to have more than one P+ Δ P per pulse generator programmed in the EL table when TRE or TRM command is executed

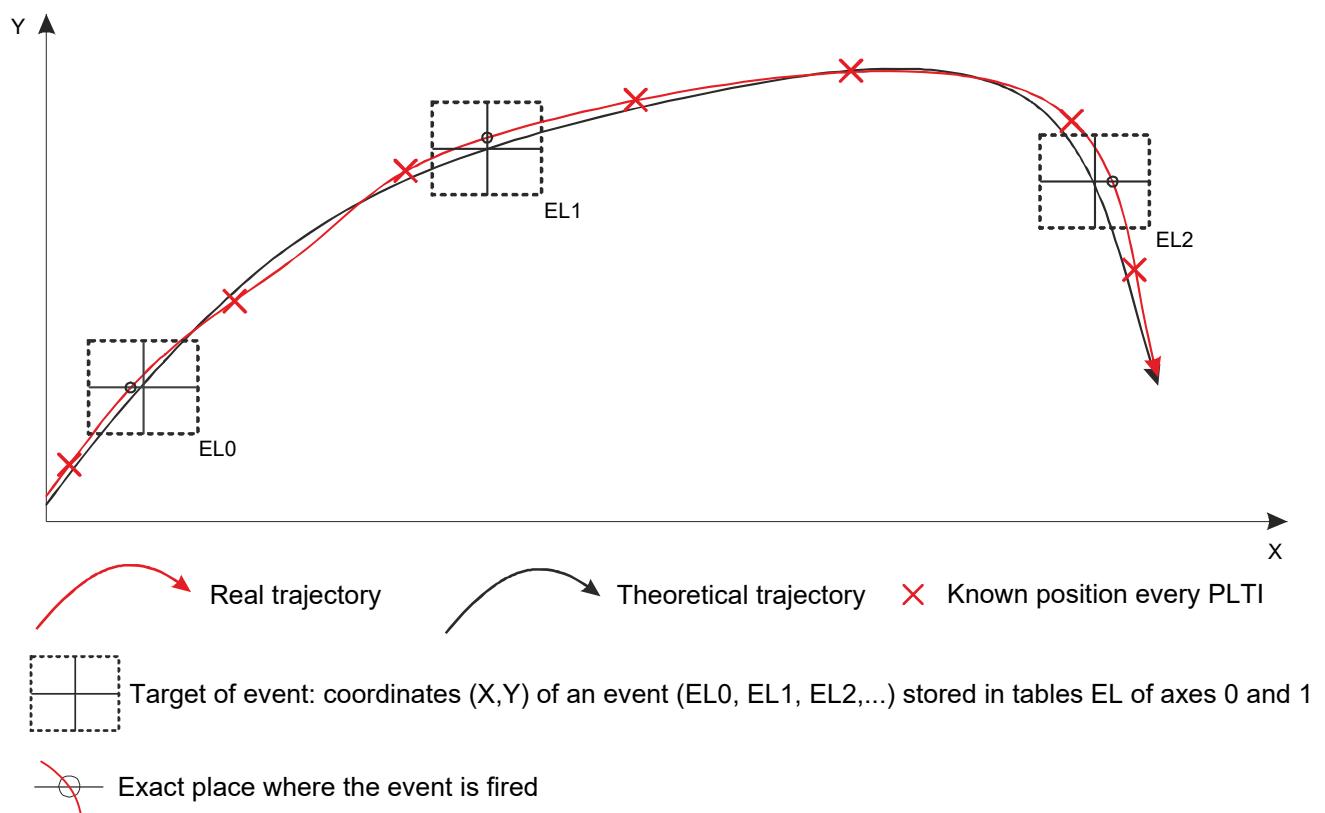
8.9.4 2D Fast Triggers

This feature is available only with a firmware \geq 3.10A.

8.9.4.1 Description

The 2D Fast Triggers are the combination of the existing 1D Fast Triggers functionalities of each axes into one single functionality. As a consequence, it is not possible to use both 1D and 2D Fast Triggers functions together in the same AccurET.

The 2D Fast Triggers make possible to trigger events based on X and Y coordinates of theoretical or real positions:



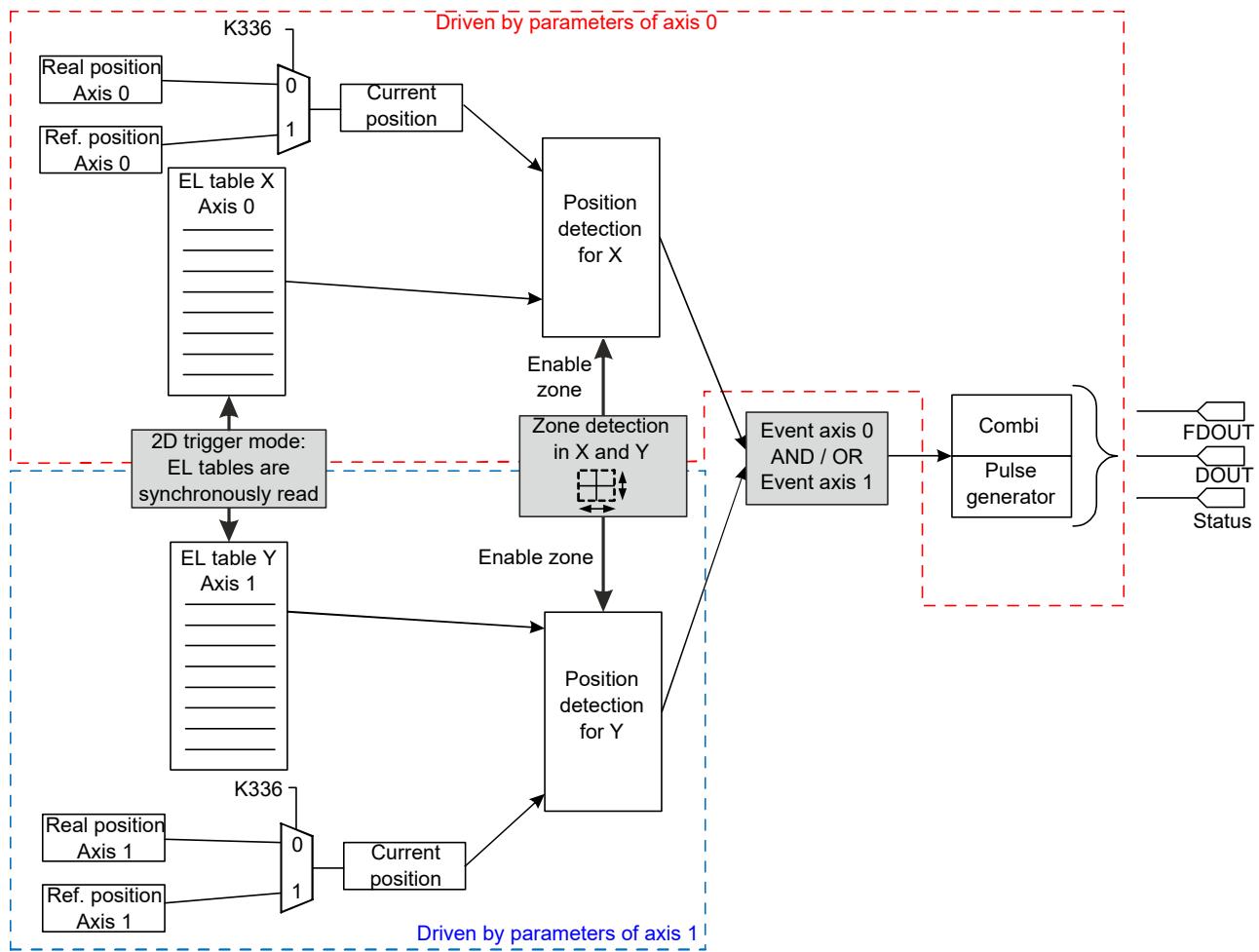
Several options are offered to the user:

- Event is fired on **X OR Y** threshold (the first that occurs fires the event).
- Event is fired once **X AND Y** thresholds have been crossed.
- Event is fired on **X ONLY** threshold (not necessary to cross Y)
- Event is fired on **Y ONLY** threshold (not necessary to cross X)
- Threshold crossing can be either **positive** or **negative** or even **bidirectional**.
- Event is fired when the trajectory goes through a box (a zone) defined around the coordinates X and Y. Note that the box can be the unique condition to fire the event or it can be an "enable" for the previous thresholds detections.

Refer to [§8.9.4.3](#) for more information.

8.9.4.2 Implementation drawing

The existing mechanisms of 1D Fast Triggers of each axis are combined and synchronized as shown by the 2D-specific grey blocks in the next schema:



Remark: Fast Triggers are programmed using the two EL tables: EL of first axis for X and EL of second axis for Y.

A very fast reacting mechanism (10ns resolution) will link the two EL tables together to guarantee that they are always synchronously taken into account.

A unique command "TRM2D" will activate the 2D Fast Triggers functionality on first axis of AccurET (noted in the document as "Axis 0").

The "Combi" and "Pulse Generator" are usable only on axis 0 which must be considered as the "master of 2D Fast Triggers".

To use 2D Fast Triggers with a Gantry you need to use an additional feature describe in [§8.9.9.4](#).

It is not possible to use "incremental mode" in 2D Fast Triggers.

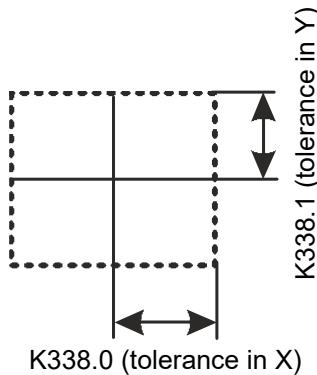
8.9.4.3 Specific to 2D Fast Triggers

8.9.4.3.1 Box Fast Triggers

For each event coordinate XY (given in EL tables), it is possible to enable (or not) a 'box' zone allowing the user to fire the event only if both current positions in X and Y are inside the box. The box enable (or not) is done with the parameter EL "Fast Triggers type" (refer to [§8.9.4.3.2](#)).

Two modes are possible: it can be used either as an enable allowing to fire the event only when the position is inside the box, or as a unique criterion firing the event when the trajectory enters into the box.

The box is always centered at the coordinates XY of the event:



K	Name	Comment
K338	Fast Triggers box tolerance	When box is enabled, the Fast Triggers event will be fired only when the position is within the +/- tolerance.

K338.0 is the tolerance in X: min value=0, max value= $2^{31}-1$.

K338.1 is the tolerance in Y: min value=0, max value= $2^{31}-1$.

Remark: Axis 0 refers to the first axis of AccurET and respectively axis 1 to the second axis of AccurET. The parameter K338 must be set before enabling Fast Triggers (by TRM2D) and remains valid for all the events.
K338 does not have a default unit conversion. It must be converted in the same units as EL's depth 0 which is in [UPI].
The minimum value is 0 and maximum value is $2^{31}-1$.

8.9.4.3.2 Fast Triggers type add on

With the 2D Fast Triggers feature, more types of Fast Triggers conditions are defined:

- box
- threshold detection

Bits	Name	Comment
3	Box	0 = box is disabled 1 = box is enabled (Fast Triggers must occur inside the box)
2	Threshold detection X OR Y / X AND Y	0 = OR 1 = AND
1 and 0	Threshold crossing (for corresponding axis)	0 = could be either positive or negative 1 = positive only 2 = negative only 3 = axis threshold is not taken into account

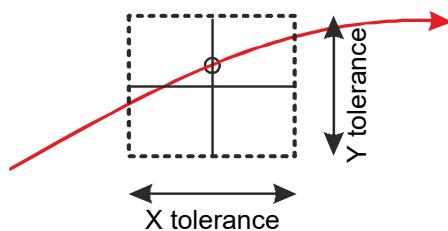
Remark: Bits#0 to 1 must be set in both tables (EL) for each corresponding axis.
Bits#2 to 3 must be set only in table (EL) of axis 0 (first axis of AccurET) as it is global for both axes.

8.9.4.3.3 How to use it

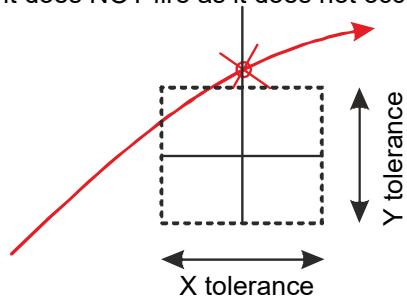
The box can be enabled to allow the threshold detections that occur only inside the box. The setting of the threshold detection mode (X AND Y / X OR Y) is described in [§8.9.4.3.4](#). The following figures illustrate the Fast Triggers condition.

- With box enabled:

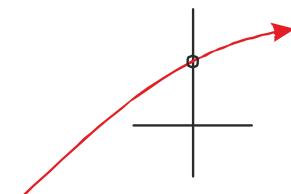
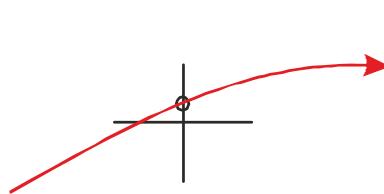
Event fires in the box on X threshold



Event does NOT fire as it does not occur in the box



- Without box: The event is fired whatever the distance from the specified coordinates XY

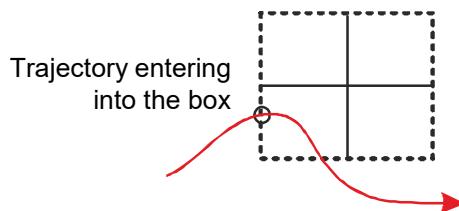


- Box only

The box is used in this case as a unique criterion that fires the event immediately as the trajectory enters into the box.

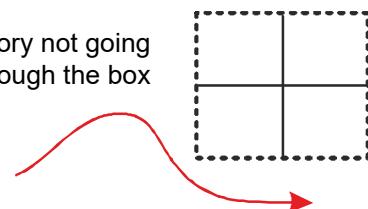
Examples:

Event fires when trajectory enters the box



Event does NOT fire

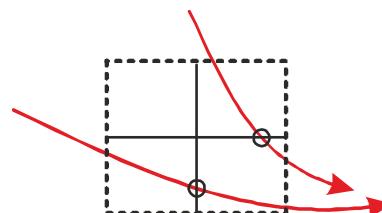
Trajectory not going through the box



- X OR Y threshold crossing

When X OR Y threshold crossing mode is selected, the event is fired at first threshold occurrence.

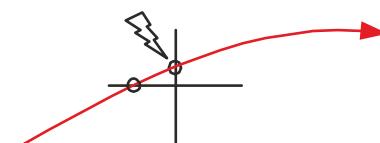
Example of two different trajectories (here with box enabled):



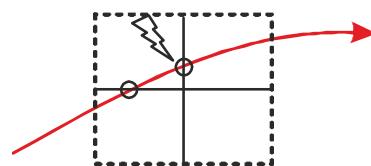
- X AND Y threshold crossing

When X AND Y threshold crossing mode is selected, the event is fired at the second threshold crossed.

Without box



With box enabled

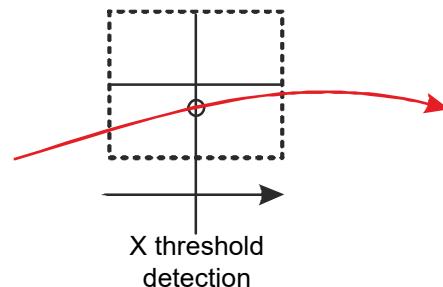


- **X ONLY / Y ONLY threshold crossing**

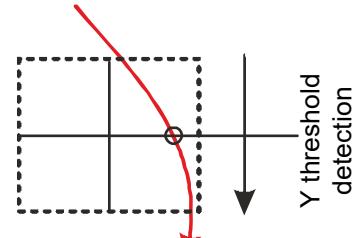
In this mode, we specify which threshold X or Y must be crossed:

Examples (here with box enabled):

With positive X only threshold detection



With negative Y only threshold detection

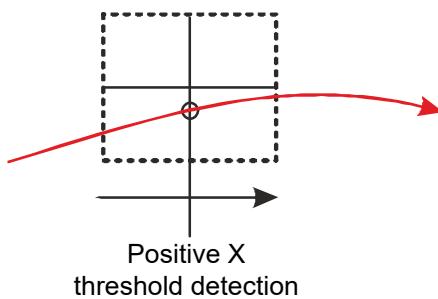


- **Sense of threshold crossing: positive, negative, or without care of the sense**

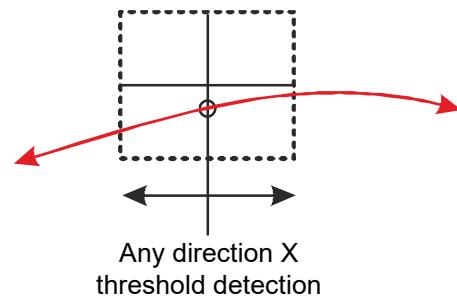
It is possible for each event and for each axis to set threshold sense crossing: as **positive / negative** or even as **not specified**.

Example with a trajectory from left to right or right to left where the X threshold can be detected in both senses (here with box enabled):

with positive X threshold detection



with any direction X threshold detection



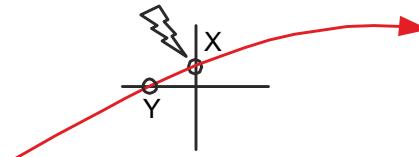
8.9.4.3.4 Usage of EL table per axis

The two EL tables (both axes) are used together but not all fields (depths of EL) are used for both axes:

- For both axes (X and Y), only 'Position in UPI' and 'Fast Triggers type' must be defined separately.
- For axis 0 (X) only, it is needed to define the 'Combi' and 'Pulse generator' in the first table.

Hereafter are illustrated different use cases and settings of the EL table:

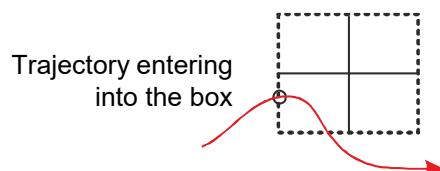
- **Event 125: X AND Y threshold, without care of the crossing direction, no box**



EL125.0=<Position X in UPI>, 4, <Combi>, ...

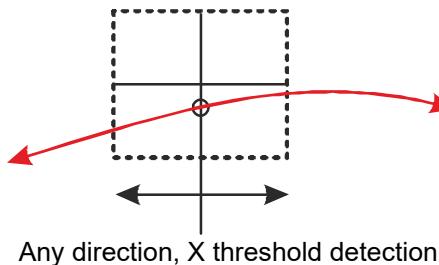
EL125.1=<Position Y in UPI>, 0

- **Event 126: Box Fast Triggers only**



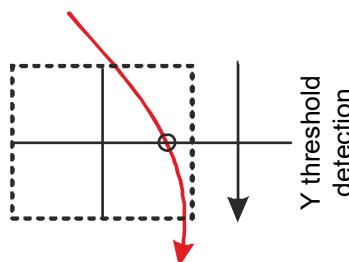
EL126.0=<Position X in UPI>, 11, <Combi>, ... # box enabled, X threshold not taken into account
 EL126.1=<Position Y in UPI>, 3 # Y threshold not taken into account

- **Event 127: X threshold only, without taking care of the crossing direction, box enabled**



EL127.0=<Position X in UPI>, 12, <Combi>, ... # box enabled AND positive or negative X threshold
 EL127.1=<Position Y in UPI>, 3 # Y threshold not taken into account

- **Event 128: Y threshold only, negative crossing direction, box enable**



EL128.0=<Position X in UPI>, 11, <Combi>, ... # box enabled, AND X threshold not taken into account
 EL128.1=<Position Y in UPI>, 2 # negative Y threshold

8.9.4.3.5 Enabling 2D Fast Triggers

Working on the same principle as TRM command (for 1D Fast Triggers), the **TRM2D** command will start the 2D Fast Triggers. One must keep in mind that when 2D Fast Triggers are started, then the events coordinates are taken from both EL tables (from axis 0 and axis 1).

Command	Comment
TRM2D.<axis>=<P1>,<P2>	Enable/disable 2D Fast Triggers function. This command can be used before or even during an axis is executing a movement: <P1>=table starting line (If -1, 2D Fast Triggers are disabled), <P2>=table line number.

Remark: When the last event is reached, the Fast Triggers are disabled.
 The TRM2D command needs to be sent to re-enable events.
 2D Fast Triggers cannot be used simultaneously with 1D Fast Triggers or Fast Triggers status
 TRM2D commands cannot be restarted for the same axis until all the enabled events are not reached.
 It is not possible to use the “incremental mode” with 2D Fast Triggers.

8.9.5 Fast Triggers status

8.9.5.1 Description

The Fast Triggers status is 1D Fast Triggers running at MLTI frequency and with status change as only available action. It is possible to accept few missed events in a certain condition.

Limitation:

- The event must be introduced in a chronological order into the EL table
- All events are defined in the same direction (either positive or negative moves only; no back and forward moves).
- The only possible action is status change (no DOUT firing, etc.)

- This function will run in the MLTI, thus with a precision of about 400 μ s. A maximum of 10 events can be fired in a MLTI.
 - Only available in 1D Fast Triggers
 - The Fast Triggers status cannot be run in the same time as the Fast Triggers function.

8.9.5.2 Command to enable Fast Triggers status

The **TRS** command allows the user to enable/disable Fast Triggers status function when the axis is moving.

Command	<P>	Comment
TRS.<axis>=<P1>,<P2>[,<P3>]	<P1> <P2> [<P3>]	Table starting line (If -1, Fast Triggers are disabled) Table line number Maximum number of missed elements (by default 10)

Remark: If there are more than <P3> events reached when the command is activated, an error will be generated. (refer to [§8.9.7.1](#) for more information).

If the parameter <P3> is not set, the default value is equal to 10.

When the last event is reached the command Fast Triggers status is disabled. A reactivation of this command is needed to reach another time the previous position.

The command TRS cannot be restarted for the same axis as long as all the events activated previously are not reached.

The command TRS cannot be run simultaneously with 1D Fast Triggers or 2D Fast Triggers.

8.9.6 Additional features

8.9.6.1 Output Initialization

The TRR command allows the user to set or reset the DOUT/FDOUT reserved by the Fast Triggers. If TRR is sent during the execution of a triggers sequence (started with TRE or TRM command), this sequence is stopped and the DOUT/FDOUT is set according to the TRR parameters.

Command	<P>	Comment
TRR.<axis>==<P1>[,<P2>]	<P1> [<P2>]	State for the DOUT State for the FDOUT.

- <P1> => Bit#0-15: set state 0 to DOUT (bit#0 for DOUT1=0, bit#1 for DOUT2=0, etc.)
Bit#16-31: set state 1 to DOUT (bit#16 for DOUT1=1, bit#17 for DOUT2=1, etc.)
 - [<P2>] => Bit#0-15: set state 0 to FDOUT (bit#0 for FDOUT1=0, bit#1 for FDOUT2=0, etc)
Bit#16-31: set state 1 to FDOUT (bit#16 for FDOUT1=1, bit#17 for FDOUT2=1, etc)

This command is executed without waiting for the end of the movement.

If the TRR command is sent without parameters, or if a DOUT/FDOUT is set in both states (0 and 1), or if a DOUT/FDOUT is set but not reserved for the Fast Triggers (K359 or C359:1), the **BAD CMD PARAM** error (M64=70) occurs.

8.9.6.2 Command SPG: Start Stop pulse generator

The **SPG** command is used to start or stop the pulse generator.

Command	<P>	Comment
SPG.<axis>=<P1>,<P2>	<P1> <P2>	Pulse generator number (1 or 2) Start or stop (1 or 0).

Remark: This command could be used without Fast Triggers feature. The only thing to do is to reserve the output with register C359 or k359 (refer to [§8.9.2.5](#) for more information).

8.9.6.3 Position source coming from any register

This feature opens the possibility to use 2D Fast Triggers on a gantry. The principle is to share the position with RTV and use it in another drive for Fast Triggers. This feature is only available with firmware \geq 3.12A. Please refer to the example in [§8.9.9.4](#).

8.9.6.3.1 External position register selection

The depth 2 of the parameters **K353** to **K356** are used to define which registers will be used as position for triggering.

K	Depth	Name	Comment
K353	2	Type of register used as a source for Fast Triggers functions.	Register reference type: 3: Source type is a monitoring M, 35: Source type is a monitoring MF, 67: Source type is a monitoring ML
K354	2	Index of register used as a source for Fast Triggers functions.	Register reference index: 0-511
K355	2	Depth of register used as a source for Fast Triggers functions.	Register reference depth: 0-7
K356	2	Axis of register used as a source for Fast Triggers functions	Register reference axis: 0: used the current axis, 1: other axis controller

Remark: If k353:2 is set with a value different of above, an error will be generated (refer to [§8.9.7.1](#) for more information).

8.9.6.3.2 Position type selection add on

Additional values were added to **K336** to select which position will be used by the Fast Triggers feature.

K	Name	Value	Comment
K336	Fast Triggers position type	2	Fast Triggers based on a register
		3	Publishing the real position on ML340 and the real speed on M341
		4	Publishing the theoretical position on ML340 and the theoretical speed on M341

K336 = 3 and 4 allows the update of ML340/M340 and M341. Then these values can be sent over TransnET (with RTV) (refer to [§8.9.2.4](#)). The big advantage to use ML340 is the additional time compensation depending on the encoder type

Use case: 2D trigger on a Gantry configuration with X1, X2 and Y (beam). Trigger are executed on the X1 and X2 axes of the gantry.

X1: K336 = 0 or 1

X2: K336 = 2

Y: K336 = 3 or 4

Remark: ML340/M340 is not equal to ML6 or ML7 and M341 is not equal to M10 or M11.

8.9.6.3.3 Filter selection

The parameter **K360** allows the user to define a mean filter smoothing the input position value when the update frequency is different than 20 KHz.

K	Name	Comment
K360	Mean filter for Fast Triggers position	Mean filter for Fast Triggers position. 1=no filtering. 2-8 = value * PLTI, length of the mean filter

The filter must be set to 2 when the input is updated at 10 KHz (TransnET) or 8 when the input is updated at 2.5 KHz (MLTI).

Remark: The axis taken into account is the one where the Fast Triggers are running.

8.9.6.3.4 Time compensation

The parameter **KF360** allows the user to define the time compensation of the Fast Triggers.

K	Name	Comment
KF360	Fast Triggers time compensation	Defines the Fast Triggers time compensation in seconds

Due to the fact the position is send through RTV, a compensation needs to be done in order to generate the Fast Triggers at the right position. Based on the last variation of the register (in fact the speed), a linear (first order) extrapolation is done to compensate for this delay.

Remark: During the acceleration and deceleration phase the compensation will not be so accurate. Register KF360 is only used when K336=2.

8.9.6.3.5 Setting of external position source

The setting of the external position source can be done by writing all registers directly, or in order to make faster the setting, by using the **CFGTRIGEXTPOS** command.

Command	<P>	Comments
CFGTRIGEXTPOS.<axis> = <P1>,<P2>,<P3>,<P4>,<P5> [<P6>[,<PF7>]]	<P1> <P2> <P3> <P4> <P5> <P6> <PF7>	Define Trigger Pulse Generator (1 or 2) Fast Triggers external register reference type (K353 depth 0 or 1) Fast Triggers external register reference index (K354 depth 0 or 1) Fast Triggers external register reference depth (K355 depth 0 or 1) Fast Triggers external register reference axis (K356 depth 0 or 1) Mean filter for Fast Triggers position (K360) (optional) Fast Triggers time compensation (KF360) (optional)

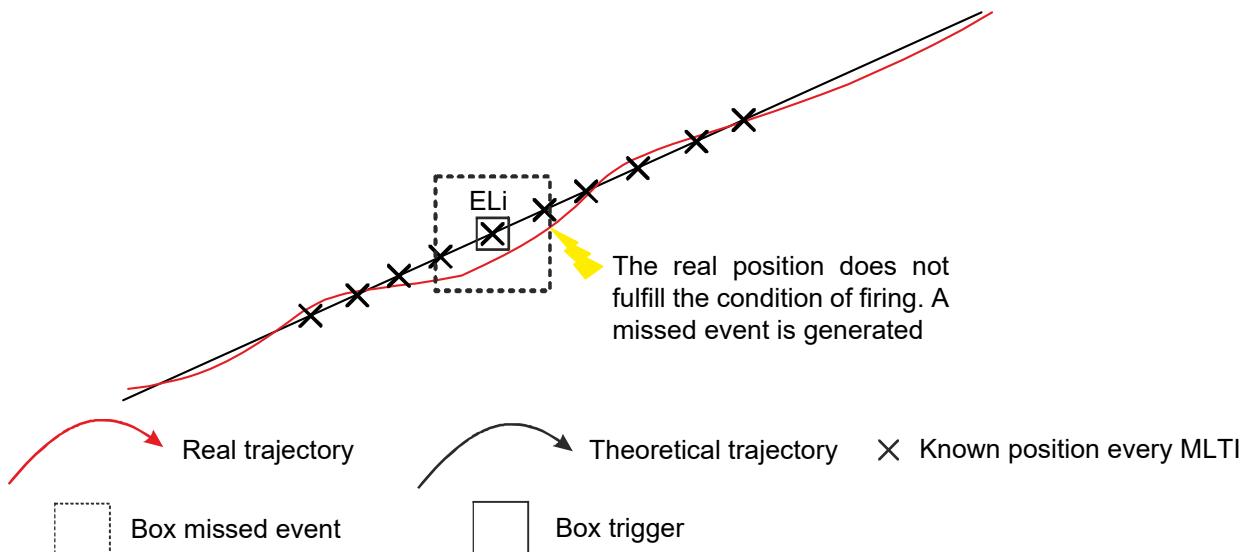
8.9.6.3.6 Missed event

This feature is only available with firmware \geq 3.12A. Refer to the example in [§8.9.9.5](#) for more information.

8.9.6.3.7 Description

The 'missed event' feature informs the user when a Trigger is missed. The user can then generate a warning or an error when an event is missed. It allows the machine to react and adapt the process with limited delays and/or avoid damages to the processed material.

Here is an example of a missed event:



In this case the real position is not following as expected the theoretical position which leads to missing the event.

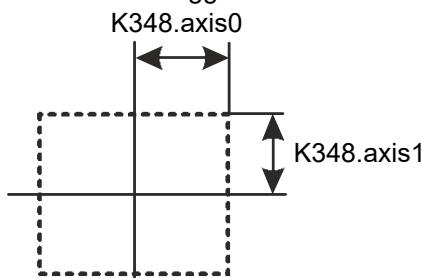
8.9.6.3.8 Specific parameters

- **Missed event configuration**

The following parameters allow the configuration of the missed event detection.

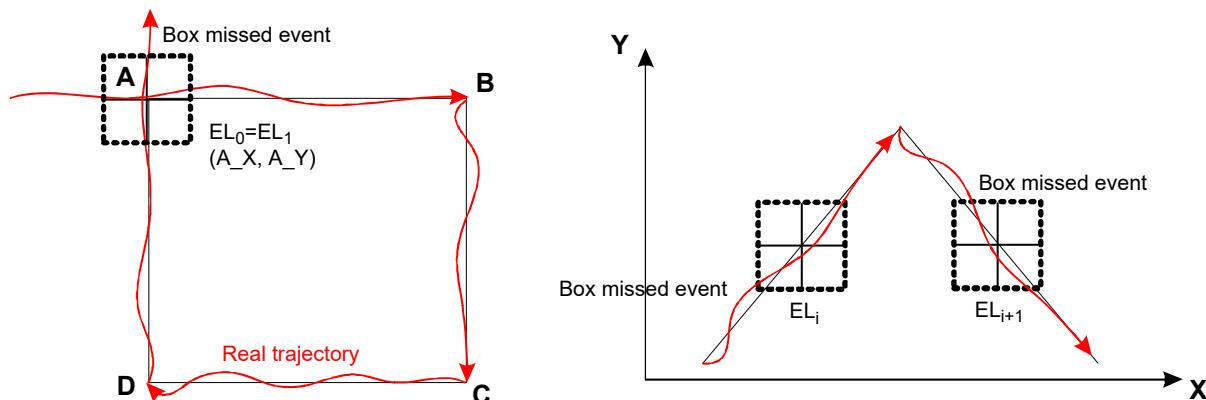
K	Depth	Name	Comment
K347	0	Fast Triggers missed event timeout	Defines the timeout value in [mlti]. The minimum value is 1 MLTI.
K348	0	Fast Triggers missed event detection tolerance	Defines the tolerance of the Fast Triggers missed event detection
K349	0	Fast Triggers missed event action	Defines the action of the Fast Triggers missed event 0: generates the MISSED EVENT error (M64=54), disables the Fast Triggers and fixes the state of output with K357 1: generates the MISSED EVENT warning (M66=24), disables the Fast Triggers and fixes the state of output with K357

Example of setting for k348 in case of 2D Fast Triggers with missed event:



Remark: The setting of this parameter is always valid when 1D Fast Triggers or 2D Fast Triggers are enabled.
The tolerance has to be equal to the maximum distance covered in one MLTI during the time for which the Fast Triggers are enabled.

The two following drawings show examples of false detection of miss event when 2 consecutive events have the same coordinate.



In these cases an error (or a warning) will be raised to indicate Eli+1 is missed. To avoid this false detection a "dummy" event needs to be insert between Eli and Eli+1 or the timeout (K347) need to set to the time to cross them.

- **Additional parameters when feature is used with a position coming from another register (K336=2)**

When the feature is used with a position coming from another register (K336=2), an additional depth has been added to define from which register the missed event will be based.

K	Depth	Name	Comment
K353	3	Type of register used as reference position for missed event feature	Register reference type: 3: Source type is a monitoring M, 67: Source type is a monitoring ML
K354	3	Index of register used as reference position for missed event feature	Register reference index: (0-511)
K355	3	depth of register used as reference position for missed event feature	Register reference depth: 0-7
K356	3	Axis of register used as reference position for missed event feature	Register reference axis: 0: used the current axis, 1: other axis controller
K360	1	Time compensation for missed event	Defines the position compensation: 2 (when missed event used ML0), 3 (when missed event usedML6)

Remark: The setting of K360 has to be done on the axis that manages the missed event (only when the position for missed event is coming from TransnET).

8.9.6.3.9 Enabling missed event

To enable the “missed event” feature, a third parameter is added to the commands **TRE**, **TRM** and **TRM2D**.

Command	Comment
TRE.<axis>=<P1>,<P2>,[<P3>]	Enable/disable Fast Triggers function when axis is not moving: <P1>=table starting line (If <P1>=-1, 1D Fast Triggers are disabled.), <P2>=table line number, [<P3>]=0 missed event disabled 1 Missed event enabled.
TRM.<axis>=<P1>,<P2>,[<P3>]	Enable/disable Fast Triggers function when axis is moving: <P1>=table starting line (If <P1>=-1, 1D Fast Triggers are disabled.), <P2>=table line number, [<P3>]=0 missed event disabled 1 Missed event enabled.
TRM2D.<axis>=<P1>,<P2>,[<P3>]	Enable/disable 2D Fast Triggers function. This command can be used before or even during an axis is executing a movement: <P1>=table starting line (If <P1>=-1, 2D Fast Triggers are disabled.), <P2>= table line number, [<P3>]=0 missed event disabled 1 Missed event enabled.

8.9.6.4 Continuous Fast Triggers

The continuous Fast Triggers is the way to use 1D or 2D Fast Triggers on a circular buffer. This feature is only available with firmware \geq 3.14A. Please refer to example in [§8.9.9.6](#).

8.9.6.4.1 Description

The principle is to fill the EL table and enable continuous Fast Triggers with (TRx.=first_line, -1) and during the execution re-fill the EL already fired.

Several parameters and monitorings have been added to help the user re-fill the events:

- A configurable bit in the user status to inform when there are more free events than predefined number (K337:0).
- M349 returns the number of free events.
- M350 returns the first event free.

Another counter for treated events (M351). This counter can be reset via depth 3 of K320-335.

Remark: The circular buffer has a fixed size of 512.

The continuous Fast Triggers cannot be used with the incremental mode

The events have to be loaded considering they will be used on the continuous Fast Triggers mode:

- Cannot use the events loading from flash
- Cannot load one time the event and loop multiple times through them

It is possible to use the feature missed event with the continuous Fast Triggers mode Event have to be loaded line by line.

- **Specific parameters / commands / monitorings**

K	Name	Comment
K337	Continuous Fast Triggers setting	Depth 0: Number of free events before setting user status bit (reserved by depth1) to 1 Depth 1: Select which bit from user status is used to inform when the user can fill again the EL table

Remark: When the continuous Fast Triggers are enabled, the bit reserved by K337:1 of controller status (M63) will be set when there is more free events than K337:0. The set is done in the MLTI.
 Do not use the bit reserved by K337:1 for something else in the controller status (M63).
 If the continuous Fast Triggers are used with 2D Fast Triggers only the setting of K337.axis0 (first axis of AccurET) is taken in account.
 An error is raised if the parameter's value is out of range (refer to [§8.9.7.1](#) for more information).

To use continuous Fast Triggers, the standard commands to enable Fast Triggers were changed. These changes are highlighted in the following table:

Command	Comment
TRE.<axis>=<P1>,[<P2>],[<P3>]	Enable/disable Fast Triggers function when axis is not moving: <P1>=table starting line (If <P1>=-1, 1D Fast Triggers are disabled.) <P2>=(1-512) number of event enabled -1 infinite events [<P3>]=0 missed event disabled 1 Missed event enabled
TRM.<axis>=<P1>,[<P2>],[<P3>]	Enable/disable Fast Triggers function when axis is moving: <P1>=table starting line (If <P1>=-1, 1D Fast Triggers are disabled.) (if <P1>=-255, the last event for continuous Fast Triggers is entered) [<P2>]=(1-512) number of event enabled -1 infinite events [<P3>]=0 missed event disabled 1 Missed event enabled
TRM2D.<axis>=<P1>,[<P2>],[<P3>]	Enable/disable 2D Fast Triggers function. This command can be used before or even during an axis is executing a movement: <P1>= table starting line (If <P1>=-1, 2D Fast Triggers is disabled.) (if <P1>=-255, the last event for continuous Fast Triggers is entered) [<P2>]=(1-512) number of event enabled -1 infinite events [<P3>]=0 missed event disabled 1 Missed event enabled

Remark: The enable is done by sending a command TRE/TRM or TRM2D with <P2> set to “-1”.
 The parameter P2 is not mandatory if <P1> is equal to -255 or -1.
 When the last event is entered send the command TRM or TRM2D=-255 in order to stop the Fast Triggers when the all programmed events have been fired.
 The command TRE=-255 is not allowed because the command TRE cannot be executed when the axis is moving.
 The commands TRM or TRM2D=-255 have to be sent when the trigger is enabled.

M	Name	Comment
M349	Number of free event	Returns number of free event in EL memory
M350	Index of the first event free	Return the index of the first event free

Remark: When the whole EL table is loaded and the Fast Triggers are enabled, M349 indicated 2 and not 0, because the two first events are preloaded.

8.9.7 Error and warning

8.9.7.1 Error troubleshooting

- **ERR CFG TRIGGER (M64=46)**

This error is raised when a register or command has an inconsistent value. List of M264 values:

M264	Comment
460001	Depth 1 (Fast Triggers type) of EL parameter is out of the range.
460002	Depth 2 (combi) of EL parameter is out of the range
460003	Depth 3 (pulse generator) of EL parameter is out of the range
460004	Too many events are enabled with command TRS (paramete1 + parameter2 > 256) with firmware < 3.10A (paramete1 + parameter2 > 512) with firmware ≥ 3.10A
460005	Too many events are enabled with command TRE,TRM or TRM2D (paramete1 + parameter2 > 256) with firmware < 3.10A (paramete1 + parameter2 > 512) with firmware ≥ 3.10A
460006	PG1 or PG2 is defined by both axes. This error occurs only when the TRE, TRM, TRM2D or SPG is executed. At least one of the following parameters is set on both axis: K342 (on both axis the value > 2) K343 (on both axis the value > 2) K344 (on both axis the value > 2) K339 (on both axis the value is different of 0) K345 (on both axis the value is different of 0) Kf342 (on both axis the value is different of 0) Kf343 (on both axis the value is different of 0) Kf339 (on both axis the value is different of 0)
460007	Both K342 and Kf342 are set for the same PG: condition not allowed
460008	Both K343 and Kf343 are set for the same PG: condition not allowed
460009	Both K339 and Kf339 or K344 and Kf339 are set for the same PG: condition not allowed
460011	KF353 is set to 67. It is not allowed to use ML register to modulate the pulse generator
460013	Depth 0 of K337 is out of range. (1 to 512 included)
460014	Depth 1 of K337 is out of range. (0 to 15 included)

- **TOO MISS EVENTS (M64=52)**

This error is raised when the TRS command is entered and there are too many missed events during the first MLTI. (More than <parameter3> or 10 if this one was not set).

- **TRIGGER ENABLED (M64=53)**

This error is raised when the command TRS is issued and there is one of the following features already running:

- Fast Triggers activation (TRE/TRM or TRM2D)
- Fast Triggers status (TRS)

- **TRIGGER MISSED (M64=54)**

This error is raised when the feature Missed ([§8.9.6.3.6](#)) event is enabled and an event is missed. List of M264 values:

M264	Comment
540001	An event is missed on 1D Fast Triggers
540002	An event is missed on 2D Fast Triggers

8.9.7.2 Warning troubleshooting

- **TRIGGER MISSED (M66=24)**

This warning is raised when the feature Missed event ([§8.9.6.3.6](#)) is enabled with K349=1 and an event is missed.

- **TRIGGER CONTINUO (M66=29)**

This warning is raised when the next event to fire is not programmed in the EL table. The warning is cleared after a RST command or after re-enabling the Fast Triggers.

8.9.7.3 Error management

With firmware < 3.10A, in case of error on an axis, the Fast Triggers function is not stopped and the next event is still pending.

With firmware ≥ 3.10A, in case of error on an axis the Fast Triggers will automatically stop. It is like sending automatically a 'TRE= -1' command.

For 1D Fast Triggers mode, the Fast Triggers functionality is stopped only on the considered axis while for 2D Fast Triggers mode when the first axis of AccurET is in error then the Triggers are stopped.

- **Parameter K357 (refer also to [§7.4.3.3](#)) in case of error:**

The parameter K357 lets the possibility to define every DOUT and FDOUT that must be set or reset when an error occurs on an axis. It is also possible to use K357 with the DOUT or FDOUT which are used by the 1D Fast Triggers or 2D Fast Triggers (specified by registers K359 and/or C359).

When 1D Fast Triggers are used, each axis can specify its own K357 while with 2D Fast Triggers, only the K357 of first axis can be used.

In case of error when 1D Fast Triggers or 2D Fast Triggers are used, it defines the mask of the digital output (DOUT) or fast digital output (FDOUT) that must be set or reset when the controller is in error.

Depth 0:

Bit#0-15: initial state 1 for DOUT (bit#0 for DOUT1=0, bit#1 for DOUT2=0)

Bit#16-31: initial state 0 for DOUT (bit#16 for DOUT1=1, bit#17 for DOUT2=1)

Depth 1:

Bit#0-15: initial state 1 for FDOUT (bit#0 for FDOUT1=0, bit#1 for FDOUT2=0)

Bit#16-31: initial state 0 for FDOUT (bit#16 for FDOUT1=1, bit#17 for FDOUT2=1, etc.)

When depth 0=0 then the DOUTs will not be affected in case of error

When depth 1=0 then the FDOUTs will not be affected in case of error

- **Behavior in case of RST command for DOUT/FDOUT reserved for Fast Triggers:**

When an error is reset (for instance by a RST command), then the DOUT and FDOUT states does not change until a new Fast Triggers event occurs.

8.9.8 Limitation

- The minimum time between 2 events is about 60ns.
- When using the "incremental mode", the ΔPosition (EL Parameter) needs to be entered in increment. If a conversion is used, check that the conversion does not return a rounded value. If it is not the case the "rounded error" will be accumulated.

EL0.<axis>=Start_pos, type, combi, PG, ΔP, 0L

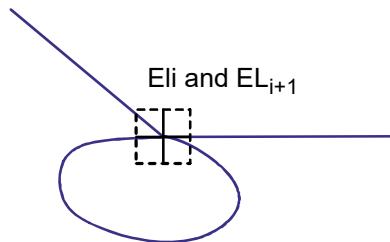
EL1.<axis>=Stop_pos, type, combi, PG, 0L, 0L

After each event reached, the next event to reach is obtained by adding the ΔP to the previous event until reaching the last position. So if this ΔP is rounded, the error will be accumulated.

- When using the 2D Fast Triggers with a gantry system.

If the gantry is in level 2 there is no possibility to run the 2D Fast Triggers on the gantry axis because it is not possible to load EL table on slave axis. Two solutions are available:

- Run 2D Fast Triggers on the other axis if this one is not in gantry level2
 - Switch to level 1 the gantry
- There are some applications that might require executing action at the same point. The following example depicts an application where a laser needs to be turned off at a specific position after completing for e.g. a cutting operation in one direction and then needs to be turned on at the same position to initiate the cutting in another direction. The status of the laser could be typically managed via a digital output set/reset via the triggering mechanism.
- The blue line correspond to the trajectory where the laser is on. (Eli disable the laser and Eli+1 re-enable it).



In such a situation, it is possible that both events (Eli and Eli+1) are fired simultaneously if the condition to trigger them is “not sufficiently distinctive”. This issue can be addressed by using the IMP library.

8.9.9 How to

First of all, using ETEL’s Fast Triggers feature requires configuration of Fast Triggers parameter registers. Once the parameter registers are correctly set, the user can enable the Fast Triggers function by sending a command.

If the Fast Triggers configuration is incorrectly set, the AccurET controller will generate Error 46. Refer to [§8.9.7.1](#) for more information.

Several examples on how to use the Fast Triggers feature can be found in this section. Each example contains a short description, to facilitate the user’s understanding of how it works.

- All the examples use ETEL’s sequence program.
- The general rules for programming ETEL’s sequence is applied to all examples in this section. Refer to chapter Sequence programming for more information.

How to use the Fast Triggers’ example sequences:

- 1) Update AccurET FW to the latest version (3.14A or higher).
- 2) Configure the motion system and make sure it is safe to operate.
- 3) Download a Sequence sample code to AccurET or UltimET (recommended)
- 4) Execute the sample

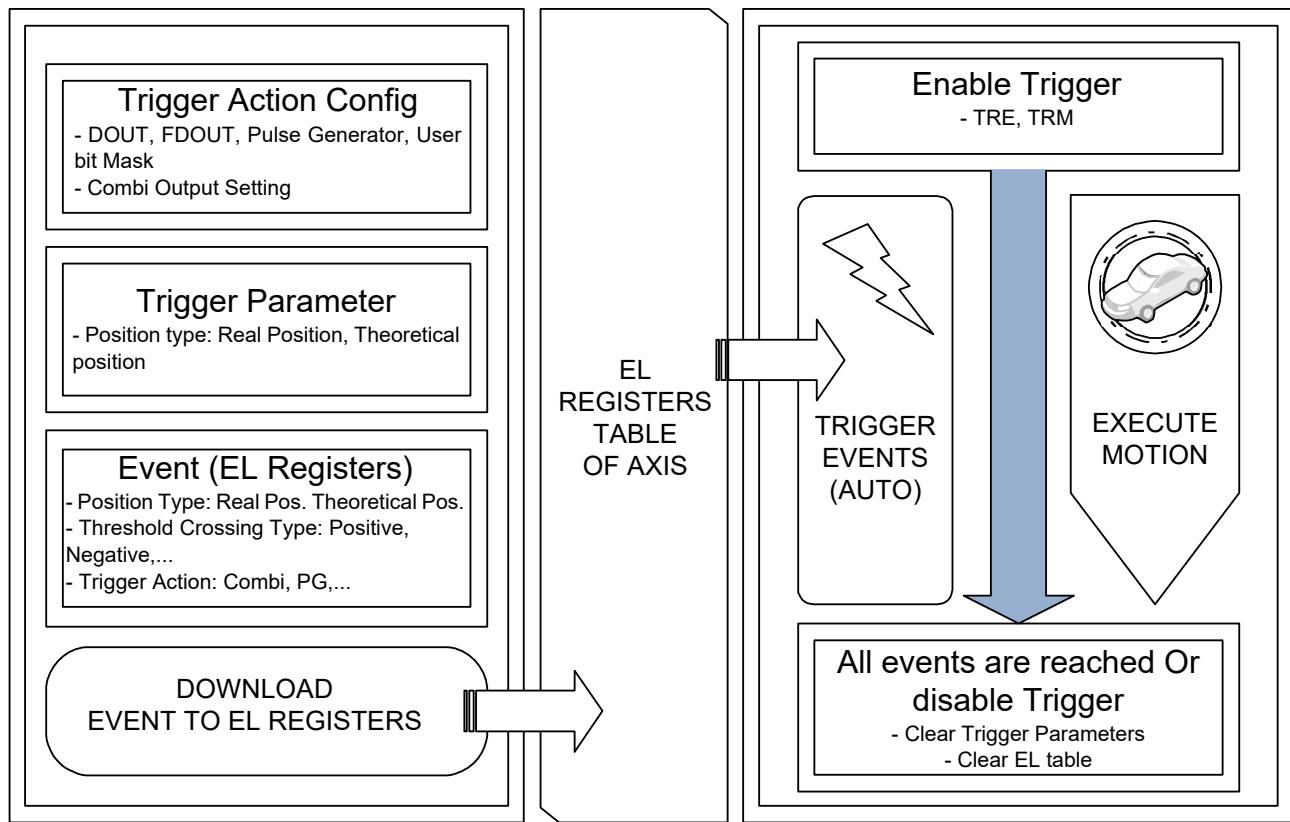
Overview of the Fast Triggers examples:

- The simplest Fast Triggers feature is 1D position threshold Fast Triggers (refer to [§8.9.9.1](#)).
- The simplest event feature is Combi with set/clear Digital Output (refer to [§8.9.9.1](#)).
- 1D or 2D Fast Triggers can be configured for UltimET interpolated motion (e.g. PVT refer to [§8.9.9.2](#)).
- “Position source coming from any register” feature allows user to setup a 2D Fast Triggers on gantry XY system (refer to [§8.9.9.4](#)).
- More than 512 EL registers can be continuously executed with the “Continuous Fast Triggers” feature (refer to [§8.9.9.6](#)).
- Wrong Fast Triggers setting might cause Error 46 (refer to [§8.9.7.1](#)). The reset function example can be used to optionally clear the Fast Triggers configuration (refer to [§8.9.9.7](#)).

WARNING: The sample codes were created to support the user in getting familiar with ETEL’s Fast Triggers feature. Any use of this sample code is covered by the latest ETEL software distribution disclaimer. For more information, please contact your nearest ETEL representative.

8.9.9.1 1D Fast Triggers

This example shows how to setup 1D Fast Triggers on an axis of the AccurET controller. The 1D Fast Triggers "action" generates digital outputs (DOUT, FDOUT, Pulse Generator) when the Fast Triggers algorithm detects the crossing of EL register's position threshold.

**Example:**

```

# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: deg, deg/s, deg/s2, V, A, s
#define Ax 1
#define DOUT_TriggerMask 0x2          //DOUT2
#define DOUT_PulseMask 0x2           //DOUT2
#define FDOUT_TriggerMask 0x8         //FDOUT4
#define FDOUT_PulseMask 0x1          //FDOUT1
void trigger1Dtest(void)
{
    // Set TriggerReference type;
    k336.Ax=1;                      // When 0, uses real position as trigger reference.
    //TRR.Ax=0,2;
    // Flush out and reserve DOUT and FDOUT Trigger masks and initialize them.
    if (K359.Ax!=0)
        K359.Ax=0;
    if (C359:1.Ax!=0)
        C359:1.Ax=0;
    // ...or better yet, use logical operations instead.
    // if (K359.Ax & DOUT_TriggerMask) K359.Ax ^= DOUT_TriggerMask;
    // if (C359:1.Ax & FDOUT_TriggerMask)
    C359:1.Ax ^= FDOUT_TriggerMask;
    // Reset DOUT and FDOUTs.
    K171.Ax=0;
  
```

```
C5.Ax=0;
K359.Ax=DOUT_TriggerMask;
C359:1.Ax=FDOUT_TriggerMask + FDOUT_PulseMask;
// Set events
EL0.Ax=lconv(0.02, _upi, Ax), 1, 0L, 0L;
EL1.Ax=lconv(0.03, _upi, Ax), 1, 1L, 0L;
EL2.Ax=lconv(0.04, _upi, Ax), 1, 1L, 1L;
//Start incremental trigger with Pulse Generator 2 (i.e., pulse for every
delta position)
EL3.Ax=lconv(0.049, _upi, Ax), 2, 16L, 3L, lconv(-0.001, _upi, Ax), 0L;
//Stop incremental trigger with Pulse Generator 2 (i.e., pulse for every
delta position)
EL4.Ax=lconv(0.005, _upi, Ax), 2, 16L, 4L, 0L, 0L;
// Ending incremental mode trigger will add 2 to M346.
// Set scenarios (combi)
// Combi 1
K320:0.Ax=DOUT_TriggerMask;
K320:1.Ax=FDOUT_TriggerMask;
// Combi 2
K321:0.Ax=DOUT_TriggerMask<<0X10;
K321:1.Ax=FDOUT_TriggerMask<<0X10;
// Combi 3
K322:0.Ax=0;
K322:1.Ax=0;
// Set scenarios (pulse)
// PG1
K340:1.Ax=FDOUT_PulseMask; // set Trigger Pulse Generator1 mask (DOUT,FDOUT)
k342:0.Ax=0.1; // (100ms) Trigger Pulse Generator delay
k343:0.Ax=0.004; // (4ms) Trigger Pulse Generator pulse width
k344:0.Ax=0.006; // (6ms) Trigger Pulse Generator interval pulse
k345:0.Ax=4; //Trigger Pulse Generator pulse count
// PG2
K341:0.Ax=DOUT_PulseMask; // set Trigger Pulse Generator1 mask (DOUT, FDOUT)
k342:1.Ax=0.0; // (0ms) Trigger Pulse Generator delay
k343:1.Ax=0.0002; // (200µs) Trigger Pulse Generator pulse width
k344:1.Ax=0.0001; // (100µs) Trigger Pulse Generator interval pulse
k345:1.Ax=2; // Trigger Pulse Generator pulse count
TRE.Ax=0,5;
PWR.Ax=1;
spd.Ax=0.3;
acc.Ax=5.0;
jrt.Ax=0.03;
mve.Ax=0.0;
wtm.Ax;
mve.Ax=0.05;
wtm.Ax;
wtt.Ax=0.05;
acc.Ax=1.0;
jrt.Ax=0.3;
mve.Ax=0.0;
wtm.Ax;
TRE.Ax=-1;
TRR.Ax=DOUT_TriggerMask,FDOUT_TriggerMask + FDOUT_PulseMask;
}
```

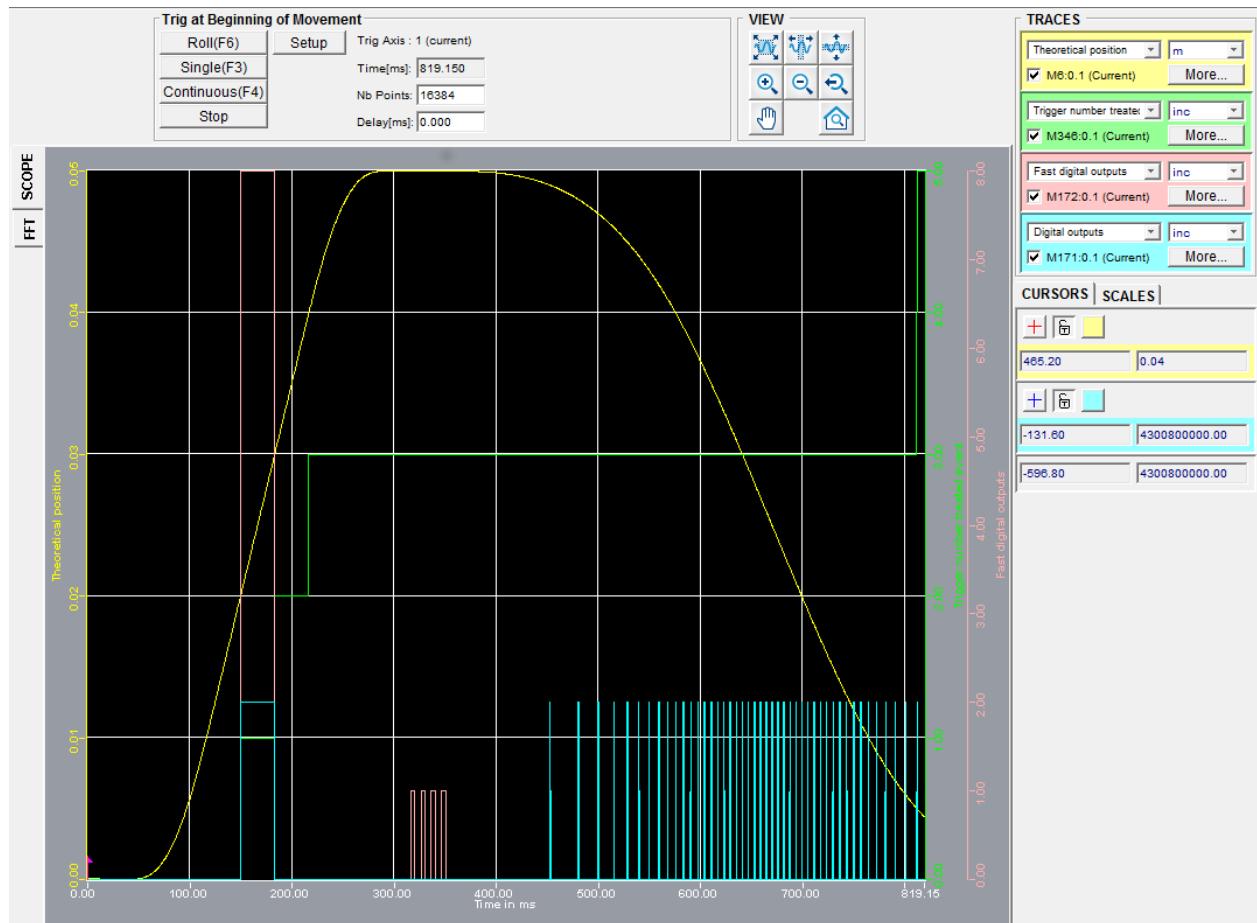
```

void reset_all_pg(void)
{
    TRM.Ax = -1;
    SPG.Ax = 1,0;
    SPG.Ax = 2,0;
    C359.Ax = 0, 0;
    K340.Ax = 0, 0;
    K341.Ax = 0, 0;
    KF339.Ax = 0f, 0f;
    KF342.Ax = 0f, 0f;
    KF343.Ax = 0f, 0f;
    K339.Ax = 0, 0;
    K342.Ax = 0, 0;
    K343.Ax = 0, 0;
    K344.Ax = 0, 0;
    K345.Ax = 0,0;
    K353.Ax = 0,0;
    K354.Ax = 0,0;
    K355.Ax = 0,0;
    K356.Ax = 0,0;
    NEW.Ax = 8;      // Clears EL tables in RAM memory
}

void main(void) {
    reset_all_pg();
    trigger1Dtest();
}

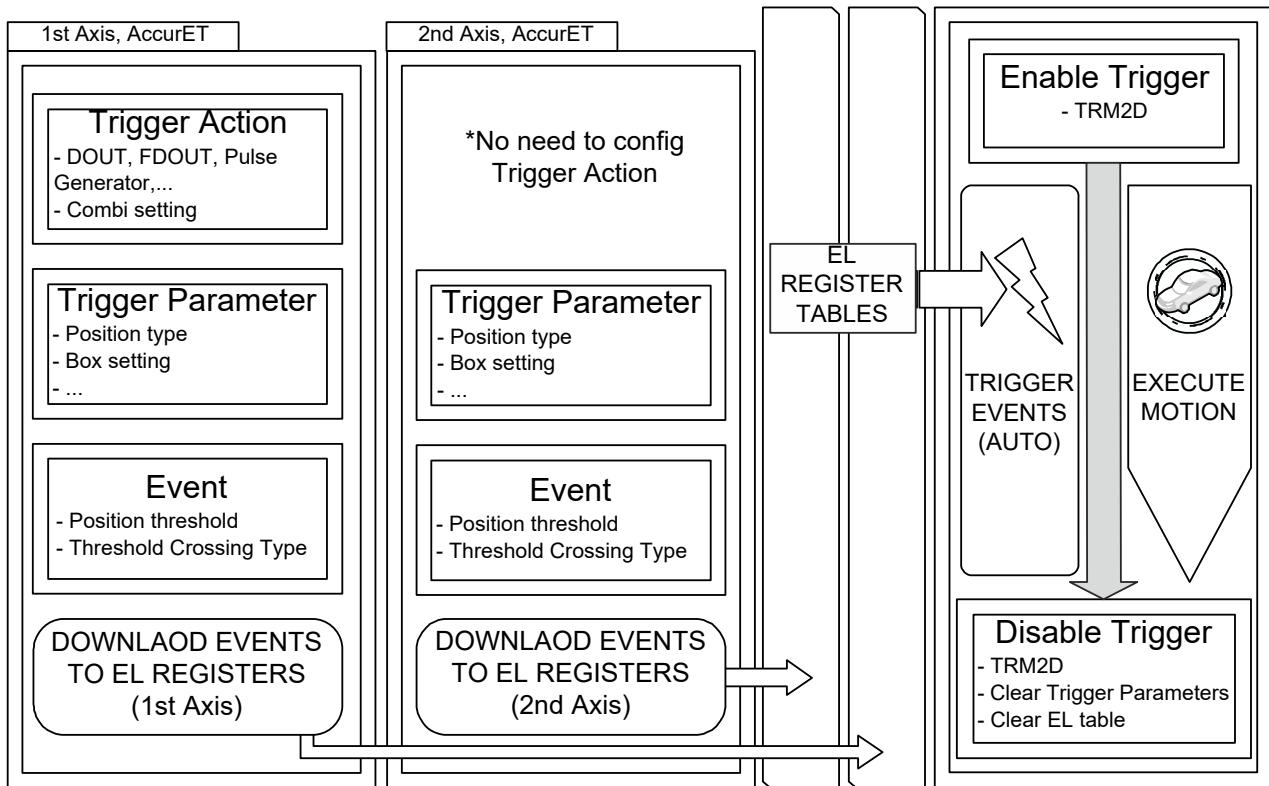
void func123(void) {main();}

```



8.9.9.2 2D Fast Triggers

The 2D Fast Triggers require configuration on two axes of the AccurET to be used with two-axis motion.



It can be also combined with two axes interpolated motion if UltimET light interpolated motion is available. Refer to the UltimET documentation.

Example:

```

# COMET special parameters
# ISO linear units: m, m/s, m/s2, A, V, s (used in case of interpolation)
# ISO rotary units: t, t/s, t/s2, A, V, s

#define AX0 2
#define AX1 3
#define AX0_IPOL_MASK 4L
#define AX1_IPOL_MASK 8L
#define FDOUT_TriggerMask 0x8
#define NOPG 0L
#define BOX 8L
#define XY_OR 0L
#define XY_AND 4L
#define THR_ANY 0L
#define THR_POS 1L
#define THR_NEG 2L
#define NO_THR 3L
#define COMBI0 0L
#define COMBI1 1L

void triggerConfig(void) {
    C359:1.AX0 = 0xC; // FDOUT3,4 is used for trigger
    TRR.AX0 = 0,FDOUT_TriggerMask;
    K336.(AX0,AX1)=1;// Trigger type real or theo.
  
```

```

// Box config.
K338.AX0=lconv(0.00001, _upi, AX0);
K338.AX1=lconv(0.00001, _upi, AX1);

// Combies
K320:1.AX0=FDOUT_TriggerMask;// Combi 0 for FDOUT (depth 1): Set FDOUT
K321:1.AX0=FDOUT_TriggerMask << 0x10;// Combi 1 for FDOUT (depth 1): ReSet
FDOUT

// PG
// NO PG
}

void eventTableConfig(void) {
    EL0.AX0=lconv(0.000536, _upi, AX0), BOX|XY_AND|NO_THR, 0x0, 0x0;
    EL0.AX1=lconv(-0.000115, _upi, AX1), THR_POS, 0x0, 0x0;
    EL1.AX0=lconv(0.0, _upi, AX0), BOX|XY_AND|NO_THR, 0x1, 0x0;
    EL1.AX1=lconv(0.002, _upi, AX1), THR_POS, 0x0, 0x0;
    EL2.AX0=lconv(0.0, _upi, AX0), BOX|XY_AND|THR_NEG, 0x0, 0x0;
    EL2.AX1=lconv(0.002, _upi, AX1), NO_THR, 0x0, 0x0;
    EL3.AX0=lconv(-0.001, _upi, AX0), BOX|XY_AND|THR_NEG, 0x1, 0x0;
    EL3.AX1=lconv(0.003, _upi, AX1), NO_THR, 0x0, 0x0;
    EL4.AX0=lconv(-0.001, _upi, AX0), BOX|XY_AND|THR_NEG, 0x0, 0x0;
    EL4.AX1=lconv(0.003, _upi, AX1), NO_THR, 0x0, 0x0;
    EL5.AX0=lconv(-0.003, _upi, AX0), BOX|XY_AND|THR_NEG, 0x1, 0x0;
    EL5.AX1=lconv(0.001, _upi, AX1), NO_THR, 0x0, 0x0;
    EL6.AX0=lconv(-0.003, _upi, AX0), BOX|XY_AND|NO_THR, 0x0, 0x0;
    EL6.AX1=lconv(0.001, _upi, AX1), THR_NEG, 0x0, 0x0;
    EL7.AX0=lconv(-0.002, _upi, AX0), BOX|XY_AND|THR_ANY, 0x1, 0x0;
    EL7.AX1=lconv(-0.002, _upi, AX1), THR_ANY, 0x0, 0x0;
    EL8.AX0=lconv(-0.002, _upi, AX0), BOX|XY_AND|THR_ANY, 0x0, 0x0;
    EL8.AX1=lconv(-0.002, _upi, AX1), THR_ANY, 0x0, 0x0;
    EL9.AX0=lconv(-0.0011, _upi, AX0), BOX|XY_AND|THR_POS, 0x1, 0x0;
    EL9.AX1=lconv(-0.003, _upi, AX1), NO_THR, 0x0, 0x0;
    EL10.AX0=lconv(-0.0011, _upi, AX0), BOX|XY_AND|THR_POS, 0x0, 0x0;
    EL10.AX1=lconv(-0.003, _upi, AX1), NO_THR, 0x0, 0x0;
    EL11.AX0=lconv(0.0001, _upi, AX0), BOX|XY_AND|THR_POS, 0x1, 0x0;
    EL11.AX1=lconv(-0.0021, _upi, AX1), NO_THR, 0x0, 0x0;
    EL12.AX0=lconv(0.0001, _upi, AX0), BOX|XY_AND|NO_THR, 0x0, 0x0;
    EL12.AX1=lconv(-0.0021, _upi, AX1), THR_POS, 0x0, 0x0;
    EL13.AX0=lconv(0.000536, _upi, AX0), BOX|XY_AND|NO_THR, 0x1, 0x0;
    EL13.AX1=lconv(-0.000115, _upi, AX1), THR_POS, 0x0, 0x0;
}

void moveTest(void) {

MVE.AX0=0.0010000; //Start movement
WTM.AX0;
MVE.AX1=-0.0020000; //Start movement
WTM.AX1;
// IPOL MVProfile Units
//Multiplying factor for the position. The depths correspond to the ipol
group numbers.
*K522=1; //Multiplied by 1
}

```

```
// Power of ten for the position. The depths correspond to the ipol group
numbers.
*K523=-6; //1 micrometer
//Power of ten for the speed. The depths correspond to the ipol group
numbers.
*K524=-6;// 1 micrometer / second
//Power of ten for the acceleration. The depths correspond to the ipol group
numbers.
*K525=-6;// 1 micrometer / second squared
//Power of ten for the time. The depths correspond to the ipol group
numbers.
*K526=-6; // 1 microsecond
*ISET=0,AX0_IPOL_MASK,AX1_IPOL_MASK,0x0L,0x0L;// Sets the interpolation axis
-- Ax2 and Ax3
*IBEGIN = 0;
*ITSPD = 0,10000; // tangential speed of 10000 um/s
*ITACC = 0,10000; // tangential acceleration of 10000 um/s2
*ITDEC = 0,10000; // tangential deceleration of 10000 um/s2
*ITJRT = 0,14; // tangential jerk time of 14 microsecs

*IABSMODE=0,1;
///*ICONC=0;

TRM2D.(AX0,AX1)=0,14;
X100.AX0 = 1;

// EVT 0 motion
*ILINE=0, 1000, -2000, 0, 0;
*IWTT=0,10000;

// EVT 1,2 motion
//TRM2D.(AX0,AX1)=1,2;
*IPVT=0, 0, 2000, 0,0, -300, 700, 0,0, 500000;
*IPVT=0, 0, 2000, 0,0, -500, 500, 0,0, 725000;

// EVT 3,4 motion
//TRM2D.(AX0,AX1)=3,2;
*IPVT=0, -1000, 3000, 0,0, -500, 500, 0,0, 500000;
*IPVT=0, -1000, 3000, 0,0, -300, -700, 0,0, 500000;

// EVT 5,6 motion
//TRM2D.(AX0,AX1)=5,2;
*IPVT=0, -3000, 1000, 0,0, -300, -700, 0,0, 500000;
*IPVT=0, -3000, 1000, 0,0, 300, -700, 0,0, 500000;

// EVT 7,8 motion
//TRM2D.(AX0,AX1)=7,2;
*IPVT=0, -2000, -2000, 0,0, 300, -700, 0,0, 500000;
*IPVT=0, -2000, -2000, 0,0, 500, -500, 0,0, 500000;

// EVT 9,10 motion
//TRM2D.(AX0,AX1)=9,2;
*IPVT=0, -1100, -3000, 0,0, 500, -500, 0,0, 500000;
*IPVT=0, -1100, -3000, 0,0, 500, 500, 0,0, 500000;
```

```

// EVT 11,12 motion
//TRM2D.(AX0,AX1)=11,2;
*IPVT=0, 100, -2100, 0,0, 500, 500, 0,0, 500000;
*IPVT=0, 100, -2100, 0,0, 300, 700, 0,0, 500000;

*IPVT=0, 1000, 2000, 0,0, 300, 700, 0,0, 500000;

*WTM=0;
/*INCONC=0;
*IEND=0;

TRM2D.(AX0,AX1)=-1;
X100.AX0 = 0;
}

void reset_all_trigger_settings(void)
{
    // Stop trigger mode
    //TRM.(AX0, AX1) = -1;
    // Stop PGs

    //SPG.(AX0, AX1) = 1,0;

    //SPG.(AX0, AX1) = 2,0;
    // Reset Position Type selection
    K336.(AX0, AX1) = 0; // Real pos (ML7)
    // Reset output masks for trigger
    K359.(AX0, AX1) = 0;// DOUT output mask
    C359.AX0 = 0, 0;// FDOUTs output mask
    // K171.(AX0, AX1) = 0;// Reset DOUTs
    // C5.(AX0, AX1) = 0;// Reset FDOUTs

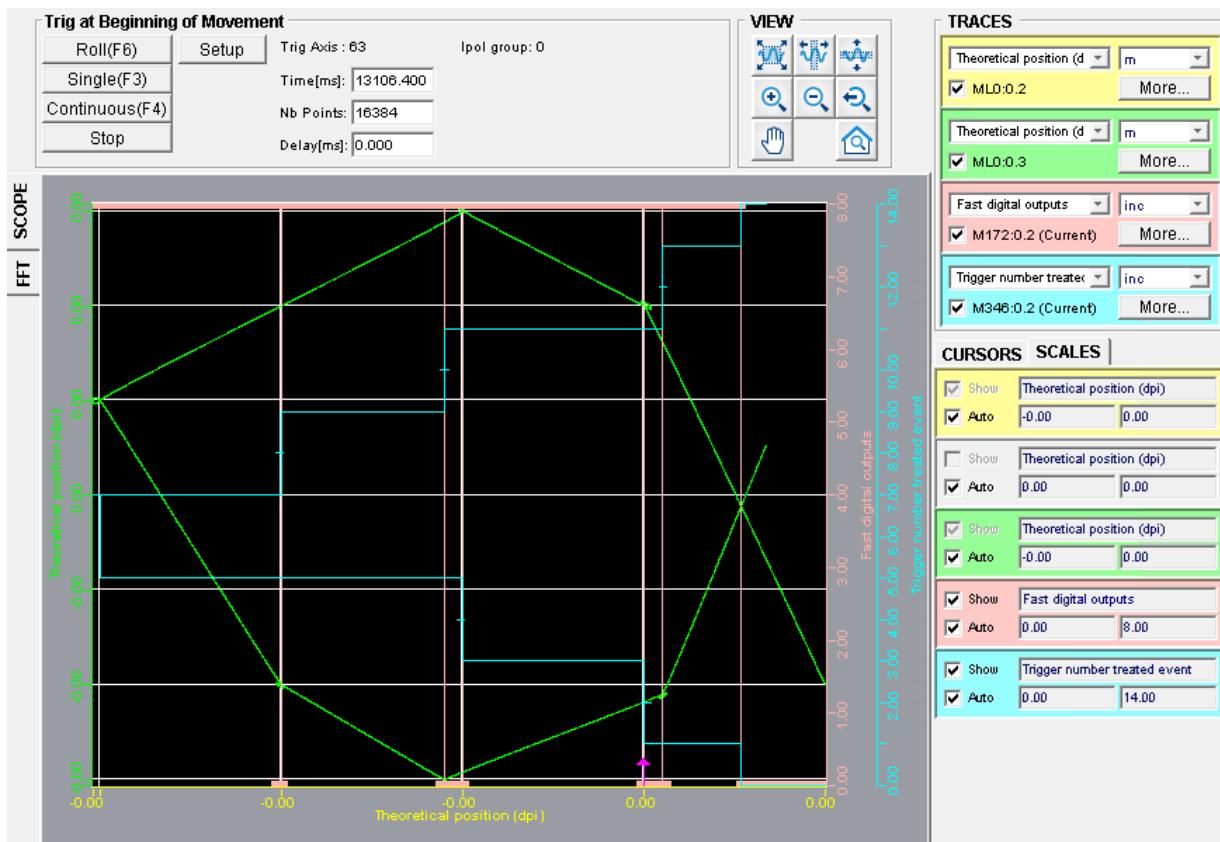
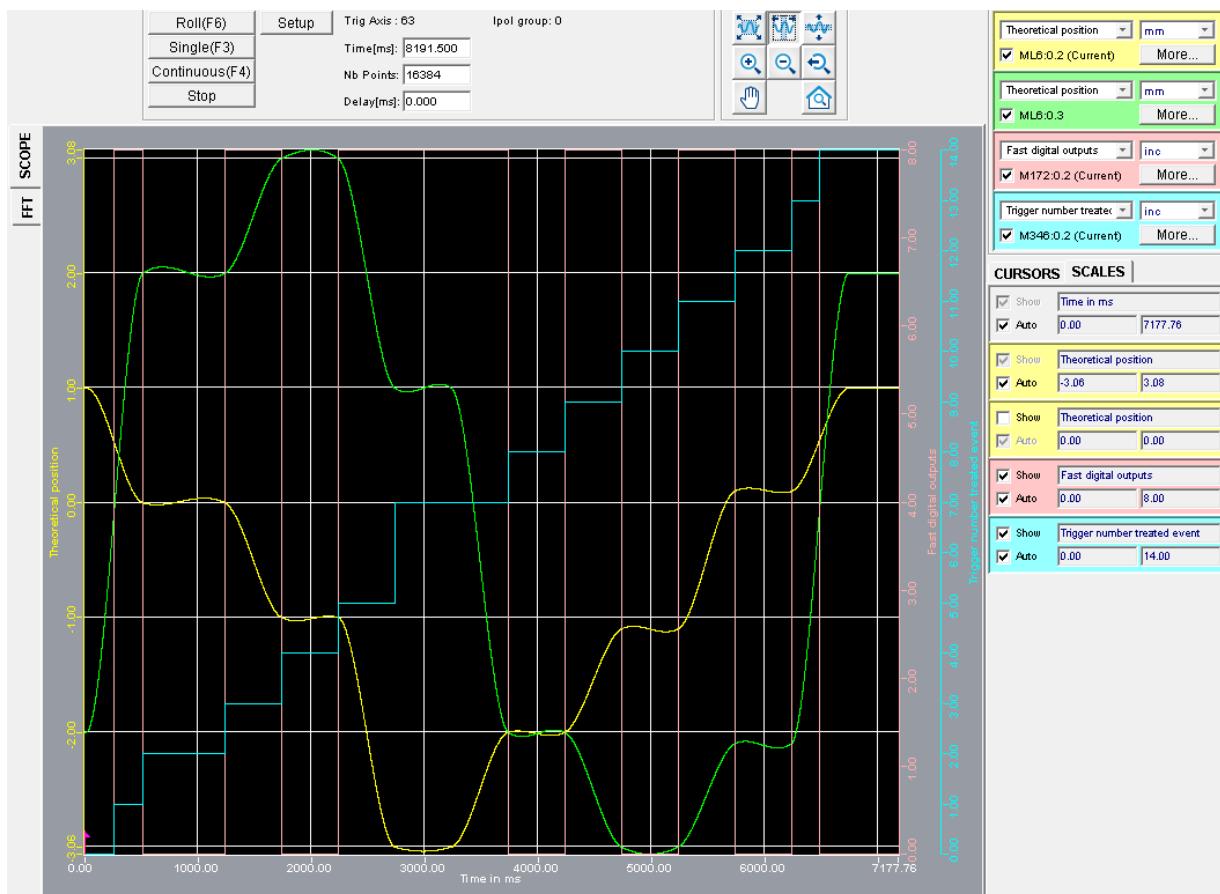
    // Clear trigger combos DOUT masks
    K320.(AX0, AX1) = 0;K321.(AX0, AX1) = 0;K322.(AX0, AX1) = 0;K323.(AX0, AX1)
= 0;K324.(AX0, AX1) = 0;
    K325.(AX0, AX1) = 0;K326.(AX0, AX1) = 0; K327.(AX0, AX1) = 0;K328.(AX0,
AX1) = 0; K329.(AX0, AX1) = 0;
    K330.(AX0, AX1) = 0; K331.(AX0, AX1) = 0;K332.(AX0, AX1) = 0;K333.(AX0,
AX1) = 0;K334.(AX0, AX1) = 0;
    K335.(AX0, AX1) = 0;

    // Clear trigger combos FDOUT masks
    K320:1.(AX0, AX1) = 0; K321:1.(AX0, AX1) = 0; K322:1.(AX0, AX1) = 0;
    K323:1.(AX0, AX1) = 0; K324:1.(AX0, AX1) = 0;
    K325:1.(AX0, AX1) = 0; K326:1.(AX0, AX1) = 0; K327:1.(AX0, AX1) = 0;
    K328:1.(AX0, AX1) = 0; K329:1.(AX0, AX1) = 0;
    K330:1.(AX0, AX1) = 0; K331:1.(AX0, AX1) = 0; K332:1.(AX0, AX1) = 0;
    K333:1.(AX0, AX1) = 0; K334:1.(AX0, AX1) = 0;
    K335:1.(AX0, AX1) = 0;

    // Clear trigger combos Users status masks
    K320:7.(AX0, AX1) = 0; K321:7.(AX0, AX1) = 0; K322:7.(AX0, AX1) = 0;
    K323:7.(AX0, AX1) = 0; K324:7.(AX0, AX1) = 0;
    K325:7.(AX0, AX1) = 0; K326:7.(AX0, AX1) = 0; K327:7.(AX0, AX1) = 0;
    K328:7.(AX0, AX1) = 0; K329:7.(AX0, AX1) = 0;
}

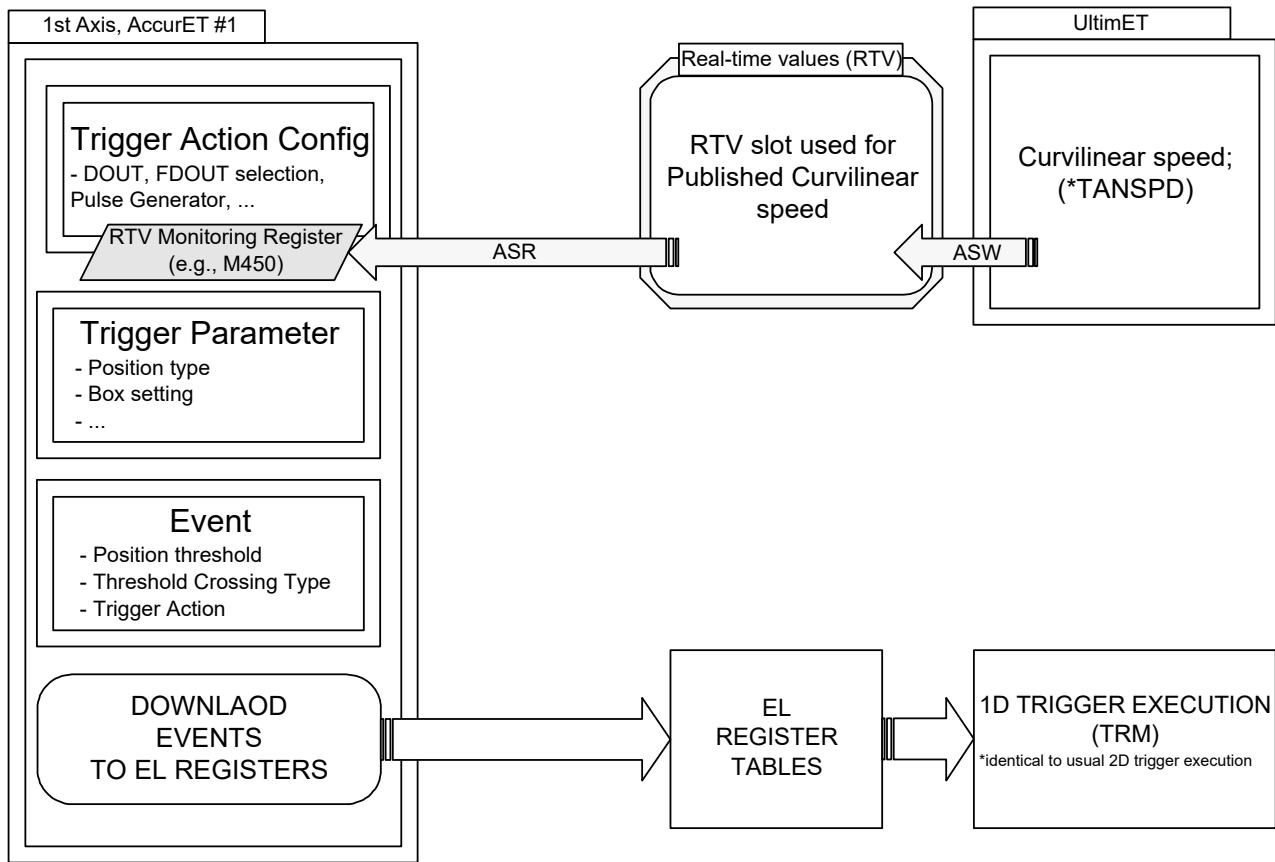
```

```
K330:7.(AX0, AX1) = 0; K331:7.(AX0, AX1) = 0; K332:7.(AX0, AX1) = 0;  
K333:7.(AX0, AX1) = 0; K334:7.(AX0, AX1) = 0;  
K335:7.(AX0, AX1) = 0;  
  
// Clear Pulse Generator output masks  
K340.(AX0, AX1) = 0;// PG1 mask DOUT  
K340:1.(AX0, AX1) = 0;// PG1 mask FDOUT  
K341.(AX0, AX1) = 0;// PG2 mask DOUT  
K341:1.(AX0, AX1) = 0;// PG2 mask FDOUT  
  
// Reset Pulse Generator settings (fixed timing) of controller  
K342.(AX0, AX1) = 0;// PG1 delay  
K342:1.(AX0, AX1) = 0;// PG2 delay  
K343.(AX0, AX1) = 0;// PG1 width  
K343:1.(AX0, AX1) = 0;// PG2 width  
K344.(AX0, AX1) = 0;// PG1 interval  
K344:1.(AX0, AX1) = 0;// PG2 interval  
K345.(AX0, AX1) = 0;// PG1 pulse count  
K345:1.(AX0, AX1) = 0;// PG2 pulse count  
K339.(AX0, AX1) = 0;// PG1 periodicity  
K339:1.(AX0, AX1) = 0;// PG2 periodicity  
  
// Reset Pulse Generator settings (modulated timing) of controller  
KF339.(AX0, AX1) = 0f;// PG1 period modulation coefficient  
KF339:1.(AX0, AX1) = 0f;// PG2 period modulation coefficient  
KF342.(AX0, AX1) = 0f;// PG1 delay modulation coefficient  
KF342:1.(AX0, AX1) = 0f;// PG2 delay modulation coefficient  
KF343.(AX0, AX1) = 0f;// PG1 width modulation coefficient  
KF343:1.(AX0, AX1) = 0f;// PG2 width modulation coefficient  
K353.(AX0, AX1)= 2;// PG1 trigger external reference type  
K353:1.(AX0, AX1)= 2;// PG2 trigger external reference type  
K354.(AX0, AX1)= 0;// PG1 trigger external reference index  
K354:1.(AX0, AX1)= 0;// PG2 trigger external reference index  
K355.(AX0, AX1)= 0;// PG1 trigger external reference depth  
K355:1.(AX0, AX1)= 0;// PG2 trigger external reference depth  
K356.(AX0, AX1)= 0;// PG1 trigger external reference axis  
K356:1.(AX0, AX1)= 0;// PG2 trigger external reference axis  
  
// Clears EL tables in RAM memory  
NEW.(AX0,AX1) = 8;  
  
if ( (M60.AX0 & 0x1) == 0 )PWR.AX0=1;  
if ( (M60.AX1 & 0x1) == 0 )PWR.AX1=1;  
}  
  
void main(void) {  
    reset_all_trigger_settings();  
    triggerConfig();  
    eventTableConfig();  
    moveTest();  
}  
  
void func123(void) {main();}
```



8.9.9.3 Pulse modulated

This example shows how to setup pulse generated with modulated timing combined with the 1D Fast Triggers. PG1 is configured with a period modulated by the curvilinear speed in order to generate a pulse each 2mm.



Example:

```

# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: deg, deg/s, deg/s2, V, A, s
#define AX0 2
#define AX1 3
#define FDOUT_PulseMask 0x4

int slotRTV_Master_M680;

void setReadRTV(void)
{
    slotRTV_Master_M680 = get_32bitDataSlot();
    *ASW = *M680, slotRTV_Master_M680;      // Real-time sharing of current
                                                // interpolation tangential speed
                                                // calculated by UltimET via TransNET.
                                                // (curvilinear speed; *TANSPD)

    ASR.AX0 = M450.AX0, slotRTV_Master_M680;
    ASR.AX1 = M450.AX1, slotRTV_Master_M680;
}

void stopReadRTV(void)
{
    ASR.AX0 = M450.AX0, -1;
    ASR.AX1 = M450.AX1, -1;
    *ASW = 0, slotRTV_Master_M680;           // stop the real-time sharing
}

```

```

        free_32bitDataSlot(slotRTV_Master_M680);
    }

void TriggerSettings(void) {
    float delta_p;
    float speed_conv;

    C359:1.AX0 = FDOUT_PulseMask; // FDOUT4 is used for pulse generator 1
    TRR.AX0 = 0,FDOUT_PulseMask;
    K336.(AX0,AX1)=1;           // Trigger type real or theo.
    // Combi
    K320:1.AX0=0;              // Combi 0 Disabled
    K321:1.AX0=0;              // Combi 1 Disabled
    // PG1 : Every fixed distance over a curvilinear axis whatever the speed is.
    K340:1.AX0 = FDOUT_PulseMask; // <mask of FDOUT>
    K340:0.AX0 = 0;             // <mask of DOUT>

    K342:0.AX0 = 0;             // delay: no fixed time delay before pulse(s)
    KF342:0.AX0 = 0.0f;         // no variable delay before pulse(s)

    K343:0.0 = 10000;           // 10000*10ns=100us fixed time width pulse
    KF343:0.AX0 = 0.0f;         // variable width pulse is disabled
    K344:0.AX0 = 0.0;           // no fixed interval specified
    K339:0.AX0 = 0.0;           // no fixed period time specified
    delta_p = 2.0E-3f;          // trigger on each 2 mm
    speed_conv = 1E3f;          // corresponds to the number of increment of
    speed in source register for
                                // 1 m/s (or 1t/s) (inverse of *k526)
    KF339:0.AX0 = 1.0f / (delta_p * speed_conv); // coefficient that specify
    the delta_P between two rising edges
                                // of pulse
    K345:0.AX0 = 0;             // number of pulses. 0 Means infinity of pulses
    // pointer on M450 (realtime of TransnET that contains the curvilinear
    speed)
    K353:0.AX0 = 3;             // Register source type
    K354:0.AX0 = 450;           // Register source index
    K355:0.AX0 = 0;             // Register source depth
    K356:0.AX0 = 0;             // Register source axis (0 first axis, 1
    second axis)
}

void EventTables(void)
{
    EL0.AX0=lconv(-0.009, _upi, AX0), 1L ,0L ,1L; // Start PG1
    EL1.AX0=lconv(0.0111, _upi, AX0), 1L ,1L ,2L; // Stop PG1
}

void TestMotion(void) {

    SPD.(AX0,AX1)=0.05;
    ACC.(AX0,AX1)=0.1;
    JRT.(AX0,AX1)=0.02;

    MVE.(AX0,AX1)==-0.008;
    WTM.(AX0,AX1);
    *WTT=0.5;
}

```

```
*ISET=0,4L,8L,0L,0L;      // Axis 0(Mask 1L, Upper) is used as "X", Axis
1(Mask 2L, Lower) is used as "Y"
*K522 = 1;                // position value gain by : 1
*K523 = -3;               // power of ten for the position, -6 means 1
micrometer
*K524 = -3;               // power of ten for the speed, -6 means 1 micrometer
/ second
*K525 = -3;               // power of ten for the acceleration, -6 means 1
micrometer / second squared
*K526 = -3;               // power of ten for the time, -6 means 0.001
millisecond

*IBEGIN=0;

*ITSPD=0,500;             // tangential speed of 0.5 m/s
*ITACC=0,5000;            // tangential acceleration of 5 m/s2
*ITDEC=0,5000;            // tangential deceleration of 5 m/s2
*ITJRT=0,10;              // tangential jert time = 10 ms

*ILINE=0,-2,-2,0,0;
*WTM;

TRM.AX0=0,2;

*ILINE=0,20,20,0,0;
*WTM;

*WTT=0.1;
*ILINE=0,-2,-2,0,0;
*WTM;

TRM.AX0=-1;

*IEND=0;
}

void reset_all_pg(void)
{
    TRM.AX0 = -1;
    TRM.AX1 = -1;
    SPG.AX0 = 1,0;
    SPG.AX0 = 2,0;
    C359.AX0 = 0, 0;
    K340.AX0 = 0, 0;
    K341.AX0 = 0, 0;
    K340.AX1 = 0, 0;
    K341.AX1 = 0, 0;
    KF339.AX0 = 0f, 0f;
    KF342.AX0 = 0f, 0f;
    KF343.AX0 = 0f, 0f;
    K339.AX0 = 0, 0;
    K342.AX0 = 0, 0;
    K343.AX0 = 0, 0;
    K344.AX0 = 0, 0;
    K345.AX0 = 0,0;
    K353.AX0 = 0,0;
```

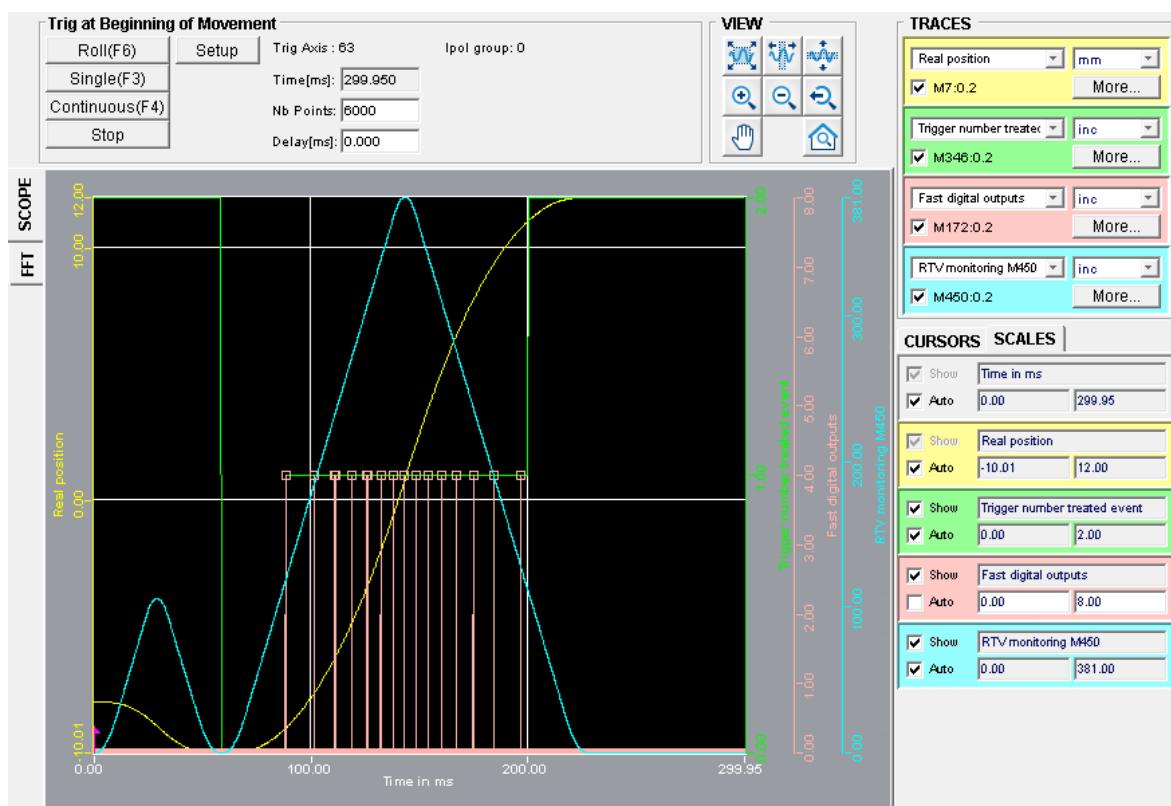
```

K354.AX0 = 0,0;
K355.AX0 = 0,0;
K356.AX0 = 0,0;
KF339.AX1 = 0f, 0f;
KF342.AX1 = 0f, 0f;
KF343.AX1 = 0f, 0f;
K339.AX1 = 0, 0;
K342.AX1 = 0, 0;
K343.AX1 = 0, 0;
K344.AX1 = 0, 0;
K345.AX1 = 0,0;
K353.AX1 = 0,0;
K354.AX1 = 0,0;
K355.AX1 = 0,0;
K356.AX1 = 0,0;

NEW.AX0 = 8;      // Clears EL tables in RAM memory
NEW.AX1 = 8;
if ( (M60.AX0 & 0x1) == 0 )
    PWR.AX0=1;
if ( (M60.AX1 & 0x1) == 0 )
    PWR.AX1=1;
}

void main(void) {
    /reset_all_pg();
    setReadRTV();
    EventTables();
    TriggerSettings();
    TestMotion();
    stopReadRTV();
}
void func123(void) { main(); }

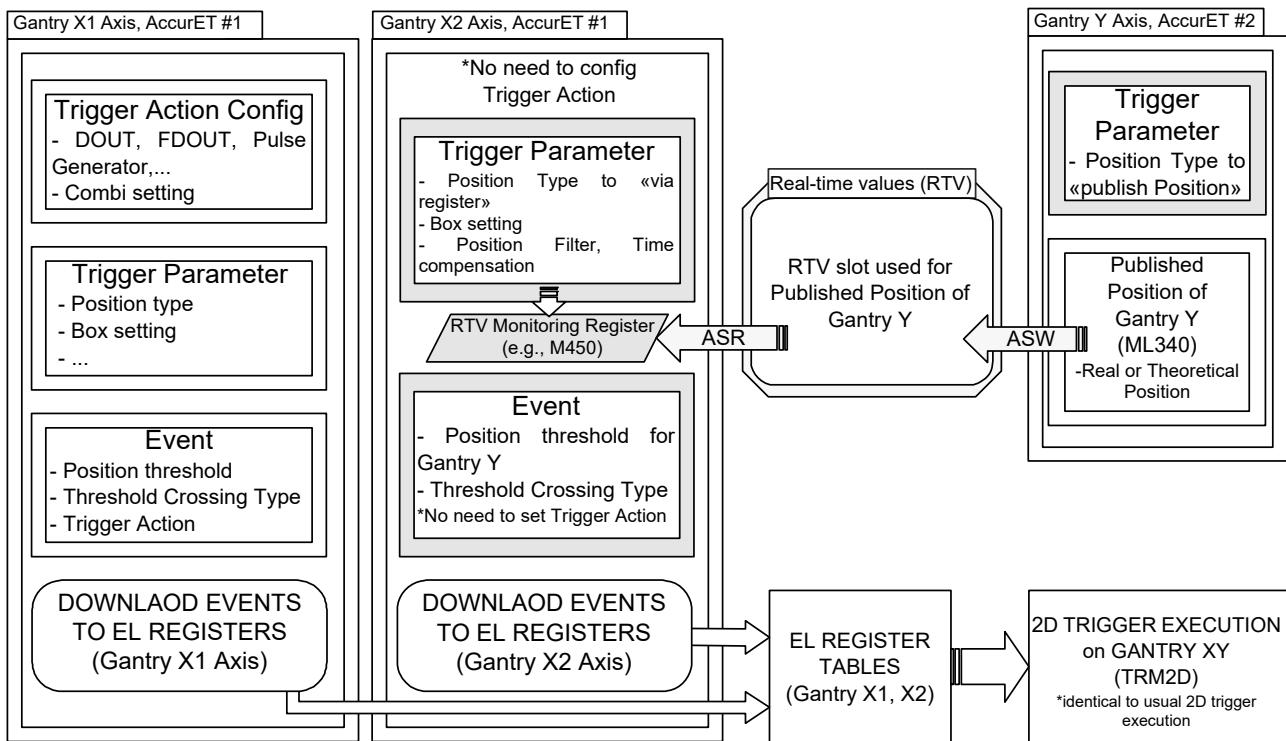
```



8.9.9.4 Position source coming from any register

This example shows how to use “Position source coming from any register” feature to enable 2D Fast Triggers on XY Gantry system in Gantry level 1 (refer to [§12](#) for more information).

- The gantry system has two motors for X axis (X1, X2)
- Y axis position information is sent to X2 axis (2nd axis of AccurET#1) for implementing the 2D Fast Triggers.



Example:

```
# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: t, t/s, t/s2, V, A, s
// ****
// Define
// ****
#define AXIS_X1 0
#define AXIS_X2 1
#define AXIS_Y 2
#define AXIS3 3
#define AXISMASK_X1X2 3I

#define SPEED1.0

// Target positions (Coordinates of points A, B, C, D are a rectangle)
#define A_X 0.00
#define A_Y 0.00
#define B_X 0.05
#define B_Y 0.00
#define C_X 0.05
#define C_Y -0.05
#define D_X 0.0
#define D_Y -0.05
```

```

// Constants for Trigger types: Used into EL table on depth 1
#define TRIG_X_AND_Y 0x4
#define TRIG_POS 0x1
#define TRIG_NEG 0x2
#define TRIG_AX_DISABLED 0x3
#define TRIG_AY_DISABLED 0x3
#define TRIG_COMBO 0L
#define TRIG_NOPG 0L

//
*****Global variables*****
//
*****Coordinates of Triggers (situated in the middle of specified segments)*****
long AB_X;
long AB_Y;
long BC_X;
long BC_Y;
long CD_X;
long CD_Y;
long DA_X;
long DA_Y;
// For RTV slots
int slot_trig_pos_lsl;
int slot_trig_pos_msl;
/***
*
* Motor Power ON, and homing, initial position
*
*/
void init_motor(void)
{
    pwr.(AXIS_X1,AXIS_X2) = 1;
    pwr.AXIS_Y = 1;
    if ( (M60.AXIS_X1 & 0x4) == 0 ) {
        *IND=AXISMASK_X1X2;WTM.(AXIS_X1,AXIS_X2); // Not necessarily for EnDat Abs.
        Encoder
    }
    if ( (M60.AXIS_Y & 0x4) == 0 ) {
        IND.AXIS_Y;WTM.AXIS_Y; // Not necessarily for EnDat Abs. Encoder
    }

    spd.(AXIS_X1,AXIS_X2)=SPEED;
    acc.(AXIS_X1,AXIS_X2)=2.0;
    jrt.(AXIS_X1,AXIS_X2)=0;

    spd.AXIS_Y=SPEED;
    acc.AXIS_Y=2.0;
    JRT.AXIS_Y=0;

    MVE.(AXIS_X1,AXIS_X2) = A_X;
    MVE.AXIS_Y = A_Y;
    WTM.(AXIS_X1,AXIS_X2);
    WTM.AXIS_Y;
}

```

```
}

/***
 *
 * Function that makes a sequence of movements to draw a rectangle
 *
 *
 */

void move_a_rectangle(void)
{
    MVE.(AXIS_X1,AXIS_X2) = B_X;
    MVE.AXIS_Y = B_Y;
    WTM.(AXIS_X1,AXIS_X2);
    WTM.AXIS_Y;
    MVE.(AXIS_X1,AXIS_X2) = C_X;
    MVE.AXIS_Y = C_Y;
    WTM.(AXIS_X1,AXIS_X2);
    WTM.AXIS_Y;
    MVE.(AXIS_X1,AXIS_X2) = D_X;
    MVE.AXIS_Y = D_Y;
    WTM.(AXIS_X1,AXIS_X2);
    WTM.AXIS_Y;
    MVE.(AXIS_X1,AXIS_X2) = A_X;
    MVE.AXIS_Y = DA_Y;
    WTM.(AXIS_X1,AXIS_X2);
    WTM.AXIS_Y;

}

/***
 *
 * Trigger initialisation
 *
 */

void config_2dtrigger(void)
{
// Calculate the coordinates in [UPI] of Trigger targets. The target are placed
in the middle of each side of the rectangle
    AB_X = (lconv(A_X, _upi, AXIS_X1) + lconv(B_X, _upi, AXIS_X1)) / 2L;
    AB_Y = (lconv(A_Y, _upi, AXIS_Y) + lconv(B_Y, _upi, AXIS_Y)) / 2L;
    BC_X = (lconv(B_X, _upi, AXIS_X1) + lconv(C_X, _upi, AXIS_X1)) / 2L;
    BC_Y = (lconv(B_Y, _upi, AXIS_Y) + lconv(C_Y, _upi, AXIS_Y)) / 2L;
    CD_X = (lconv(C_X, _upi, AXIS_X1) + lconv(D_X, _upi, AXIS_X1)) / 2L;
    CD_Y = (lconv(C_Y, _upi, AXIS_Y) + lconv(D_Y, _upi, AXIS_Y)) / 2L;
    DA_X = (lconv(D_X, _upi, AXIS_X1) + lconv(A_X, _upi, AXIS_X1)) / 2L;
    DA_Y = (lconv(D_Y, _upi, AXIS_Y) + lconv(A_Y, _upi, AXIS_Y)) / 2L;

// Mask DOUT/FDOUT for triggers
    k359.AXIS_X1 = 0x00000003; // DOUT1 is used for trigger
    C359:1.AXIS_X1 = 0x00000003; // FDOUT1 is used for trigger
// ##### AXIS 0 (Gantry X) #####
// Position type selection
    K336.AXIS_X1 = 1;
// Combi configuration
    K320.AXIS_X1 = 0x00010001; // Toggle DOUT1
    K320:1.AXIS_X1 = 0x00010001; // Toggle FDOUT1
// Event programming
    EL0.AXIS_X1 = AB_X, TRIG_X_AND_Y | TRIG_POS,TRIG_COMBIO,TRIG_NOPG;
```

```

EL1.AXIS_X1 = BC_X, TRIG_X_AND_Y | TRIG_AX_DISABLED,TRIG_COMBIO,TRIG_NOPG;
EL2.AXIS_X1 = CD_X, TRIG_X_AND_Y | TRIG_NEG,TRIG_COMBIO,TRIG_NOPG;
EL3.AXIS_X1 = DA_X, TRIG_X_AND_Y | TRIG_AX_DISABLED,TRIG_COMBIO,TRIG_NOPG;

// ##### AXIS 1 #####
// Position type selection
K336.AXIS_X2 = 2;// Position source: M450 RTV monitoring register

K353:2.AXIS_X2 = 67;
K354:2.AXIS_X2 = 450;
K355:2.AXIS_X2 = 0;
K356:2.AXIS_X2 = 0;

K360.AXIS_X2 = 2;// Position filter selection (ML0)
KF360.AXIS_X2 = 250e-6;// Time compensation (TransnET)
// Event for Axis 1
EL0.AXIS_X2 = AB_Y, TRIG_AY_DISABLED,TRIG_COMBIO,TRIG_NOPG;
EL1.AXIS_X2 = BC_Y, TRIG_NEG,TRIG_COMBIO,TRIG_NOPG;
EL2.AXIS_X2 = CD_Y, TRIG_AY_DISABLED,TRIG_COMBIO,TRIG_NOPG;
EL3.AXIS_X2 = DA_Y, TRIG_POS,TRIG_COMBIO,TRIG_NOPG;

// ##### AXIS 2 (Gantry Y)#####
K336.AXIS_Y= 4; // Publish theoretical position for trigger into M340
}

void clear_config_2d_trigger(void)
{
    // Force DOUT FDOUT to 0
    TRR.AXIS_X1 = 0x00000001,0x00000001;
    //clear output reservation
    C359:1.AXIS_X1 = 0;
    k359.AXIS_X1 = 0;
    k359:1.AXIS_X1 = 0;
    // Clear setting of position for trigger
    K336.AXIS_X1 = 0;
    K336.AXIS_X2 = 0;
    // Clear Combi configuration
    K320.AXIS_X1 = 0x0;
    K320:1.AXIS_X1 = 0x0;

    K336.AXIS_X2 = 0;
    K353:2.AXIS_X2 = 0;
    K354:2.AXIS_X2 = 0;
    K355:2.AXIS_X2 = 0;
    K356:2.AXIS_X2 = 0;
    K360.AXIS_X2 = 0;
    KF360.AXIS_X2 = 0;

    for(K198.0=0; K198.0<8; K198.0++)
    {
        ELY.AXIS_X1 = 0L,0L,0L,0L,0L,0L;
        ELY.AXIS_X2 = 0L,0L,0L,0L,0L,0L;
    }
}
//*****

```

```
// Real time channel configuration
//*****
void conf_rtv_trigger(void)
{
    //#####
    //RTV used for trigger from external position
    //#####
    // Get free slot
    slot_trig_pos_lsl = get_32bitDataSlot();
    slot_trig_pos_msl = get_32bitDataSlot();
    // Assign slot
    ASW.AXIS_Y = ML340, slot_trig_pos_lsl, slot_trig_pos_msl;
    ASR.AXIS_X2 = ML450, slot_trig_pos_lsl, slot_trig_pos_msl;
}

void clear_rtv_slot(void)
{
    // Stop sharing register on transnET
    ASR.AXIS_X2 = ML450, -1, -1;
    ASW.AXIS_Y = 0, slot_trig_pos_lsl , slot_trig_pos_msl;
    // free slot
    free_32bitDataSlot(slot_trig_pos_lsl);
    free_32bitDataSlot(slot_trig_pos_msl);
}

void func123(void)
{
    init_motor();

    *wtt = 1.0;

    config_2dtrigger();

    conf_rtv_trigger();

    x1.AXIS_X1= 1;

    TRM2D.AXIS_X1 = 0, 4; //Enable 2D trigger with missed event
    move_a_rectangle();

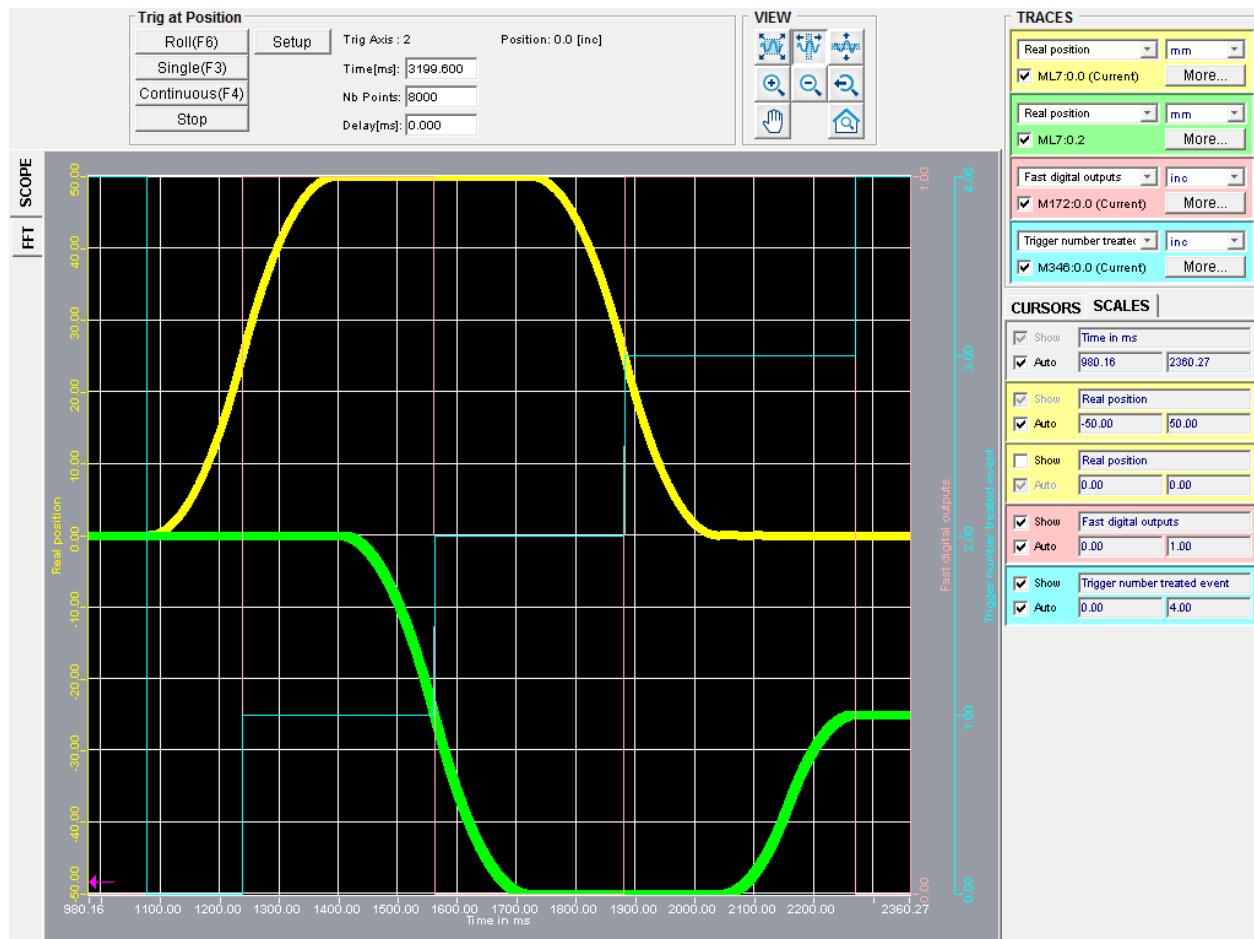
    *wtt= 2.0;

    x1.AXIS_X1= 0;
    pwr.(AXIS_X1,AXIS_X2)=0;
    pwr.AXIS_Y=0;

    // rst.AXIS_X1;
    // rst.AXIS_X2;

    *wtt = 5.0;

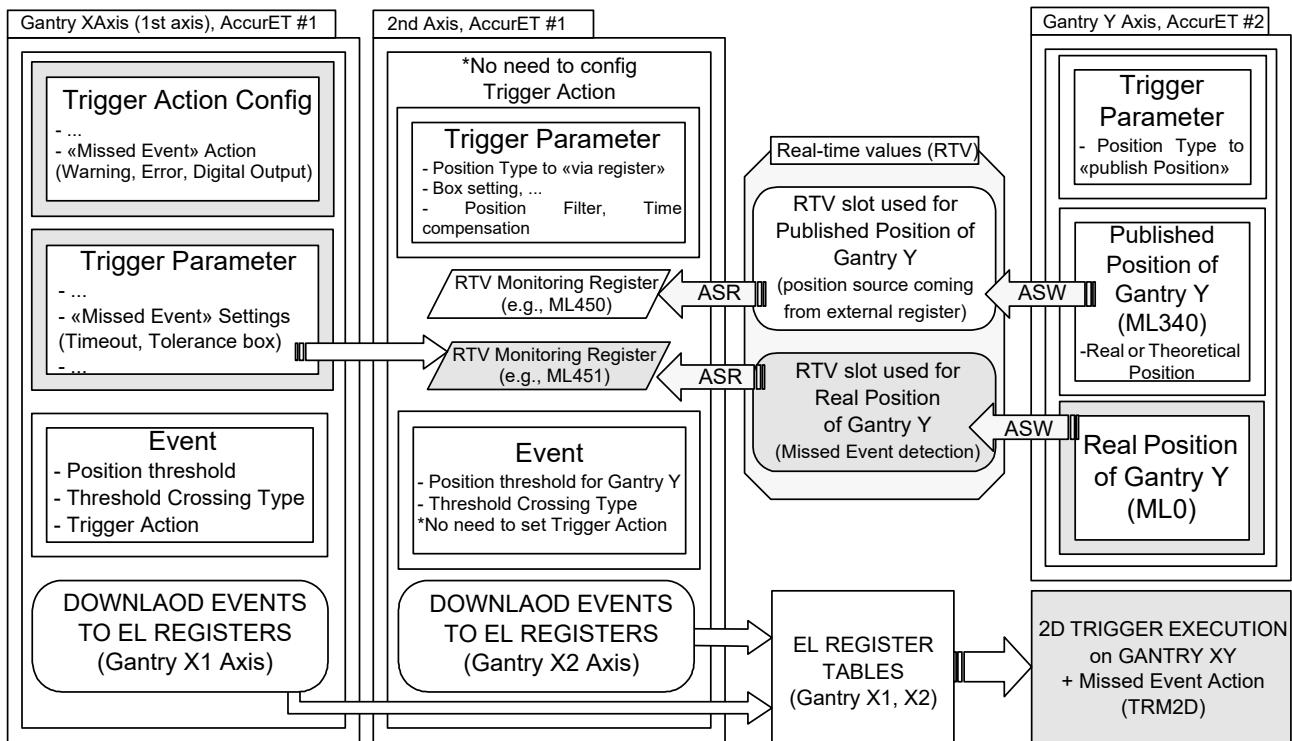
    clear_rtv_slot();
    clear_config_2d_trigger();
}
```



8.9.9.5 Position source coming from any register with missed event

This example shows how to use “Position source coming from any register” feature to enable 2D Fast Triggers on XY Gantry system in Gantry level 2 (refer to [§12](#) for more information).

- The gantry system has two motors for X axis but only the first motor axis is visible as “gantry X” axis. (1st axis of AccurET#1)
- Y axis position information is sent to Axis 1 (2nd axis of AccurET#1) for implementing the 2D Fast Triggers.



Example:

```

# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: t, t/s, t/s2, V, A, s
//*****
// Define
//*****
#define AXIS_X 0
#define AXIS1 1
#define AXIS_Y 2
#define AXIS3 3

#define SPEED 1.0

// Target positions (Coordinates of points A, B, C, D are a rectangle)
#define A_X 0.00
#define A_Y 0.00
#define B_X 0.05
#define B_Y 0.00
#define C_X 0.05
#define C_Y -0.05
#define D_X 0.0
#define D_Y -0.05

// missed event specification

```

```

// A timer is started when the theoretical position enters in the box of missed
event. If this timer is bigger than the timeout (refer to K347), an error or a
warning is generated.
#define MEVT_TIMEOUT400 // If D_Y is 0.15 instead of 0.20, the speed is decreasing
at the end of this position, and the error (warning) could raise even inside the
target range. (i.e., the timeout could be not enough.)
#define MEVT_TOLERANCE_BOX0.4E-3 // Speed * MLTI duration

// Constants for Trigger types: Used into EL table on depth 1
#define TRIG_X_AND_Y 0x4
#define TRIG_POS 0x1
#define TRIG_NEG 0x2
#define TRIG_AX_DISABLED 0x3
#define TRIG_AY_DISABLED 0x3
#define TRIG_COMBIO 0L
#define TRIG_NOPG 0L
//*********************************************************************
// Global variables
//*********************************************************************
//Coordinates of Triggers (situated in the middle of specified segments)
long AB_X;
long AB_Y;
long BC_X;
long BC_Y;
long CD_X;
long CD_Y;
long DA_X;
long DA_Y;
// For RTV slots
int slot_trig_pos_lsl;
int slot_trig_pos_msl;
int slot_trig_missed_pos_lsl;
int slot_trig_missed_pos_msl;
/***
 *
 * Motor Power ON, and homing, initial position
 *
 */
void init_motor(void)
{
    pwr.AXIS_X = 1;
    pwr.AXIS_Y = 1;
    if ( (M60.AXIS_X & 0x4) == 0 ) {
        IND.AXIS_X; WTM.AXIS_X;// Not necessarily for EnDat Abs. Encoder
    }
    if ( (M60.AXIS_Y & 0x4) == 0 ) {
        IND.AXIS_Y; WTM.AXIS_Y;// Not necessarily for EnDat Abs. Encoder
    }

    spd.AXIS_X=SPEED;
    acc.AXIS_X=2.0;
    JRT.AXIS_X=0;

    spd.AXIS_Y=SPEED;
    acc.AXIS_Y=2.0;
}

```

```
JRT.AXIS_Y=0;

MVE.AXIS_X = A_X;
MVE.AXIS_Y = A_Y;
WTM.AXIS_X;
WTM.AXIS_Y;
}

/**
*
* Function that makes a sequence of movements to draw a rectangle
*
*
*/
void move_a_rectangle(void)
{
    MVE.AXIS_X = B_X;
    MVE.AXIS_Y = B_Y;
    WTM.AXIS_X;
    WTM.AXIS_Y;
    MVE.AXIS_X = C_X;
    MVE.AXIS_Y = C_Y;
    WTM.AXIS_X;
    WTM.AXIS_Y;
    MVE.AXIS_X = D_X;
    MVE.AXIS_Y = D_Y;
    WTM.AXIS_X;
    WTM.AXIS_Y;
    MVE.AXIS_X = A_X;
    //MVE.AXIS_Y = DA_Y;
    MVE.AXIS_Y = DA_Y - lconv(MEVT_TOLERANCE_BOX, _upi, AXIS_Y);
    WTM.AXIS_X;
    WTM.AXIS_Y;
}

/**
*
* Trigger initialization
*
*/
void config_2dtrigger(void)
{
// Calculate the coordinates in [UPI] of Trigger targets. The target are placed
in the middle of each side of the rectangle
    AB_X = (lconv(A_X, _upi, AXIS_X) + lconv(B_X, _upi, AXIS_X)) / 2L;
    AB_Y = (lconv(A_Y, _upi, AXIS_Y) + lconv(B_Y, _upi, AXIS_Y)) / 2L;
    BC_X = (lconv(B_X, _upi, AXIS_X) + lconv(C_X, _upi, AXIS_X)) / 2L;
    BC_Y = (lconv(B_Y, _upi, AXIS_Y) + lconv(C_Y, _upi, AXIS_Y)) / 2L;
    CD_X = (lconv(C_X, _upi, AXIS_X) + lconv(D_X, _upi, AXIS_X)) / 2L;
    CD_Y = (lconv(C_Y, _upi, AXIS_Y) + lconv(D_Y, _upi, AXIS_Y)) / 2L;
    DA_X = (lconv(D_X, _upi, AXIS_X) + lconv(A_X, _upi, AXIS_X)) / 2L;
    DA_Y = (lconv(D_Y, _upi, AXIS_Y) + lconv(A_Y, _upi, AXIS_Y)) / 2L;

    // Mask DOUT/FDOUT for triggers
    k359.AXIS_X = 0x00000003; // DOUT1 is used for trigger
    C359:1.AXIS_X = 0x00000003; // FDOUT1 is used for trigger
```

```

// ##### AXIS 0 (Gantry X) #####
// Position type selection
K336.AXIS_X = 0;
// Combi configuration
K320.AXIS_X = 0x00010001; // Toggle DOUT1
K320:1.AXIS_X = 0x00010001; // Toggle FDOUT1
// Event programming
EL0.AXIS_X = AB_X, TRIG_X_AND_Y | TRIG_POS,TRIG_COMBIO,TRIG_NOPG;
EL1.AXIS_X = BC_X, TRIG_X_AND_Y | TRIG_AX_DISABLED,TRIG_COMBIO,TRIG_NOPG;
EL2.AXIS_X = CD_X, TRIG_X_AND_Y | TRIG_NEG,TRIG_COMBIO,TRIG_NOPG;
EL3.AXIS_X = DA_X, TRIG_X_AND_Y | TRIG_AX_DISABLED,TRIG_COMBIO,TRIG_NOPG;

// ##### configure output in case of error or missed event #####
// DOUT0 will be set to 0 and DOUT1 will be set to 1 in case of error
k357.AXIS_X = 0x10002;
// FDOUT0 will be set to 0 and FDOUT1 will be set to 1 in case of error
k357:1.AXIS_X = 0x10002;

// ##### AXIS 1 #####
// Position type selection
K336.AXIS1 = 2;// Position source: M450 RTV monitoring register

K353:2.AXIS1 = 67;
K354:2.AXIS1 = 450;
K355:2.AXIS1 = 0;
K356:2.AXIS1 = 0;

K360.AXIS_X2 = 2;// Position filter selection (ML0)
KF360.AXIS_X2 = 250e-6;// Time compensation (TransnET)
// Event for Axis 1
EL0.AXIS1 = AB_Y, TRIG_AY_DISABLED,TRIG_COMBIO,TRIG_NOPG;
EL1.AXIS1 = BC_Y, TRIG_NEG,TRIG_COMBIO,TRIG_NOPG;
EL2.AXIS1 = CD_Y, TRIG_AY_DISABLED,TRIG_COMBIO,TRIG_NOPG;
EL3.AXIS1 = DA_Y, TRIG_POS,TRIG_COMBIO,TRIG_NOPG;

// ##### AXIS 2 (Gantry Y) #####
k336.AXIS_Y= 3; // Publish real position for trigger into M340
}

void clear_config_2d_trigger(void)
{
    // Force DOUT FDOUT to 0
    TRR.AXIS_X = 0x00000001,0x00000001;
    //clear output reservation
    C359:1.AXIS_X = 0;
    k359.AXIS_X = 0;
    k359:1.AXIS_X = 0;
    // Clear setting of position for trigger
    K336.AXIS_X = 0;
    K336.AXIS1 = 0;
    // Clear Combi configuration
    K320.AXIS_X = 0x0;
    K320:1.AXIS_X = 0x0;

    K336.AXIS1 = 0;
}

```

```
K353:2.AXIS1 = 0;
K354:2.AXIS1 = 0;
K355:2.AXIS1 = 0;
K356:2.AXIS1 = 0;
K360.AXIS1 = 0;
KF360.AXIS1 = 0;

for(K198.0=0; K198.0<8; K198.0++)
{
    ELY.AXIS_X = 0L,0L,0L,0L,0L,0L;
    ELY.AXIS1 = 0L,0L,0L,0L,0L,0L;
}
}

//*****
// Real time channel configuration
//*****
void conf_rtv_trigger(void)
{
    //#####
    //RTV used for trigger from external position
    //#####
    // Get free slot
    slot_trig_pos_lsl = get_32bitDataSlot();
    slot_trig_pos_msl = get_32bitDataSlot();
    // Assign slot
    ASW.AXIS_Y = ML340, slot_trig_pos_lsl, slot_trig_pos_msl;
    ASR.AXIS1 = ML450, slot_trig_pos_lsl, slot_trig_pos_msl;
    //#####
    //RTV used for trigger missed event
    //#####
    // Get free slot
    slot_trig_missed_pos_lsl = get_32bitDataSlot();
    slot_trig_missed_pos_msl = get_32bitDataSlot();
    //Assign slot
    ASW.AXIS_Y = ML0, slot_trig_missed_pos_lsl , slot_trig_missed_pos_msl;
    ASR.AXIS1 = ML451, slot_trig_missed_pos_lsl, slot_trig_missed_pos_msl;
}

void clear_rtv_slot(void)
{
    // Stop sharing register on transnET
    ASR.AXIS1 = ML450, -1, -1;
    ASR.AXIS1 = ML451, -1, -1;
    ASW.AXIS_Y = 0, slot_trig_pos_lsl , slot_trig_pos_msl;
    ASW.AXIS_Y = 0, slot_trig_missed_pos_lsl, slot_trig_missed_pos_msl;
    // free slot
    free_32bitDataSlot(slot_trig_pos_lsl);
    free_32bitDataSlot(slot_trig_pos_msl);
    free_32bitDataSlot(slot_trig_missed_pos_lsl);
    free_32bitDataSlot(slot_trig_missed_pos_msl);
}

void conf_missed_event_param(void)
{
```

```

// Define condition to detect a missed event
k347.AXIS_X = MEVT_TIMEOUT;
k348.AXIS_X = lconv(MEVT_TOLERANCE_BOX, _upi, AXIS_X);
k349.AXIS_X = 1; // Warning will be generated in case of missed event
k347.AXIS1 = MEVT_TIMEOUT;
k348.AXIS1 = lconv(MEVT_TOLERANCE_BOX, _upi, AXIS_Y);
k349.AXIS1 = 1; // Warning will be generated in case of missed event

// Define where to get position for missed event
K353:3.AXIS1= 67; // Type for ML is 67
K354:3.AXIS1= 451;// Index
K355:3.AXIS1= 0;
K356:3.AXIS1= 0;
}

void clear_conf_missed_event_param(void)
{
    // Define condition to detect a missed event
    k347.AXIS_X = 0;
    k348.AXIS_X = 0;
    k349.AXIS_X = 0;
    k347.AXIS1 = 0;
    k348.AXIS1 = 0;
    k349.AXIS1 = 0;

    // Define where to get position for missed event
    K353:3.AXIS1= 0;
    K354:3.AXIS1= 0;
    K355:3.AXIS1= 0;
    K356:3.AXIS1= 0;
}

void func123(void)
{
    init_motor();

    *wtt = 1.0;

    config_2dtrigger();

    conf_rtv_trigger();

    conf_missed_event_param();

    x1.AXIS_X= 1;

    TRM2D.AXIS_X = 0, 4, 1; //Enable 2D trigger with missed event
    move_a_rectangle();

    *wtt= 2.0;
    x1.AXIS_X= 0;
    pwr.AXIS_X=0;
    pwr.AXIS_Y=0;

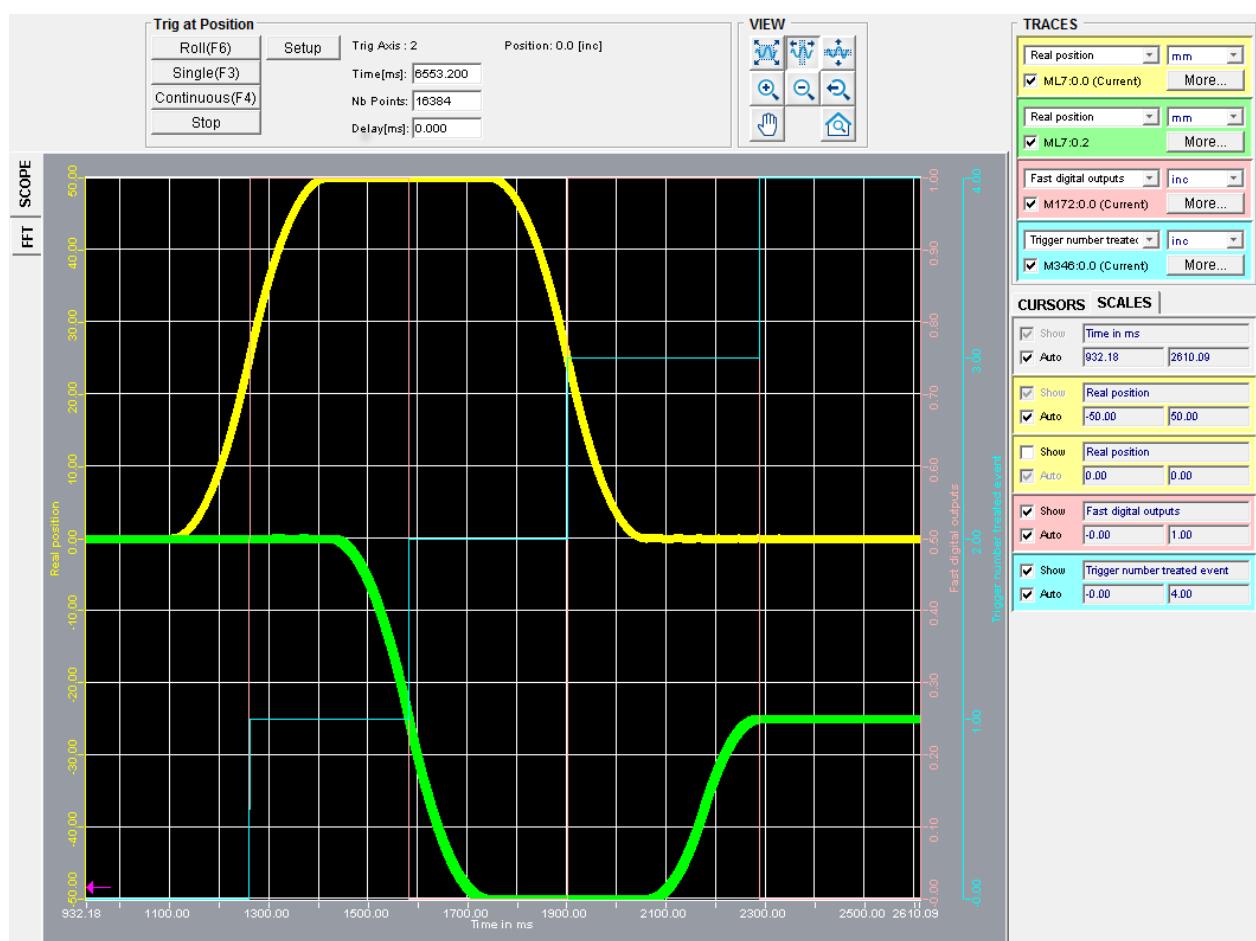
    // rst.AXIS_X;
}

```

```
// rst.AXIS1;

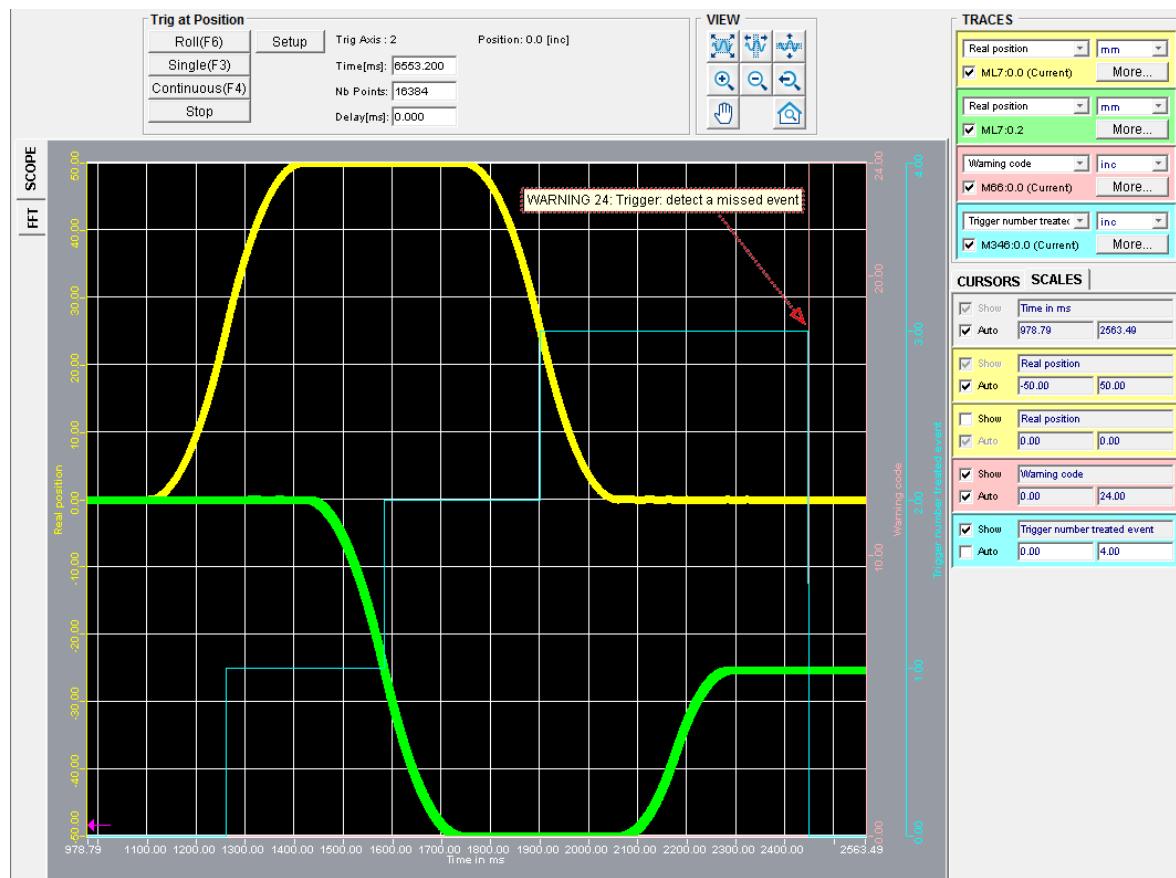
*wtt = 5.0;

clear_rtv_slot();
clear_config_2d_trigger();
clear_conf_missed_event_param();
}
```



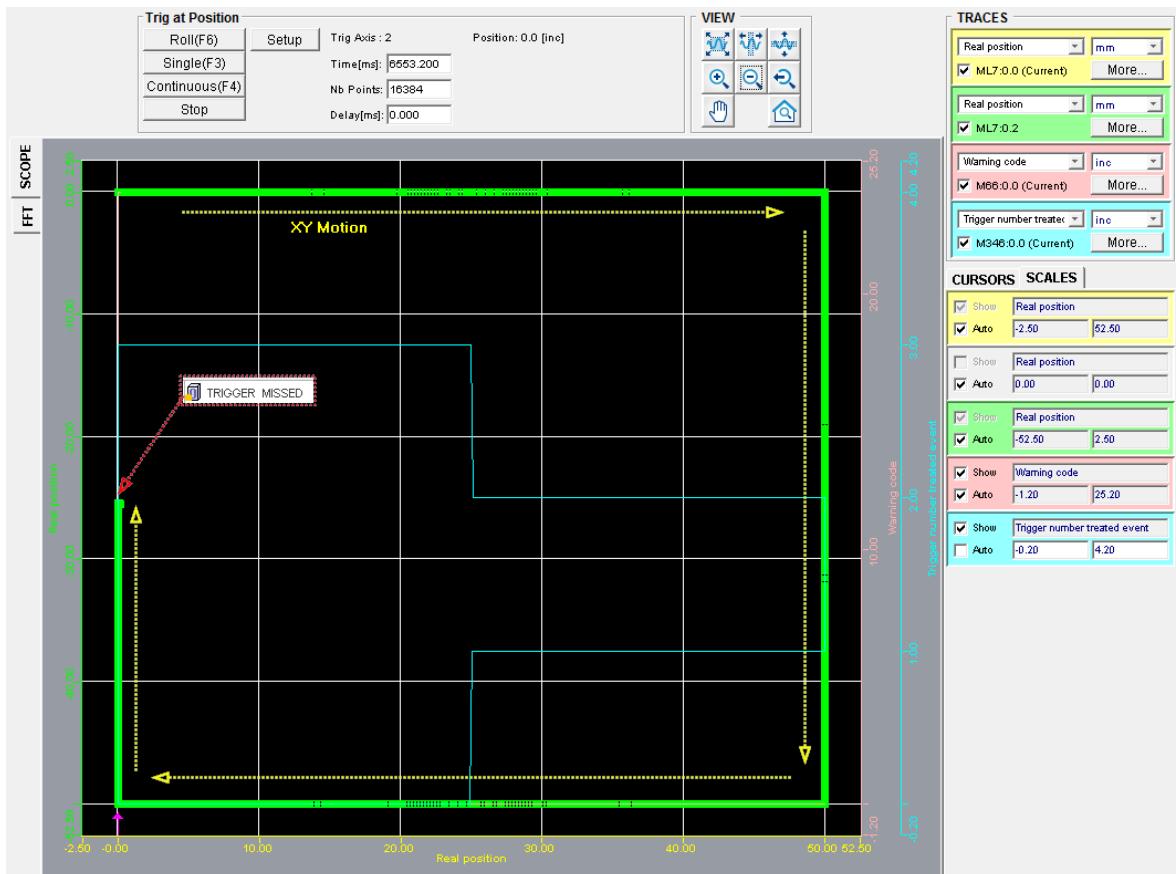
Changing the example sequence as follow will cause the Missed Event issue:

```
... (void move_a_rectangle(void))
MVE.AXIS_Y = DA_Y;
=> MVE.AXIS_Y = DA_Y - lconv(MEVT_TOLERANCE_BOX, _upi, AXIS_Y);
```



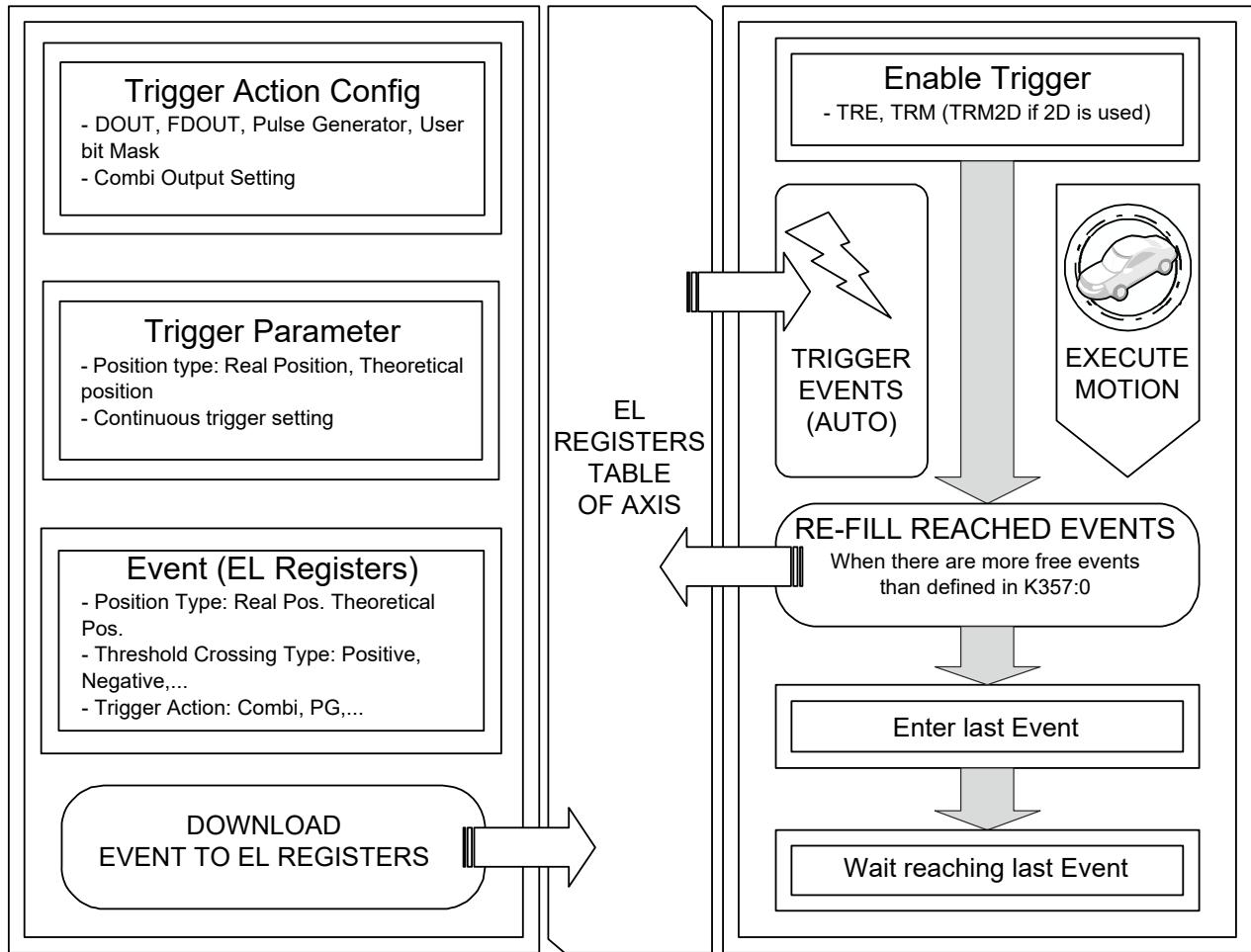
Missed event detection algorithm will generate a warning or error depending on setting.

Check drive status to see if a warning or error was correctly generated.



8.9.9.6 Continuous Fast Triggers

This example shows how to setup 1D Fast Triggers in continuous mode on an axis of AccurET controller. In this example we execute 2304 events.



```

# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: t, t/s, t/s2, V, A, s

#define SPEED 0.75
#define ACCELERATION 50.0

#define BIT_USE_CONTINUOUS_TRIGGER 3
#define TRIGGER_TYPE_POSITIVE_ONLY 0x1

#define NB_TOT_TRIGGER 2304
//*****
// Function who fill the EL table in thread 2
//*****
void fill_el_table(void)
{
    int idx;
    int cmpt;
    int event_tot = 512;

    for(;;)
    {
        x25:1.!= event_tot;

```

```

        if (M63.! & (1 << BIT_USE_CONTINUOUS_TRIGGER))
        { // Check if there are enough free events
            idx = M350.!;
            cmpt = 0;
            while (cmpt < 256)
            {
                if (idx > 512)
                {
                    k198.! = idx - 512;
                }
                else
                {
                    k198.! = idx;
                }
                ELY.! = trigPos;
                trigPost+=lconv(0.0015, _upi);
                idx++;
                cmpt++;
            }
            event_tot += 256;
            if (event_tot >= NB_TOT_TRIGGER)
            {
                trm.!=-255;
                end.! = 2;
            }
        }
        yield();
    }
}

//*****
// Run trigger 1D on 2304
// reset counter of group (M351) is done every 4 events.
//*****
void func0(void)
{
    long combi_num;
    x25:1.0 = 0;

    pwr.!= 1;
    ind.!;
    spd.!=SPEED;
    acc.!=ACCELERATION;

    k177.! &= ~(1 << BIT_USE_CONTINUOUS_TRIGGER);
    // Position type selection
    k336.! = 1;
    // Output reservation (only for FDOUT)
    C359:1.! = 0x00000001;      // FDOUT1
    // Combi configuration
    K320:1.!=0x00000001;      // Set FDOUT1
    K321:1.!=0x00010000;      // Reset FDOUT1
    K322:1.!=0x00000001;      // Set FDOUT1
    K323:1.!=0x00010000;      // Reset FDOUT1
    K323:3.!=0x00000001;      // Reset event counter
    // number of free event before rising the user status bit.
}

```

```
k337:0.! = 256;
// indicate which bit is used in user status to prevent that new event can
be load.
k337:1.! = BIT_USE_CONTINUOUS_TRIGGER;
// Event Setting
trigPos = lconv(0.1, _upi);
combi_num = 0;
for(K198.!=0; K198.!<512; K198.!++)
{
    ELY.! = trigPos,TRIGGER_TYPE_POSITIVE_ONLY,combi_num,0L,0L;
    trigPos+=lconv(0.0015, _upi);
    if (combi_num < 3)
        combi_num = combi_num + 1;
    else
        combi_num = 0;
}
nb_event_prog = 512;
x25:1.0 = nb_event_prog;
// Start the fill of event thread2
start(fill_el_table,2);
TRE.!=0,-1;
MVE.!= 5.0;
WTM.!;
WTT.!=0.2;
MVE.!=0.0;
WTM.!;
pwr.!= 0;

end.! = 2;
}
```

8.9.9.7 Fast Triggers reset function sample

The Fast Triggers feature has improved protection since FW 3.10A. When AccurET generates the **ERR CFG TRIGGER** error (M64=46), the user has to correctly reset the Fast Triggers settings of both axis of AccurET. (EL, Combi, PG,). Refer to [§8.9.7.1](#) for more information.

```
void reset_all_trigger_settings(void)
{
    // Stop trigger mode
    //TRM.(Ax0, Ax1) = -1;
    // Stop PGs
    //SPG.(Ax0, Ax1) = 1,0;
    //SPG.(Ax0, Ax1) = 2,0;
    // Reset Position Type selection
    K336.(Ax0, Ax1) = 0; // Real pos (ML7)
    // Reset output masks for trigger
    K359.(Ax0, Ax1) = 0;// DOUT output mask
    C359.Ax0 = 0, 0;// FDOUTs output mask
    // K171.(Ax0, Ax1) = 0;// Reset DOUTs
    // C5.(Ax0, Ax1) = 0;// Reset FDOUTs

    // Clear trigger combos DOUT masks
    K320.(Ax0, Ax1) = 0;K321.(Ax0, Ax1) = 0;K322.(Ax0, Ax1) = 0;K323.(Ax0, Ax1)
    = 0; K324.(Ax0, Ax1) = 0;
```

```

K325.(Ax0, Ax1) = 0; K326.(Ax0, Ax1) = 0; K327.(Ax0, Ax1) = 0; K328.(Ax0,
Ax1) = 0; K329.(Ax0, Ax1) = 0;
K330.(Ax0, Ax1) = 0; K331.(Ax0, Ax1) = 0; K332.(Ax0, Ax1) = 0; K333.(Ax0,
Ax1) = 0; K334.(Ax0, Ax1) = 0;
K335.(Ax0, Ax1) = 0;

// Clear trigger combos FDOUT masks
K320:1.(Ax0, Ax1) = 0; K321:1.(Ax0, Ax1) = 0; K322:1.(Ax0, Ax1) = 0;
K323:1.(Ax0, Ax1) = 0; K324:1.(Ax0, Ax1) = 0;
K325:1.(Ax0, Ax1) = 0; K326:1.(Ax0, Ax1) = 0; K327:1.(Ax0, Ax1) = 0;
K328:1.(Ax0, Ax1) = 0; K329:1.(Ax0, Ax1) = 0;
K330:1.(Ax0, Ax1) = 0; K331:1.(Ax0, Ax1) = 0; K332:1.(Ax0, Ax1) = 0;
K333:1.(Ax0, Ax1) = 0; K334:1.(Ax0, Ax1) = 0;
K335:1.(Ax0, Ax1) = 0;

// Clear trigger combos Users status masks
K320:7.(Ax0, Ax1) = 0; K321:7.(Ax0, Ax1) = 0; K322:7.(Ax0, Ax1) = 0;
K323:7.(Ax0, Ax1) = 0; K324:7.(Ax0, Ax1) = 0;
K325:7.(Ax0, Ax1) = 0; K326:7.(Ax0, Ax1) = 0; K327:7.(Ax0, Ax1) = 0;
K328:7.(Ax0, Ax1) = 0; K329:7.(Ax0, Ax1) = 0;
K330:7.(Ax0, Ax1) = 0; K331:7.(Ax0, Ax1) = 0; K332:7.(Ax0, Ax1) = 0;
K333:7.(Ax0, Ax1) = 0; K334:7.(Ax0, Ax1) = 0;
K335:7.(Ax0, Ax1) = 0;

// Clear Pulse Generator output masks
K340.(Ax0, Ax1) = 0;// PG1 mask DOUT
K340:1.(Ax0, Ax1) = 0;// PG1 mask FDOUT
K341.(Ax0, Ax1) = 0;// PG2 mask DOUT
K341:1.(Ax0, Ax1) = 0;// PG2 mask FDOUT

// Reset Pulse Generator settings (fixed timing) of controller
K342.(Ax0, Ax1) = 0;// PG1 delay
K342:1.(Ax0, Ax1) = 0;// PG2 delay
K343.(Ax0, Ax1) = 0;// PG1 width
K343:1.(Ax0, Ax1) = 0;// PG2 width
K344.(Ax0, Ax1) = 0;// PG1 interval
K344:1.(Ax0, Ax1) = 0;// PG2 interval
K345.(Ax0, Ax1) = 0;// PG1 pulse count
K345:1.(Ax0, Ax1) = 0;// PG2 pulse count
K339.(Ax0, Ax1) = 0;// PG1 periodicity
K339:1.(Ax0, Ax1) = 0;// PG2 periodicity

// Reset Pulse Generator settings (modulated timing) of controller
KF339.(Ax0, Ax1) = 0f;// PG1 period modulation coefficient
KF339:1.(Ax0, Ax1) = 0f;// PG2 period modulation coefficient
KF342.(Ax0, Ax1) = 0f;// PG1 delay modulation coefficient
KF342:1.(Ax0, Ax1) = 0f;// PG2 delay modulation coefficient
KF343.(Ax0, Ax1) = 0f;// PG1 width modulation coefficient
KF343:1.(Ax0, Ax1) = 0f;// PG2 width modulation coefficient
K353.(Ax0, Ax1)= 2;// PG1 trigger external reference type
K353:1.(Ax0, Ax1)= 2;// PG2 trigger external reference type
K354.(Ax0, Ax1)= 0;// PG1 trigger external reference index
K354:1.(Ax0, Ax1)= 0;// PG2 trigger external reference index
K355.(Ax0, Ax1)= 0;// PG1 trigger external reference depth
K355:1.(Ax0, Ax1)= 0;// PG2 trigger external reference depth

```

```

K356.(Ax0, Ax1)= 0;// PG1 trigger external reference axis
K356:1.(Ax0, Ax1)= 0;// PG2 trigger external reference axis

// Reset the user status of M63( SDC) to zero
K177.(Ax0, Ax1) = 0;

// Clears EL tables in RAM memory
NEW.(Ax0,Ax1) = 8;

if ( (M60.Ax0 & 0x1) == 0 )PWR.Ax0=1;
if ( (M60.Ax1 & 0x1) == 0 )PWR.Ax1=1;
}

```

8.10 Controller status

The 'Controller status' are monitoring variables allowing the user to see if the controller is in error or warning mode.

M	Alias	Name	Comment
M60	SD1	Controller status 1	Controller status (phasing, error,...).
M61	SD2	Controller status 2	Controller status (process error label,...).
M63	SDC	Controller status	Controller status

Monitorings **M60** (or the **SD1** command which is an alias of monitoring M60) as well as **M61** (or the **SD2** command which is an alias of monitoring M61) and **M63** (or the **SDC** command which is an alias of monitoring M63), both using the parameter **K177**, enable the user to know the status of the controller (if the initialization is over, if the motor is moving, if the controller has detected an error...). It is necessary to convert the value shown by monitorings M60, M61 and M63 in binary value. The meaning of each bit of monitorings M60, M61 and M63 is described in [§19](#).

8.10.1 IsMoving bit

The parameter **K62** allows the user to choose how reacts the isMoving bit available in M60 (bit#4) and M63 (bit#20). Only the management for the isMoving bit in M60 and M63 is affected by this parameter.

K	Name	Value	Comment
K62	IsMoving bit configuration	0 1 2	Normal mode: isMoving bit managed by the trajectory generator IsMoving bit is function of the delta measured position between 2 MLTI and the threshold level. If the delta is greater to the threshold, the isMoving bit is set to 1 otherwise the bit is set to 0 IsMoving bit is function of the delta theoretical position between 2 MLTI and the threshold level. If the delta is greater to the threshold, the isMoving bit is set to 1 otherwise the bit is set to 0

Remark: In gantry mode (1 or 2), this parameter must be at the same value.
In gantry mode 2, the master value is duplicated on the slave.

The parameter **K37** allows the user to define the threshold level of the isMoving bit.

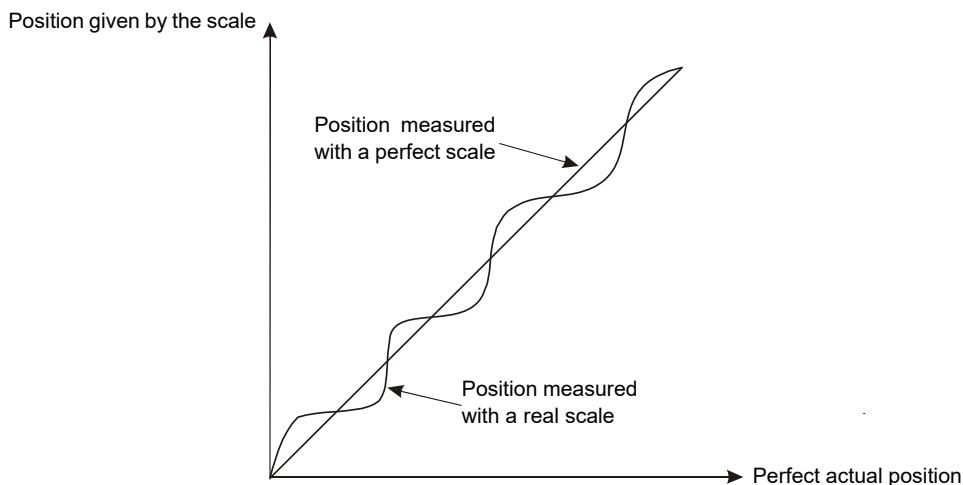
K	Name	Comment	Unit
K37	IsMoving bit threshold	Gives the IsMoving bit threshold	[upi], [rupi]

Remark: In gantry mode (1 or 2), this parameter must be at the same value.
In gantry mode 2, the master value is duplicated on the slave.
On Dual Encoder Feedback configuration, the threshold is applied to the payload position (main encoder).

This 'isMoving' bit feature does not affect the current management for the normal command manager, the WTM, WTS, TRE, WTM, WTP commands, the I2t motor in stall mode as well as the stepper in open loop.

8.11 Encoder Scale Mapping

This paragraph describes how to correct the position given by the encoder along an axis. The linear and rotary encoders have a measurement error with regard to the current position of the axis. This error comes in the form of a linear error according to the position (a μm of the encoder is not necessarily a standard μm) on which is added a random error. The following drawing gives a good idea of the error:



In the application requiring a very high absolute precision on the movement, it may be necessary to rectify the wanted position according to the known errors of the scale. These corrections will reduce the errors of the measuring system and will position the moving load with a better absolute precision. The term of **Scale Mapping** is used to talk about the compensation of those two errors along the scale.

Remark: Both corrections are given with regards to the absolute position on the scale and then, they will be activated only after the homing (if K165 is different from 0).
The correction table is in a float format in the look-up table LDx:0 but the physical correction on the real position is rounded to the nearest integer.

8.11.1 Setting

The look-up table at depth 0 (LDx:0) must be used for the setting of the correction values of the Scale Mapping. The number of correction points can be freely chosen (defined with K294), but a maximum of 8192 correction values can be programmed and saved in the controller.

Remark: The mapping setting must be done with the motor(s) in power off otherwise a **MAPPING ERROR** error (M64=74) will occur.

The parameter **K294** defines the number of point on LDx:0 used for the mapping compensation.

K	Name	Comment
K294	Point number for Scale Mapping	Gives the number of point for the Scale Mapping correction

Remark: If K294=0, the number of points is equal to 8192

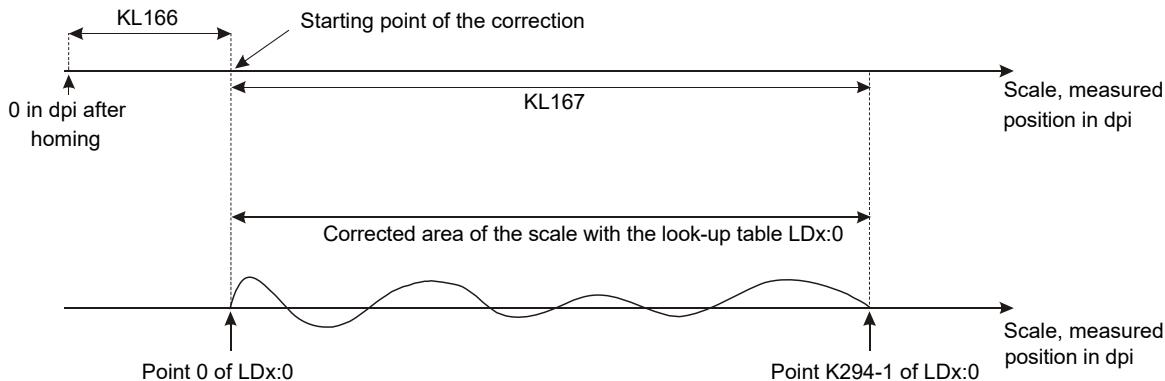
The parameter **KL166** defines the position (in dpi) of the scale from where the correction, defined by the point 0 of the look-up table, will be applied.

K	Name	Comment	Units
KL166	Start of the position correction	Position where the Scale Mapping correction starts	[dpi]

The parameter **KL167** defines the length (in dpi) on which the look-up table corrects the set point. This correction is added to the real position (measured).

K	Name	Comment	Units
KL167	Length of active correction	Length where the Scale Mapping correction is active	[dpi]

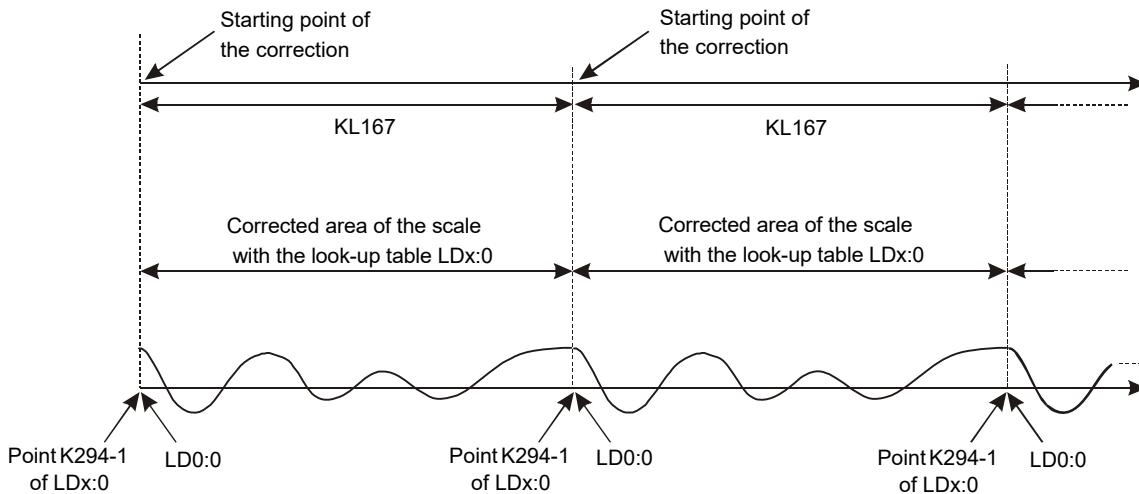
- **Linear Scale Mapping**



Remark: The first point and the last point must be at 0 in order to be positioned in the same point when the axis is out of the mapping 1D area correction.

- **Rotary Scale Mapping**

To apply cyclic corrections on one motor's turn, the parameter **KL167** must be set to include the distance in [dpi] corresponding to the value defined by **KL27**. The parameter **KL166** must be at 0 when the rotary Scale Mapping is used. Whatever the distance to be covered, the same look-up table is applied again and again as long as the motor is moving. To activate the rotary encoder mapping, the parameter **K165** must be equal to 4.



Remark: The first point and the last point must have the same value. This value can be different than 0.

8.11.2 Activation of the corrections

After the homing, the parameter **K165** allows the user to activate these corrections:

K165	Linear Scale Mapping	Rotary Scale Mapping
0	No	No
1	Yes	No
4	No	Yes

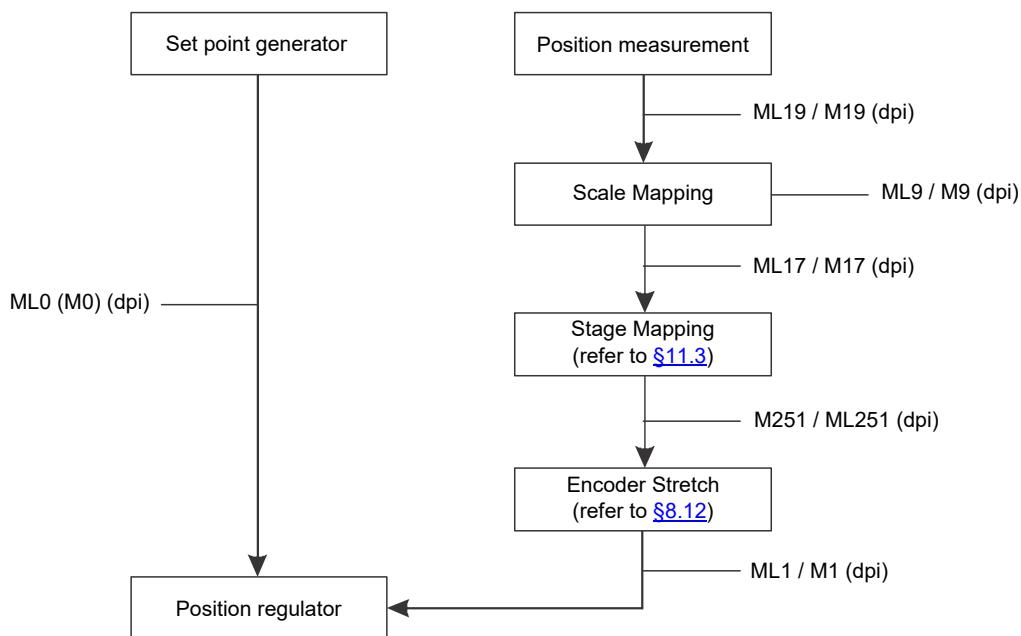
By default, there is no correction (K165=0). If K165 is equal to 1 or 4, the correction will be activated (only after the homing). The setting of the parameter K165 will be done only after the end of a movement. Bit#1 of monitoring M61 allows the user to see if the Scale Mapping feature is enabled or disabled.

Remark: The parameter K165 can be modified when the motor is in power on, however, KL166, KL167 and K294 cannot be modified when the motor is on power on, otherwise the **MAPPING ERROR** error (M64=74) occurs.
 The applied correction is rounded to the integer. When the position is out of the correction zone (defined by KL166 and KL167), the applied correction is the last value of the table.
 Monitoring ML19/M19 is linked to monitoring ML17/M17 if K165 is set to 0.
 Monitoring ML9/M9 is not reset (to 0) when K165 changes from 1 to 0.

8.11.3 Use

The monitoring **M9 (ML9)** allows the user to monitor, on the scope, the value of the correction.

M	Name	Comment	Units
M9 ML9	Scale Mapping compensation	Gives the Scale Mapping compensation value. The monitoring M9 corresponds to the LSL part of the monitoring ML9.	[dpi]



Remark: As the correction is done with regard to the measured position (in dpi), the SET command, changing the offset for the variable (in upi) can be used without affecting the correction.
 The position capture and the Fast Triggers on the real position take the correction into account.

The monitorings **M17 (ML17)** and **M19 (ML19)** allow the user to know two other positions.

M	Name	Comment	Units
M17 ML17	Real position after Scale Mapping	Gives the real position after Scale Mapping before Stage Mapping. The monitoring M17 corresponds to the LSL part of the monitoring ML17.	[dpi] [rdpi]
M19 ML19	Real position before Scale Mapping	Gives the real position coming from the encoder before Scale Mapping. The monitoring M19 corresponds to the LSL part of the monitoring ML19.	[dpi] [rdpi]

8.12 Encoder Stretch

The Encoder Stretch function allows the user to apply a correction on the measured position. The **UST** command allows the use of this stretch functionality.

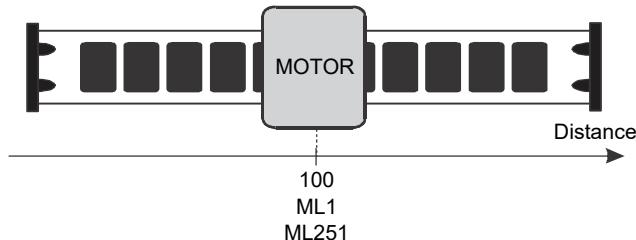
Command	Comment
UST.<axis>=<P1>,<PL2>,<PF3>	Encoder Stretch command. <P1>=Enable/disable the function (0: disabled, 1: enabled); <PL2>=indicates where is the stretch position 0; <PF3>=indicates the stretch value (limited to ±2%)

Remark: If $<\text{P1}> = 0$, the functionality is disabled and then the parameters $<\text{PL2}>$ and $<\text{PF3}>$ are optional. If the value of $<\text{P1}>$ is different than 0 or 1 or if the value of $<\text{P1}>$ is equal to 1 and the number of the command parameters is different than 3, the **BAD CMD PARAM** error (M64=70) occurs.

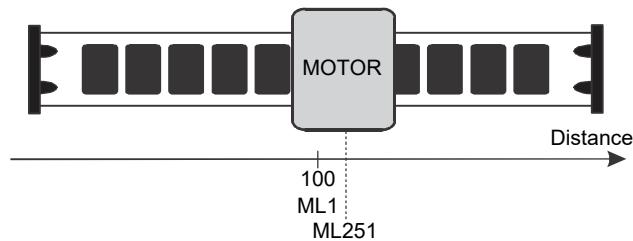
If the type value of $<\text{PL2}>$ is different than a INT64, the **BAD CMD PARAM** error (M64=70) occurs.

The minimum value for $<\text{PF3}>$ is -0.02F (-2%) and the maximum one is $+0.02\text{F}$ (+2%). If the type value of $<\text{PF3}>$ is different than a FLT32 or if the value is smaller than -0.02F or if the value is greater than $+0.02\text{F}$, the **BAD CMD PARAM** error (M64=70) occurs.

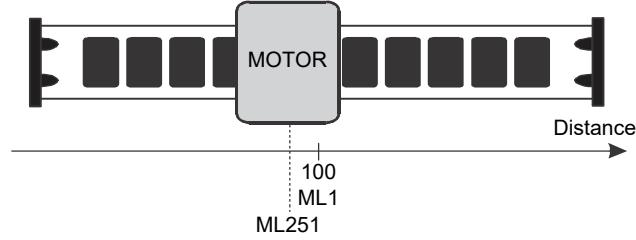
UST.<axis>=0,0L,0F
MVE.<axis>=100
ML0=100
ML251=100
ML1=100



UST.<axis>=1,0L,0.02F
MVE.<axis>=100
ML0=100
ML251=102
ML1=100



UST.<axis>=1,0L,-0.02F
MVE.<axis>=100
ML0=100
ML251=98
ML1=100



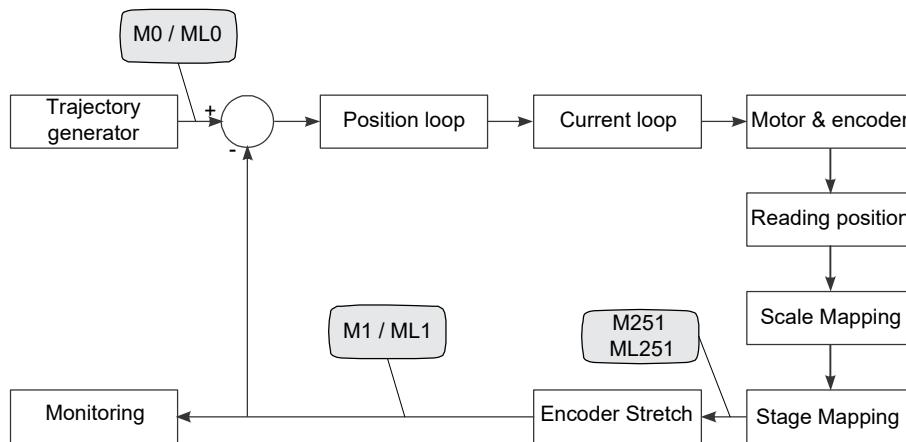
For all errors generated by the command, the Encoder Stretch is disabled, the offset is set to 0 and the stretch value is set to 0. The real value of the position given by M1 / ML1 is as follows:

$$ML1 = ML251 \times \left(\frac{1}{1 + \text{Stretch value}} \right)$$

The stretch function is available with the following configurations: movement with the trajectory generator (MVE), movement in interpolated mode through the UltimET (ITP mode), movement with external reference (K61.<axis>≠1) and movement with current reference(K63.<axis>=1).

The monitoring **M251**, **ML251**, **M252**, **ML252**, **MF253** and **M254** gives the status on the functionality:

M	Name	Comment	Units
M251 ML251	Real position after Stage Mapping	Gives the real position after Stage Mapping before the Encoder Stretch. The monitoring M251 corresponds to the LSL part of the monitoring ML251.	[dpi] [rdpi]
M252 ML252	Stretch offset value	Gives the stretch offset value. The monitoring M252 corresponds to the LSL part of the monitoring ML252.	[dpi] [rdpi]
MF253	Stretch factor value	Gives the stretch factor value.	-
M254	Stretch state	Gives the Encoder Stretch state (enable/disable).	-



There are some restrictions with the interpolated mode (ITP mode):

- The configuration of the Encoder Stretch must be set before entering to ITP mode. As the axis does not move after the UST command, it is necessary to apply a movement to assume that the axis is on the right theoretical position before entering in interpolation mode.
- If the user sends the UST command when the axis is in interpolation mode, the **STRETCH ERROR** error (M64=73) occurs.

8.13 Set / reset bit(s) management

The **CH_BIT_REG32** command allows the user to perform a reset / set on some bits in a register (integer register K, C and X) without affecting the other bits of this register.

Command	Comment
CH_BIT_REG32.<axis>=<P1>, <P2>,<P3>	Write any bit (set bits and/or clear bits mask) <P2> with defined mask value <P3> of any 32 integer register <P1> without modifying the other bits the register

This will do $<P1> = (<P1> \& \sim<P2>) | (<P3>\&<P2>)$

Example:

Thread 1 controls bits 0, 1 and 4 of K177:0.<axis>.

- Thread 1 wants to set bit#0 and 4, and clear bit#1, without modifying all the other bits. The command will be:
`CH_BIT_REG32.<axis>=K177:0.<axis>, 0x13, 0x11`
- Thread 1 wants to clear bit#4 and set bit#0 and 1, without modifying all the other bits. The command will be:
`CH_BIT_REG32.<axis>=K177:0.<axis>, 0x13, 0x3`
- Thread 2 wants to clear bit#2 and set bit#3, without modifying all the other bits. The command will be:
`CH_BIT_REG32.<axis>=K177:0.<axis>, 0xC, 0x4`

As the CH_BIT_REG32 command cannot be interrupted, there will be no conflict between threads.

8.14 Set several registers

The **SSR** command (Set Several Registers) allows the user to affect 1 up to 6 registers in the same command.

Command	Comment
SSR.<axis>=<P1>,<P2>,[<P3>,...<P12>]	Command to set several registers: <P1>: 1st register and depth definition, <P2>: 1st register value, <P3>: 2nd register and depth definition (optional parameter), <P4>: 2nd register value (optional parameter),..., <P11>: 6th register and depth definition (optional parameter), <P12>: 6th register value (optional parameter).

This command is only available for X, XL, XF and EL registers. In the same command, it is allowed to set different types of registers for example:

SSR.<axis>=X1:1, 100, XF1:1, 24.523F, XL3:1, 1234L

If the register number is greater than the maximum allowed for the type, the **REGISTERNUM ERR** error (M64=38) will occur.

If the register type is K, KL, KF, KD, C, CF, CL, CD, M, ML, MF, MD, P, XD or LD, the **SET REG BAD PAR** error (M64=44) will occur.

For EL registers, the **ERR CFG TRIGGER** error (M64=46) could occur if the value for some depth is out of range.

All registers modified by the SSR command are in the axis receiving the command. It is possible to use the indirect parameterization on the SSR command but the Y value (indirect parameterization) is the same for all the registers in the command: SSR.<axis>=XY:1, 10, XY:2, 20, XFY:1, 10F

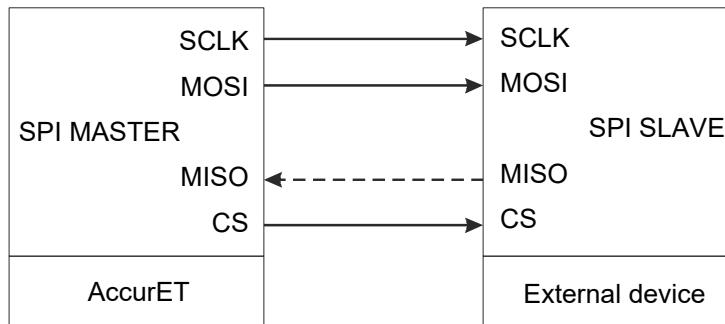
8.15 Serial Peripheral Interface bus (SPI)

The Serial Peripheral Interface (SPI) bus is available with all AccurETs (from firmware 2.07A). The FDINs and FDOUTs are used to communicate with an external device or with another AccurET (refer to the corresponding 'Hardware manual' for more information about the FDINs and FDOUTs description). The SPI bus can operate in master or slave mode. Here are the following abbreviation description to understand the SPI principle.

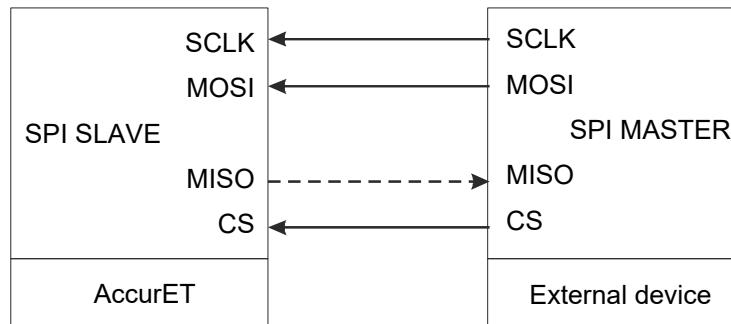
SCLK	Serial CLocK (output from master)
MOSI	Master Output/Slave Input data (output from master)
MISO	Master Input/Slave Output data (output from slave)
CS	Chip Select (output from master)
CPHA	Clock PHAse
CRC	Cyclic Redundancy Check

The following pictures show these different modes.

- **AccurET as Master and External device (or AccurET) as Slave**

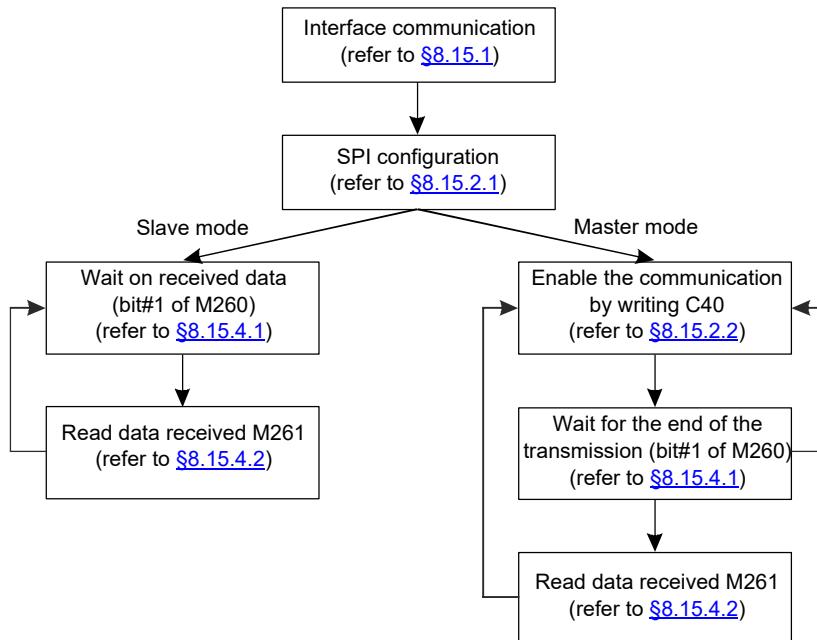


- **AccurET as Slave and External device (or AccurET) as Master**



Remark: Only one SPI configuration is possible per controller (and not per axis).

The diagram below presents the different steps to use the SPI function:



8.15.1 SPI interface configuration

8.15.1.1 Serial clock

- In master mode

The common parameter **C361:1** defines which FDOUT is used as SCLK.

C	Name	Comment
C361	FDOUT mask for SPI	FDOUT mask for SPI clock output (available at depth 1): bit#0-3 respectively FDOUT1-FDOUT4 (non-inverted) and bit#16-19 respectively FDOUT1-FDOUT4 (inverted).

- In slave mode

The common parameter **C306:1** defines which FDIN is used as SCLK.

C	Name	Comment
C306	FDIN mask for SPI	FDIN mask for SPI clock input (available at depth 1): bit#0-5 respectively FDIN1-FDIN6 (non-inverted) and bit#16-21 respectively FDIN1-FDIN6 (inverted).

Remark: Only one bit must be set at the same time
FDIN5 and 6 are not available on AccurET Modular 400/600

8.15.1.2 Data output (MOSI in master or MISO in slave)

The common parameter **C360:1** defines which FDOUT is used as MOSI in master or MISO in slave.

C	Name	Comment
C360	FDOUT mask for SPI	FDOUT mask for SPI data output (available at depth 1): bit#0-3 respectively FDOUT1-FDOUT4 (non-inverted) and bit#16-19 respectively FDOUT1-FDOUT4 (inverted).

Remark: Several bits can be configured at the same time. In case of conflict between normal and inverted output, normal has the priority.

8.15.1.3 Data input (MISO in master or MOSI in slave)

The common parameter **C305:1** defines which FDIN is used as MISO in master or MOSI in slave.

C	Name	Comment
C305	FDIN mask for SPI	FDIN mask for SPI data input (available at depth 1): bit#0-5 respectively FDIN1-FDIN6 (non-inverted) and bit#16-21 respectively FDIN1-FDIN6 (inverted)

Remark: Only one bit must be set at the same time
FDIN5 and 6 are not available on AccurET Modular 400/600

8.15.1.4 Chip Select

- In master mode

The common parameter **C362:1** defines which FDOUT is used as CS (Chip Select).

C	Name	Comment
C362	FDOUT mask for SPI	FDOUT mask for SPI chip select output (available at depth 1): bit#0-3 respectively FDOUT1-FDOUT4 (non-inverted) and bit#16-19 respectively FDOUT1-FDOUT4 (inverted).

- In slave mode

The common parameter **C307:1** defines which FDIN is used as CS (Chip Select).

C	Name	Comment
C307	FDIN mask for SPI	FDIN mask for SPI chip select input (available at depth 1): bit#0-5 respectively FDIN1-FDIN6 (non-inverted) and bit#16-21 respectively FDIN1-FDIN6 (inverted).

Remark: Only one bit must be set at the same time
FDIN5 and 6 are not available on AccurET Modular 400/600

8.15.2 Message configuration

8.15.2.1 SPI configuration

The common parameter **C41** defines the configuration of the SPI message.

C	Name	Bit#	Comment
C41	SPI configuration	0	Defines the mode: 0: Slave mode; 1: Master mode
		1	Defines the CPHA: 0: the data is captured on clock's rising edge; 1: the data is captured on clock's falling edge
		2	Defines the bit order: 0: MSB first; 1: LSB first
		3	Reserved for future use
		4-6	Defines the clock frequency: 000: Reserved; 001: Reserved; 010: Reserved; 011: 625 kHz; 100: 312 kHz; 101: 156 kHz; 110: 78 kHz; 111: 39 kHz
		7	Reserved for future use
		8-9	Defines the data size: 00: 8 bits; 01: 16 bits; 10: 32 bits; 11: Not defined
		10-15	Reserved for future use
		16	Defines the CRC: 0: CRC disable; 1: CRC enable
		17-19	Defines the CRC size: 000: CRC-1; 001: Reserved; 010: CRC-3; 011: CRC-4; 100: CRC-5; 101: CRC-6; 110: CRC-7; 111: CRC-8
		20-21	Reserved for future use
		22-23	Defines the CRC error management: 00: AccurET in master or slave mode and data input, in case of CRC error, puts directly the CRC status bit in error and increase CRC error counter M262; 01: AccurET in master only mode and data input, in case of the 1st CRC error, asks for the SPI data message again and in case of a 2nd error puts the CRC status bit in error and increase CRC error counter M262. 10 & 11: Reserved

8.15.2.2 Send message

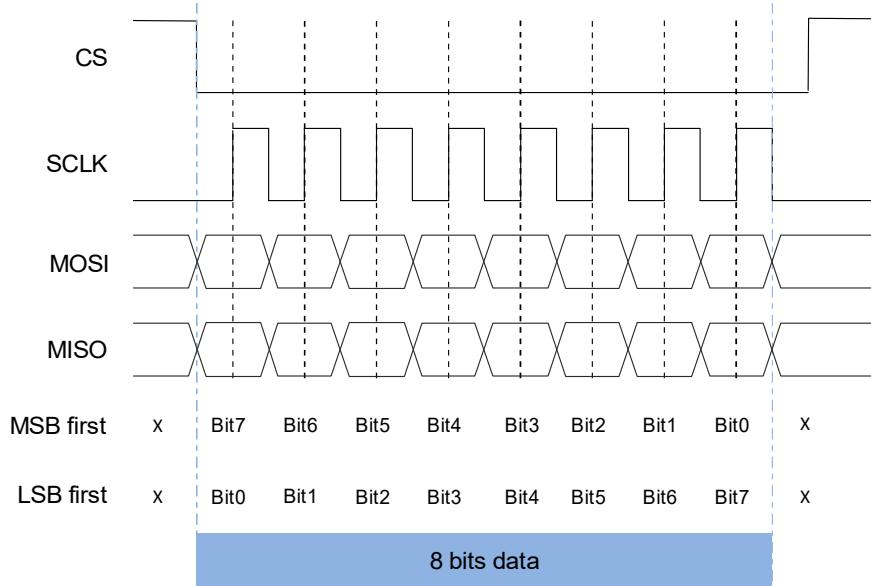
The common parameter **C40** defines the data to be sent over the SPI message.

C	Name	Comment
C40	SPI data out	Defines the SPI data out and starts communication if the SPI is set as master.

8.15.2.3 Examples

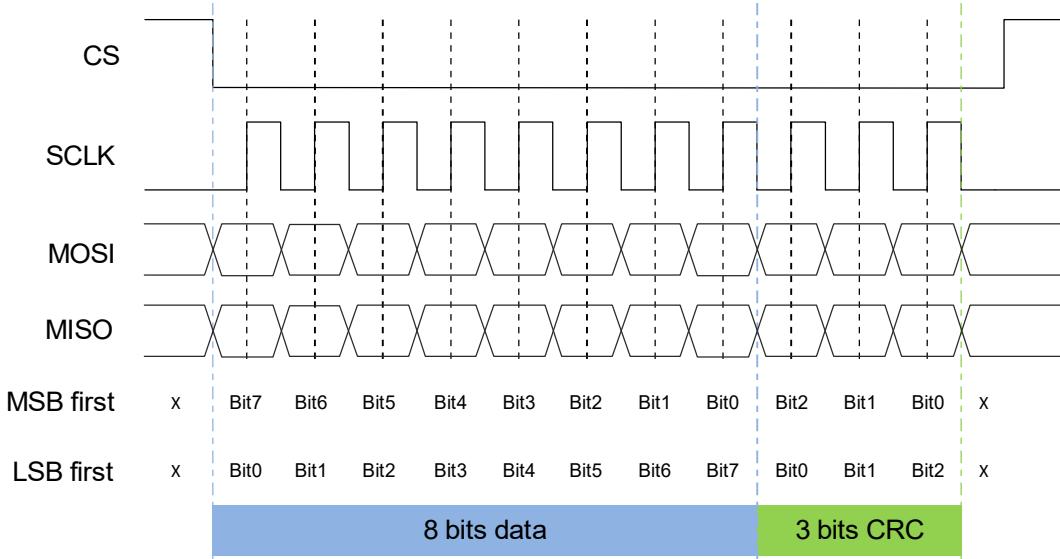
- Without CRC

The following timing diagram shows the SPI transfer of a message made up of 8 bits with CPHA=0 (data sample on clock's rising edge).

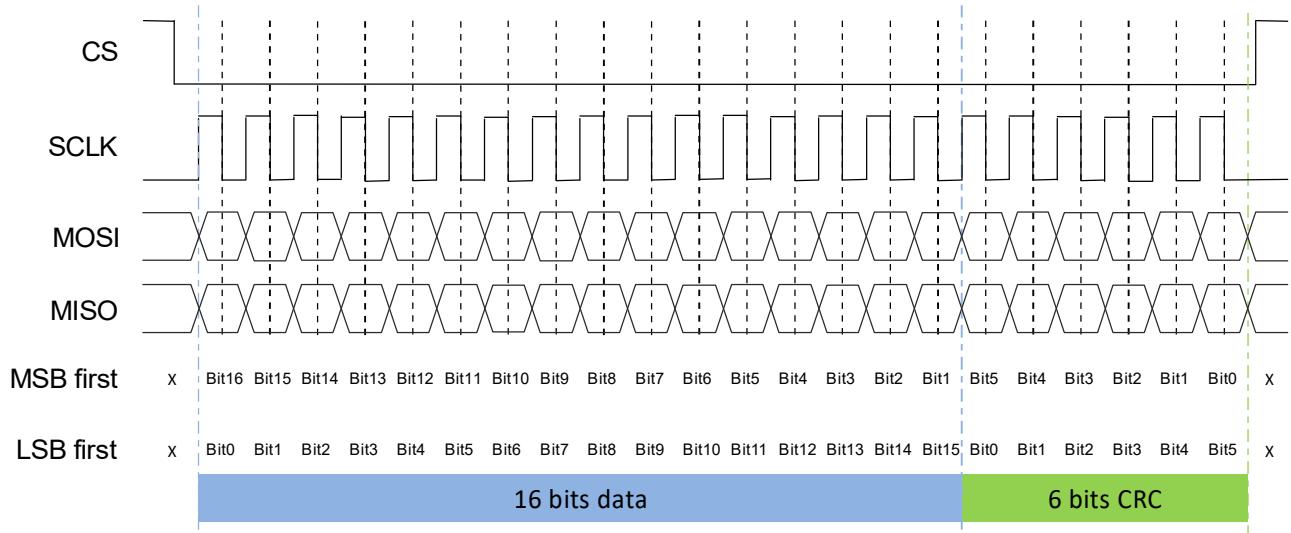


- With CRC

The following timing diagram shows the SPI transfer of a message made up of 8 bits data and 3 bits of CRC with CPHA=0 (data sample on clock's rising edge).



The following timing diagram shows the SPI transfer of a message made up of 16 bits data and 6 bits of CRC with CPHA=1 (data sample on clock's falling edge).



Remark: The message sent (or received) by AccurET is composed of two parts: data of 8, 16 and 32 bits followed by a CRC (if this one is activated) of 1 or 3-8 bits.

8.15.3 CRC computation

The CRC implemented follows the standard of CRC computing. The table below summarizes the implemented polynomial for each available CRC.

Bit's number	CRC type	Standard	Polynomial MSB first	Polynomial in binary & hexa MSB first	Polynomial LSB first	Polynomial in binary & hexa LSB first
1	CRC-1	Parity bit	X^1+1	11b 0x3	X^1+1	11b 0x3
3	CRC-3		X^3+X+1	1011b 0xB	X^3+X^2+1	1101b 0xD

Bit's number	CRC type	Standard	Polynomial MSB first	Polynomial in binary & hexa MSB first	Polynomial LSB first	Polynomial in binary & hexa LSB first
4	CRC-4	G.704	X^4+X+1	10011b 0x13	$X^4+X^3+X^2$	11100b 0x1C
5	CRC-5	USB	X^5+X^2+1	10'0101b 0x25	$X^5+X^4+X^2$	11'0100b 0x34
6	CRC-6	G.704, CRC-6-ITU	X^6+X+1	100'0011b 0x43	$X^6+X^5+X^4$	111'0000b 0x70
7	CRC-7	G.707, G.832, SD	X^7+X^3+1	1000'1001b 0x89	$X^7+X^6+X^3$	1100'1000b 0xC8
8	CRC-8	I.432, ATM HEC, ISDN, CRC-8-CCITT	X^8+X^2+X+1	1'0000'0111b 0x107	$X^8+X^7+X^6+X^5$	1'1110'0000b 0x1E0

8.15.4 Monitoring

8.15.4.1 Status of the SPI communication

The monitoring **M260** gives the status of the SPI communication.

M	Name	Bit#	Comment
M260	SPI status	0	Defines the buffer out: 0: empty (you can write some new data to be sent) 1: full, waiting for the transmission to start (work only for slave mode, as start of communication will be triggered if it is a master). If data are written while in this state, previous data will be overridden. Please read information for bit#1 before using this one.
		1	This bit indicates the end of the transmission, meaning new data are ready to be read. This bit is cleared when writing new data (C40), so in case user needs this information, it must wait for this bit to be equal to 1 (and not testing bit#0) before writing the next data to be sent. If there is no data to be sent (only reception), any value can be written to C40 to clear this bit. 0: In transmission process 1: New data is ready to be read Bit is set at the end of transmission, cleared by writing of C40
		2-4	Internal used
		5-11	Give transmission position (can be useful in case of slow speed slave mode), start=0, increment of 1 each bit send/received
		12	Defines the CRC error: 0: CRC Ok; 1: CRC error

8.15.4.2 SPI data received

The monitoring **M261** gives the data received from the SPI communication.

M	Name	Comment
M261	SPI data received	Gives the data received from the SPI communication

8.15.4.3 SPI CRC error counter

The monitoring **M262** gives the SPI CRC error counter.

M	Name	Comment
M262	SPI CRC error counter	Gives the SPI CRC error counter

8.15.5 Use of SPI communication

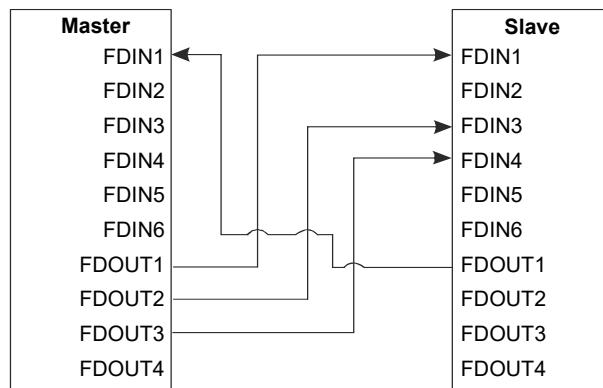
8.15.5.1 CRC error management

The error detection is only activated when the AccurET receives the data and the CRC is enabled (bit#16 of C41). The following error management is available:

- CRC Error management: “00” (bit#22-23 of C41). When the AccurET is master or slave and receives data from the SPI.
 - If a CRC error occurred, a bit is set in the status message and the CRC error counter (M262) is incremented but the monitoring M261 (data received) remains the same.
- CRC Error management: “01” (bit#22-23 of C41). When the AccurET is master (only) and receives data from the SPI. If a CRC error occurred, the AccurET asks again automatically for a second time the message. Two cases:
 - There is no CRC error, for the second time the error counter (M262) is not incremented and M261 (data received) is updated.
 - There is a CRC error, a bit is set in the status message (M260) and the CRC error counter (M262) is incremented but the monitoring M261 (data received) keeps the last validated message.

8.15.5.2 SPI configuration example

Here is an example (for all AccurET except AccurET 400 / 600 having 4 FDINs) where the master sends data to a slave and the slave sends data to the master.



• Define

```

#define SLAVE_AXIS 2
#define MASTER_AXIS 0

#define SPI_MASTER 0x1
#define SPI_SLAVE 0x0
#define SPI_CLK_SPEED_390625Hz 0x70
#define SPI_8BITS 0
#define SPI_ENABLE_CRC 0x10000
#define SPI_DATA_RECEIVED 0x2

```

• Slave configuration (FDIN/FDOUT)

```

C305:1.SLAVE_AXIS = 0x1;          // FDIN mask for SPI data input
C306:1.SLAVE_AXIS = 0x4;          // FDIN mask for SPI clock input
C307:1.SLAVE_AXIS = 0x8;          // FDIN mask for SPI chip select input
C360:1.SLAVE_AXIS = 0x1;          // FDOUT mask for SPI data output
C361:1.SLAVE_AXIS = 0x0;          // FDOUT mask for SPI clock output
C362:1.SLAVE_AXIS = 0x0;          // FDOUT mask for SPI chip select output

```

• Master configuration

```
C360:1.MASTER_AXIS = 0x1;          // FDOUT mask for SPI data output
```

```
C361:1.MASTER_AXIS = 0x2;           // FDOUT mask for SPI clock output
C362:1.MASTER_AXIS = 0x4;           // FDOUT mask for SPI chip select output
C305:1.MASTER_AXIS = 0x1;           // FDIN mask for SPI data input
C306:1.MASTER_AXIS = 0x0;           // FDIN mask for SPI clock input
C307:1.MASTER_AXIS = 0x0;           // FDIN mask for SPI chip select input
```

- **Data configuration for 8 bit of communication with CRC=1 at 39.0625kHz**

```
c41.MASTER_AXIS = SPI_MASTER | SPI_CLK_SPEED_390625Hz | SPI_8BITS |
SPI_ENABLE_CRC;
c41.SLAVE_AXIS = SPI_SLAVE | SPI_CLK_SPEED_390625Hz | SPI_8BITS | SPI_ENABLE_CRC;
```

- **Data exchange**

```
data = rand() & 0x000000FF; // Set random data for 8 bits
// prepare data to send to master
c40.SLAVE_AXIS = data;
// Start communication
c40.MASTER_AXIS = data;
if(M260.MASTER_AXIS & SPI_DATA_RECEIVED !=0) // Check that data is sent
    err;
while(M260.SLAVE_AXIS & SPI_DATA_RECEIVED == 0); // Wait data is received
//Read data received
received_data = M261.SLAVE_AXIS;
```

8.16 Traces management

The trace management allows the user to acquire up to 4 traces on each node, each trace having up to 16384 points.

- on a single controller (non-synchronized mode)
- on several controllers (synchronized mode with an UltimET) by using a trigger on only one node. For example, different signals could be acquired on all nodes at the beginning of the movement on node 2.

When an acquisition is made in the multi-axis scope of the ComET software, the trace acquisition is managed automatically. When an acquisition is requested by the customer's application, it may be done from the PC application or the sequence. If it is managed in the PC application using the EDI libraries (refer to the 'EDI User Manual') predefined functions are available otherwise the procedure described below must be used.

Remark: It is not allowed to use traces in the application and in ComET at the same time.

8.16.1 Traces configuration

The parameters **K120** to **K129** are used to set the different configurations.

K	Name	Comment
K120	Synchronization	Enables trace synchronization mode 0 = non-synchronized mode 1 = synchronized mode

K	Name	Comment
K121	Register type	Selects the type of register to trace (depth 0 for trace 0; depth 1 for trace 1,...) 1 = User variable (X) 2 = Parameter (K) 3 = Monitoring (M) 8 = Look-up table (L) 13 = Common parameter (C) 33 = User variable (XF) 34 = Parameter (KF) 35 = Monitoring (MF) 45 = Common parameter (CF) 65 = User variable (XL) 66 = Parameter (KL) 67 = Monitoring (ML) 77 = Common parameter (CL) 97 = User variable (XD) 98 = Parameter (KD) 99 = Monitoring (MD) 104 = Look-up table (LD) 109 = Common parameter (CD)
K122	Register number and depth	Selects the register number (bit#0-15) and its depth (bit#16-23). Depth 0 for trace 0; depth 1 for trace 1,...
K123	Sampling period	Sampling time between 2 points of the trace (period of 25us). As the acquisition is made in the PLTI, K123 must be a multiple of 2.
K124	Edge selection	For trigger mode, specify if the trig condition is fulfilled on falling (-1), rising (1) or both edges (0)
K125	Trigger mode	Trace trigger mode selection: 0 = Immediate acquisition 1 = Start of movement (bit moving 0=>1 (bit#4 of M60 and bit#20 of M63)). 2 = End of movement (bit moving 1=>0 (bit#4 of M60 and bit#20 of M63)). 3 = Trigger at a position defined by KL127 (K126: not used) 4 = Trigger at a given value (defined in K127, KF127, KL127 or KD127) of the trace defined in K126 5 = Trigger never starts. An external freeze command in synchro mode can stop the acquisition 6 = Trigger at a given value (defined in K127 or KF127, KL127, KD127) of a register defined in K126 7 = Trigger on a bit field state (K127 or KL127:0: bit to be low, K127 or KL127:1: bit to be high) of a register defined in K126 8 = Trigger on a bit field rising or falling edge (K127 or KL127:0: bit on rising edge, K127 or KL127:1: bit on falling edge) of a register defined in K126
K126	Trigger parameter	Definition of the trigger parameter according to the trigger mode (K125) - K125 = 1 or 2 => On AccurET K126 is not used and on UltimET K126 defines the interpolation group - K125 = 3 => K126 is not used - K125 = 4 => K126 defines the trace number that will be monitored to trigger the acquisition - K125 = 5 => K126 is not used - K125 = 6, 7 or 8 => K126:0 defines the type and K126:1 defines the depth and the number of the register that will be monitored to trigger the acquisition
K127 KD127 KF127 KL127	Trigger level	Definition of the trigger level according to the trigger mode (K125) - K125 = 1 or 2 => K127 is not used - K125 = 3 => KL127 defines the position that triggers the acquisition - K125 = 4 => K127, KF127, KL127 or KD127 defines the value that triggers the acquisition - K125 = 5 => K127 is not used - K125 = 6 => K127, KF127, KL127 or KD127 defines the value that triggers the acquisition - K125 = 7 or 8 => K127:0 or KL127:0 defines the bits on rising edge, K127 or KL127:1 defines the bits on falling edge
K128	Points number	Number of points to acquire for the traces (if K128=0, the maximum number of points is taken into account)
K129	Pre- / post-trigger	Number of points before or after the trigger. If this number is negative, a pre-trigger is defined and if this number is positive, a post-trigger is set. To get the pre / post-trigger time, compute K129 * K123 * 25 μ s.

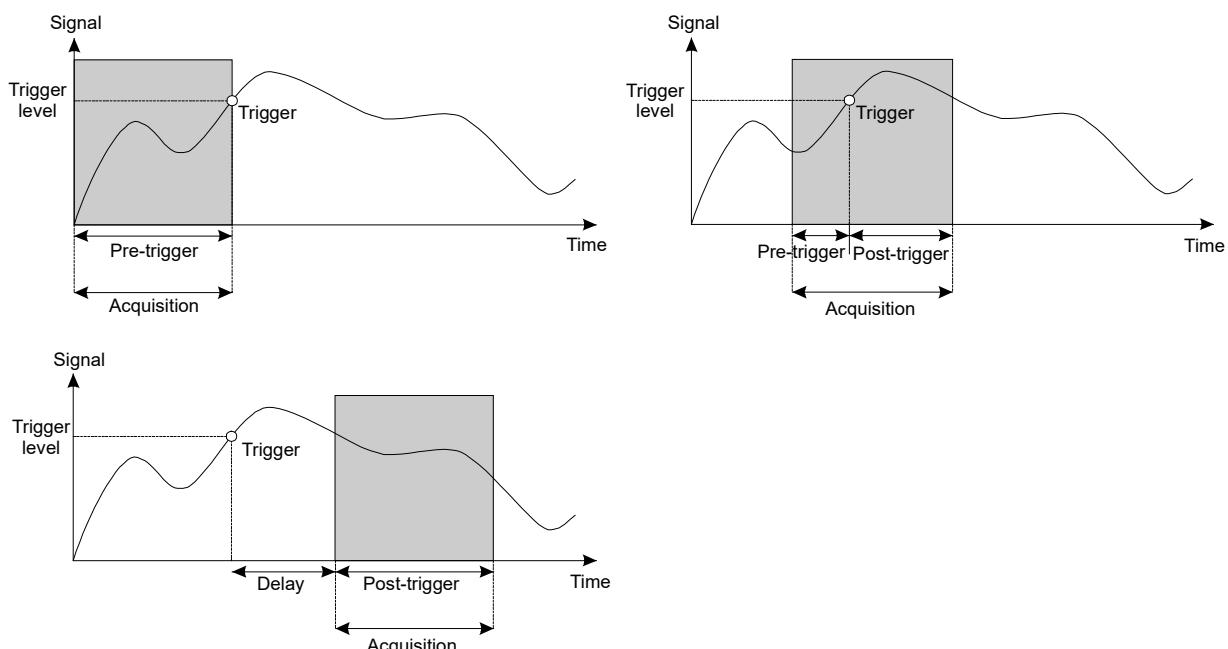
The traces configuration is stored into an internal structure and is accessible through monitoring registers. With these monitorings, the application (ComET, user application, sequence) has the information on what has been traced. Typically, ComET will be able to upload unknown traces of a controller and show which registers were traced, and when.

M	Name	Comment
M120	Synchronization mode	Gives the traces synchronization mode
M121	Register type	Gives the type of register to trace (depth 0 for trace 0; depth 1 for trace 1,...)
M122	Register number and depth	Gives the register number and its depth (bit#0-15 for register number, bit#16-23 for depth)
M123	Sampling time	Gives the sampling time between 2 points of the trace (period of 25us)
M124	Edge selection	Trace trigger edge
M125	Trigger mode	Gives the trigger mode used for the trace
M126	Trigger parameter	Gives the trigger parameter according to the trigger mode
M127 MF127 ML127	Trigger level	The value gives the trigger level according to the trigger mode
M128	Points number	Gives the number of acquired points for the traces
M129	Pre- / post-trigger	Gives the pre- or post-trigger value
M131	Acquisition state	Gives the acquisition state of the trace: 0 = Initialization 1 1 = Initialization 2 2 = Wait for buffer synchronization 3 = Wait for trigger 4 = Trigger condition reached 5 = End of acquisition
M134 ML134	Freeze time	Gives the freeze time of the measure
M135	T register number at trig condition	Gives the number of the T register at the trigger condition
M136	Points number	Gives the number of available points

8.16.1.1 Pre- and post-trigger capabilities

The user is able to visualize pre-trigger only, post-trigger only or pre-trigger and post-trigger:

- Pre-trigger: The user wants to understand the cause of an error. The error event is identified by a trigger, when the trigger occurs, the user has the capability to analyze the cause of this error by an acquisition of parameters on different traces.
- Post-trigger: The user has the possibility to analyze what is happening after trigger events on different parameters. For instance, if an error occurs, it is possible to analyze what are the consequences on other parameter. This analysis can be done from the trigger event or by choosing a delay after the trigger event.



When an acquisition with a pre-trigger is defined, the measure can stop before having all the points (defined by the parameter K128) if the trigger condition is reached before the whole pre-trigger time. To be sure that all points are available, monitorings M135 and M136 have to be checked.

8.16.2 Non-synchronized mode

The non-synchronized mode (on a single controller) is enabled when the parameter K120=0. To start the acquisition, the **ZTE** command must be used:

Command	<P1>	Comment
ZTE.<axis>=<P1>	0 1 2	Stops the acquisition (nevertheless the start and end index are available) Enables single axis (if already enabled: restart of the function) if K120=0: starts immediately the acquisition if K120=1: starts at the next interrupt 0 as soon as the master-of-acquisition has sent the order to start the buffering. If this order never comes, the acquisition will never happen (refer to S8.16.3) Enables multi-axes. if K120=0: starts immediately the acquisition if K120=1: the axis receiving this command is enabled and becomes the master-of-acquisition (refer to S8.16.3).

Remark: In non-synchronized mode (K120=0), ZTE.<axis>=1 or ZTE.<axis>=2 will have the same effect as there is only one controller.

Example:

Example of an acquisition on axis 0 with 2 traces: the real position given by the monitoring ML7 and the measured current given by the monitoring MF31. The acquisition has 16001 points with a sampling time of 50 µs and lasts 800 ms. The trigger is raising edge sensitive and is set on the real position (ML7) with a level of 1000000incr. A pre-trigger of 200ms is chosen.

```
K120.0=0x0;
K121.0=0x43,0x23,0x0,0x0; // Traces synchronization mode
K122.0=0x7,0x1F,0x0,0x0; // Trace register type selection
K123.0=2; // Trace register number and depth
K124.0=1; // Traces sampling time selection
K125.0=4; // Traces trigger edge selection
K126.0=0,0; // Traces trigger mode selection
K127.0=0,0; // Selection of the trigger parameter
KF127.0=0.0F,0.0F; // Selection of the trigger level
KL127.0=1000000L,0L; // Selection of the trigger level
KD127.0=0.0D,0.0D; // Definition of the trigger level
K128.0=16001; // Number of points to acquire
K129.0=-4000; // Pre or post trigger value
```

As soon as the traces parameters are set, the command ZTE can be sent:

```
ZTE.0=1; // Trace trigger enable
```

When M131 = 5 or when the bit#27 of M63 is reset, the acquisition is finished and the points can be read.

8.16.3 Synchronized mode

The traces can be synchronized:

- on both axes of a dual-axis controller through USB or Ethernet
- from the master of the TransnET or from another PC connected with USB to each controller, which the trace is wanted from, but in that case, the controllers must be linked by the TransnET communication bus.

To synchronize the traces, a master-of-acquisition must be defined. This master, which must be an axis involved in the synchronized acquisition (UltimET or AccurET), will send the order (via TransnET) to start the buffering to every node (from then on, the acquisition starts on the same interrupt). To perform a measure, the user has to set first the parameters K120 to K129 of all axes. The parameter K123 as well as the parameter K128 must be identical in each axis (if an UltimET is present, the minimum value for K123 is 4).

Then, the ZTE=1 command (refer to [§8.16.2](#)) must be sent to all axes, except to the master-of-acquisition, where the command is ZTE=2. When the acquisition time (taking into account the fulfill of the trig condition, the acquisition time and the pre, post trigger value) is reached in an axis, this node stops all the measures.

On UltimET, the reference time counter is given by the monitoring **M228** or **ML228**.

M	Name	Comment
M228 ML228	Reference time counter	Reference time counter in unit of 25 µs. This counter runs freely and is synchronized with UltimET when TransNET is connected

Example:

Example of a synchronized acquisition between axis 0 and UltimET:

On axis 0, two traces are measured, the real position given by the monitoring ML7 and the measured current force given by the monitoring MF31. The reference time counter given by the monitoring M228 is measured. The acquisition has 10001 points with a sampling time of 100us (the minimum possible with UltimET) and lasts 1000ms. The trigger is raising edge sensitive and is set on the real position (ML7) with a level of 1000000incr. A pre-trigger of 200ms is chosen.

Parameters of axis 0:

```
K120.0=0x1;                                // Traces synchronization mode
K121.0=0x43,0x23,0x0,0x0;                  // Trace register type selection
K122.0=0x7,0x1F,0x0,0x0;                  // Trace register number and depth
K123.0=4;                                    // Traces sampling time selection
K124.0=1;                                    // Trace trigger edge selection
K125.0=4;                                    // Trace trigger mode selection
K126.0=0,0;                                  // Selection of the trigger parameter
K127.0=0,0;                                  // Selection of the trigger level
KF127.0=0.0F,0.0F;                          // Selection of the trigger level
KL127.0=1000000L,0L;                        // Selection of the trigger level
KD127.0=0.0D,0.0D;                          // Definition of the trigger level
K128.0=10001;                               // Number of points to acquire
K129.0=-2000;                               // Pre or post trigger value
```

Parameters of UltimET:

```
*K120=0x1;                                 // Traces synchronization mode
*K121=0x0,0x0,0x3,0x0;                    // Trace register type selection
*K122=0x0,0x0,0xE4,0x0;                  // Trace register number and depth
*K123=4;                                    // Traces sampling time selection
*K124=0;                                    // Trace trigger edge selection
*K125=5;                                    // Trace trigger mode selection
*K126=0,0;                                  // Selection of the trigger parameter
*K127=0,0;                                  // Selection of the trigger level
*KF127=0.0F,0.0F;                          // Selection of the trigger level
*KL127=0L,0L;                             // Selection of the trigger level
*KD127=0.0D,0.0D;                          // Selection of the trigger level
*K128=10001;                               // Number of points to acquire
*K129=0;                                   // Pre or post trigger value
```

As soon as the traces parameters are set, the ZTE commands can be sent. The UltimET is chosen as the master-of-acquisition, thus the command ZTE is sent first to axis 0:

```
ZTE.0=1;                                     // Trace trigger enable
```

Finally, the command to start the acquisition on all nodes is sent to the UltimET:

```
*ZTE=2;                                      // Trace trigger enable
```

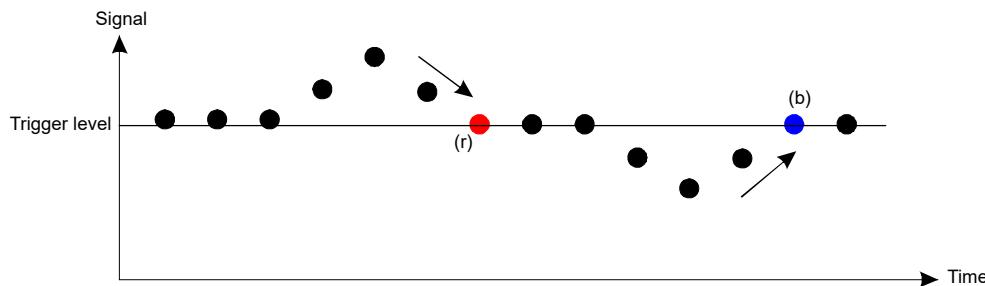
When M131 = 5 or when the bit#27 of M63 is reset, the acquisition is finished and the points can be read.

8.16.4 Trigger condition test and edge detection

The trigger condition is tested at each minimum sampling period (50us for AccurET and 100us for UltimET). It means that if an event fitting the trigger condition occurs and disappears during a sampling period (parameter K123), it will be detected but not necessarily visible on the trace.

For edge sensitive trigger conditions, the following rule is applied: the trigger condition is fulfilled when the signal reaches or crosses the trigger level in the direction of the defined edge (either falling or rising edge or both edges).

The following example illustrates the trigger condition detection: the falling edge detection occurs on the red (r) dot and the rising edge detection occurs on the blue (b) dot.



8.16.5 Traces upload

Each node receiving the ZTE=1 or 2 command, sets its 'trace busy flag' and bit#27 of M63 until it finishes its acquisition or until it gets a ZTE=0 command. When this bit is reset, the values of the traces can be uploaded.

To upload the traces, it is necessary to know the type and format of what was acquired. This information is given by the monitoring M121 describing the traces' type. Depending on the type, the user has to upload each point using T, TL, TF or TD or with the functions available with EDI libraries. There is no translation if the type is not respected, and if the user uploads a trace in double with T, the return values will have no meaning.

For example, the upload of the first trace of the acquisition set in [§8.16.2](#) can be done by asking the value TL0.0, TL1.0, TL2.0,..., TL16000.0 to the controller.

Remark: The content of the T, TL, TF or TD registers may be confusing when the sampling period is smaller than the MLTI duration (400 µs). Some unexpected jitter may be seen, because the acquisition is made in the PLTI and many monitoring registers are updated somewhere during the MLTI execution. Therefore, depending on the MLTI workload, the update time seen in the acquisition may vary from a MLTI to another.

8.16.6 Continuous traces

The aim of the "continuous traces" is to continuously upload data from an endless acquisition. This feature can be part of the process in order to get regular and continuous measurements, or can be used for debugging. It is mandatory to use the dedicated EDI functions for this special mode. Please read the related EDI documentation for more information.

It is to note that a special Wait command is available for the synchronization of the data upload: **WTD** (Wait Traces Data). This wait command will return as soon as all acquisition points are available.

Command	<P1>	<P2>	Comment
WTD[.<axis>]=<P1>,<P2>	First index of the data block	Last index of the data block	Wait the end of the acquisition of the points from <P1> to <P2>

In addition, a warning (bit#30 of M66) is available in case of data corruption. It will be raised when the pointer of the last written point overtakes the last uploaded point. This warning will be cleared as soon as a new acquisition is started (ZTE=1 or 2) or after a RSD command.

8.17 External motor braking control

The AccurET is able to control an external motor brake with a digital output (DOUT). With this feature, the motor brake can be automatically released or closed when the motor is enabled or disabled. A flexible delay is in addition available, allowing a fine tuning of the timing between the brake release and the effective power on of the motor (and vice versa).

8.17.1 DOUT selection

Parameter **K363** allows the user to selects the DOUT to be used for the management of an external motor brake. As soon as K363 is set, the brake control is enabled.

K	Name	Comment
K363	DOUT mask for motor braking	Set the DOUT mask and enable the motor brake control (bit 0-15: state 1 when axis is powered (DOUT1 bit#0 DOUT2 bit#1), bit 16-31: state 0 when axis is powered (DOUT1 bit#16 DOUT2 bit#17))

If the same DOUT is used for several actions, the **ERR CFG DOUT** error (M64=45) occurs. This error is not resettable until the different parameters associated to the corresponding DOUT are not correctly set (only one action is possible per DOUT).

8.17.2 Brake timing

Parameter **K346** defines the delay between the power on/off of the motor and the release/closure of the external brake. K346 contains 2 depths to make possible the setting of a different delay for the motor power on and for the motor power off.

K	Name	Comment	Unit
K346	Delay for motor braking	Defines the delay for the motor braking function. Depth0: Power On; Depth1: Power Off	[mili]

Motor brake sequence:

- Power on process (depth 0): The delay here can be positive (the motor is first enabled) or negative (the brake is first released). A delay of 50 ms is always present during the power on process.
 - If the value is positive, the sequence is:
 - Power on request,
 - Wait time of ~50 ms,
 - Motor is powered on,
 - Wait for the delay defined by K346:0,
 - Open the brake by changing the state of the DOUT defined by K363.
 - If the value is negative, the sequence is:
 - Power on request,
 - Open the brake by changing the state of the DOUT defined by K363,
 - Wait for the delay defined by K346:0,
 - Wait time of ~50 ms,
 - Motor is powered on.
- Power off process (depth 1):
 - Only positive value is allowed, the sequence is:
 - Close the brake by changing the state of the DOUT defined by K363.
 - Wait for the delay defined by K346:0.
 - Motor is powered off.
- On both depth, if K346.<axis>=0, the value is forced to 1.
 Default value is 250 (0,1 second).
 Minimum value is -2500 (-1 second).
 Maximum value is 2500 (1 second).

8.17.3 Motor braking state

Monitoring **M363** allows the user to display the state of the DOUT selected for the management of an external motor braking.

M	Name	Comment
M363	DOUT state for motor braking	Indicates the DOUT state for motor braking (bit 0-15: state 1 (DOUT1 bit#0 DOUT2 bit#1), bit 16-31: state 0 (DOUT1 bit#16 DOUT2 bit#17))

- For motor 0:
 - If DOUT1 is selected (1 when axis powered), M363.0=1 when the brake is open.
 - If DOUT2 is selected (1 when axis powered), M363.0=2 when the brake is open.
- For motor 1:
 - If DOUT1 is selected (0 when axis powered), M363.1=65536 (0x10000) when the brake is open.
 - If DOUT2 is selected (0 when axis powered), M363.1=131072 (0x20000) when the brake is open.

8.17.4 Brake management during the phasing process

The brake management during the phasing process depends on the phasing mode and the command used to initiate the phasing:

- First PWR or INI command
 - With K90=1, the brake is always closed during the phasing.
 - With K90=2, the brake is always open during the phasing.
 - With K90=6 or 7, the brake behavior is managed with the register K364:

K	Name	Value	Comment
K364	External motor braking configuration for phasing K90=6 or 7	0 1	The brake is open during the phasing process (default value) The brake is closed during the phasing process and open after

- AUT command
 - The brake is always open during the phasing.

Caution, when the brake is open during the phasing process, the motor could move or even fall down with vertical linear axis!

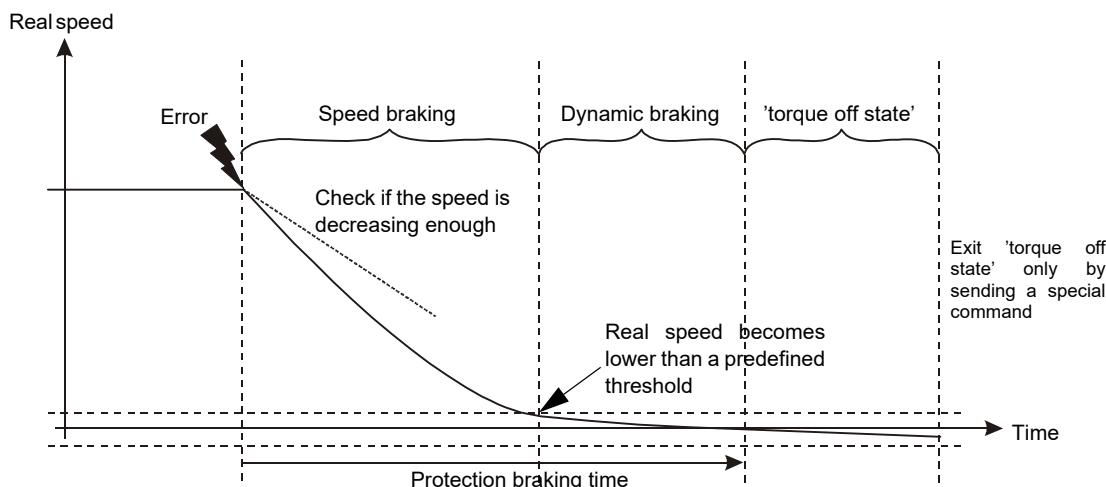
9. Stage protection

The Stage Protection function improves the security of the machine and manages an emergency stop of the motors when it is necessary. The emergency stop of the motor is done in a controlled way and therefore, the braking time and the braking distance are reduced.

When an error occurs on an AccurET, the stage protection function allows the system to brake instead of simply making a power off of the motor. Depending on the kind of error, either a 'speed braking', a 'dynamic braking' or a 'direct switch off of the motor' is performed.

- During a speed braking, a speed loop is activated with its speed reference at 0 through a slope. The motor speed and deceleration are continuously checked. If the real deceleration is not enough or when the speed is under a defined limit, the AccurET brakes then with the dynamic brake. If an error does not allow a speed braking (for example an encoder error), the AccurET brakes directly with a dynamic braking.
- During a dynamic braking, the AccurET brakes the motor by short circuiting the motor phases. The efficiency of this braking type is proportional to the speed. When the programmed braking time is elapsed (minimum 20 ms, maximum 400 ms), the AccurET goes finally to the torque off state. If the error does not allow any braking (for example with a bridge over current error), the AccurET goes directly to the torque off state.
- In the torque off state, the AccurET makes a power off of the motor and stays in this mode, till the reception of a specific command to reset the stage protection function.

If K33 bit#2 is set, the dynamic braking stays enabled even in the torque off state. In such a case, the bridge transistors continue to switch and a voltage is still present on the motor terminals. It is therefore strictly forbidden to touch the system (controller, cable and motor) as long as the mains is not switched off.



The Stage Protection function improves the protection of the machine, but is not meant to protect human. It is inspired by the Machinery Directive and the international standards but does not comply with them and is not certified. This function is not a 'human safety' function but is a 'stage protection' function

Remark: If a dynamic braking is done when the stage protection is activated, the protection against the overcurrent (75% of KF83), protection against i^2t motor (75% of MF67) and protection against i^2t controller (75% of MF167) are not activated.

9.1 Terminology

9.1.1 Type of braking

- **Speed braking:** Situation where the software brakes by setting a speed loop with a speed reference at 0m/s.
- **Dynamic braking:** Situation where an hardware module (FPGA) drives directly the bridge by short-circuiting the phases of the

motor. During the phases short circuit, neither the phases current nor the i2t is checked, therefore, the short circuit is only released when the stage protection timer has elapsed.

9.1.2 Error types

Different types of errors have been created to generate different types of actions:

- **Error type 1:**

All errors that are not related to any AccurET hardware problem and not related to position feedback (for example: 'overspeed error', 'DIN2 pressure lost error',...). An error type 1 will initiate a speed braking.

- **Error type 2:**

All other errors that are related to a loss of Position feedback on AccurET (for example: 'Encoder pos lost'). An error type 2 will initiate a dynamic braking.

- **Error type 3:**

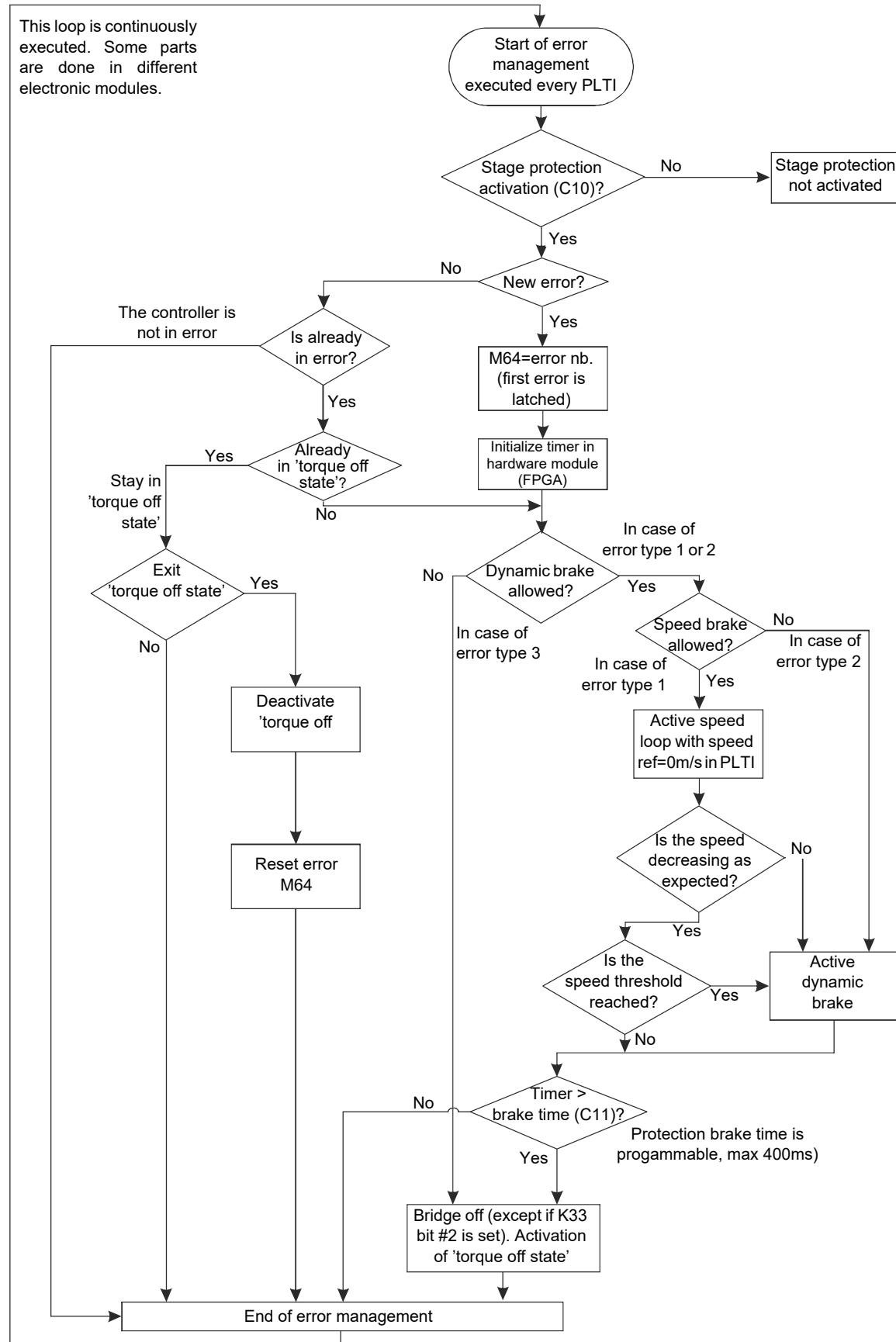
All errors leading to an impossibility to use the power bridge of the AccurET. In that case the power bridge of the AccurET is immediately switched off (for example: 'Hardware overcurrent error',...) After an error type 3, the AccurET must be rebooted, as it is not possible to reset the error with a RST command.

Remark: Refer to [§21.](#) to know the error type of each error which can occur on the AccurET controller

9.2 Restriction

- There is no speed braking phase for the stepper in open loop (K78=1). It directly goes into dynamic braking.
- The current reference (K63!=0) is not supported.

9.3 Principle diagram



When the stage protection function is enabled (C10=1 or 2),

- the RST or PWR command will not reset the error (given by the monitoring M64)
- the command which exit the 'torque off state' mode will also reset the error (given by the monitoring M64)
- even for a motor in power off, when an error occurs, the braking will be activated by powering on the controller.

9.4 Registers

The following registers are needed to set and monitor the different possibilities associated to the errors:

C / K	Name	Comment
C10	Stage protection activation	Stage protection activation (1 or 2=activated, 0=deactivated). This value is taken into account only at the power on of the controller: 1 = brake with a step on the speed reference (step to 0) to stop as fast as possible in case of error. This step with PI regulator (KF14 and KF15) creates current reference saturation up to IPEAK register (KF60). 2 = brake with a ramp defined by KL206:1 on the speed. KF15 must different than 0 to allow the regulator to follow properly the speed ramp reference.
C11	Stage protection brake time	Stage protection brake time in number of [10ms] (min 2=>20ms, max 40=>400ms). This parameter is taken into account only at the power on of the controller.
KF14	Speed loop proportional gain	Gives the proportional gain value of the speed loop.
KF15	Speed loop integrator gain	Gives the integrator gain value of the speed loop.

- DOUT mask

C	Name	Comment
C12	DOUT mask stage protection end braking	DOUT mask stage protection indicating that the AccurET has finished braking (speed is lower than predefined threshold & stage protection braking time has elapsed). This parameter is taken into account only at the power on of the controller.
C13	DOUT mask stage protection short circuit relay	DOUT mask stage protection driving an external short circuit relay after the speed braking. This parameter is taken into account only at the power on of the controller.

The common parameters C12 and C13 are 32-bit integer registers and have the following mask:

bit#31 to 20	bit#19	bit#18	bit#17	bit#16	bit#15 to 4	bit#3	bit#2	bit#1	bit#0
All 0	DOUT2 of axis 1 (to be reset) ¹	DOUT1 of axis 1 (to be reset) ¹	DOUT2 of axis 0 (to be reset) ¹	DOUT1 of axis 0 (to be reset) ¹	All 0	DOUT2 of axis 1 (to be set) ²	DOUT1 of axis 1 (to be set) ²	DOUT2 of axis 0 (to be set) ²	DOUT1 of axis 0 (to be set) ²

Remark:

1: 'To be reset' means that the DOUT associated to the function will be reset.

2: 'To be set' means that the DOUT associated to the function will be set.

When a DOUT (set by C12) is configured to indicate that both 'torque off state' and low speed have been reached, the DOUT will be driven **only after** both axes have reached the 'torque off state' and the low speed.

When a DOUT (set by C13) is configured to drive a phase short circuit external relay, the short-circuit state will come once both axes have finished the speed braking.

It is not allowed to use the parameter K357 (refer to [§7.4.3.3](#)) to activate an external relay if C10=1 or 2 (because the speed loop will be activated during the error).

- CDIN mask

C	Name	Comment
CL211	CDIN mask for error 211	CDIN mask used to generate error 211 (available at depth 0). This parameter is taken into account only at the power on of the controller when C10=1 or 2. It is taken into account immediately when C10=0.
CL212	CDIN mask for error 212	CDIN mask used to generate error 212 (available at depth 0). This parameter is taken into account only at the power on of the controller when C10=1 or 2. It is taken into account immediately when C10=0.
CL213	CDIN mask for error 213	CDIN mask used to generate error 213 (available at depth 0). This parameter is taken into account only at the power on of the controller when C10=1 or 2. It is taken into account immediately when C10=0.

The common parameters CL211, CL212 and CL213 have the following mask:

bit#63 to 52	bit#51	bit#50	bit#49 to 45	bit#44	bit#43	bit#42	bit#41	bit#40	bit#39 to 35	bit#34	bit#33	bit#32
All 0	DIN10 of axis 1 (state 0)	DIN9 of axis 1 (state 0)	All 0	DIN3 of axis 1 (state 0)	DIN2 of axis 1 (state 0)	DIN1 of axis 1 (state 0)	DIN10 of axis 0 (state 0)	DIN9 of axis 0 (state 0)	All 0	DIN3 of axis 0 (state 0)	DIN2 of axis 0 (state 0)	DIN1 of axis 0 (state 0)
bit#31 to 20	bit#19	bit#18	bit#17 to 13	bit#12	bit#11	bit#10	bit#9	bit#8	bit#7 to 3	bit#2	bit#1	bit#0
All 0	DIN10 of axis 1 (state 1)	DIN9 of axis 1 (state 1)	All 0	DIN3 of axis 1 (state 1)	DIN2 of axis 1 (state 1)	DIN1 of axis 1 (state 1)	DIN10 of axis 0 (state 1)	DIN9 of axis 0 (state 1)	All 0	DIN3 of axis 0 (state 1)	DIN2 of axis 0 (state 1)	DIN1 of axis 0 (state 1)

Remark: State 1 means that the DIN associated to the function will react on a '1' state.

State 0 means that the DIN associated to the function will react on a '0' state.

Axis 0 corresponds to the first axis and axis 1 to the second one of the controller.

- **Stage protection**

K	Name	Comment	Units
K380	Stage protection threshold low speed	Stage protection threshold low speed. This value is taken into account only at the power on of the controller.	[dsi]

This parameter is used by an hardware module (FPGA) of the AccurET to check the real speed every 10ms. If this module determines that the speed is lower than the threshold, the following consequences can occur:

- The speed braking through the speed loop is replaced by a dynamic braking by short circuiting the phases of motor.
- The protection DOUT defined by C12 is switched (only if the time has reached the value given by C11).
- The protection DOUT defined by C13 is switched to activate an optional external relay for short circuiting the phases of the motor.

Remark: Depending on the interpolation factor of the encoder (given by K77), it could appear that K380 is set too high or too low to be taken into account in the hardware module (FPGA). In that case, the **STG PROT SETTING** error (M64=215) will occur at the boot of the AccurET. This error is then not resettable by the RST command. In that case, the user must change either the parameter K380 or K77.

K	Name	Comment	Units
K381	Stage protection minimum deceleration	Stage protection minimum deceleration. This value is taken into account only at the power on of the controller.	[dai]

This parameter is used by the hardware module (FPGA) of the AccurET to check the evolution of the speed every 10ms. If this module determines that the speed is not decelerating enough compared to the value of K381, the following consequences can occur:

- The speed braking through the speed loop is replaced by a dynamic braking by short circuiting the phases of motor.
- The protection DOUT defined by C13 is switched to activate the external relay for short circuiting the phases of the motor.

Remark: Depending on the interpolation factor of the encoder (given by K77), it could appear that K381 is set too high or too low to be taken into account in the hardware module (FPGA). In that case, the **STG PROT SETTING** error (M64=215) will occur at the boot of the AccurET. This error is then not resettable by the RST command. In that case, the user must change either the parameter K381 or K77.

M	Name	Comment
M274	Stage protection current mode	Stage protection current mode. 0: stage protection disabled 1: stage protection enabled brake with a step on the speed reference (step to 0) 2: stage protection enabled brake with a ramp defined by KL206:1 on the speed. This monitoring is initialized only at the power on of the AccurET.

9.4.1 Stage protection setting

Here is a guideline to help the user to set the stage protection:

- Set the DIN or DOUT common parameters C12, C13, CL211, CL212 and CL213 if needed
- Set the speed threshold given by the parameter K380 according to the speed limit from which the system is no more dangerous
- Set the parameter K381 with the expected minimum deceleration
- Set the parameter KF14 with an initial value, for example with the derivative coefficient from the position loop (KDP)
- Set the parameter KF15 to 0 at the beginning
- Set the common parameter C11 with the time needed by the system to brake in the worst case
- Set the common parameter C10 (to 1 or 2), save all the parameters and reboot the controller
- With a compiled sequence, test the stage protection setting and check if the real speed (M11) and the real current (MF31) are optimum and do not have too many ripples. If needed adjust the parameters KF14 and KF15.

9.4.2 Error reset

To exit the 'torque off state' and to reset the error, the **RST** (ReSeT) command (RST.<axis>=123) must be used.

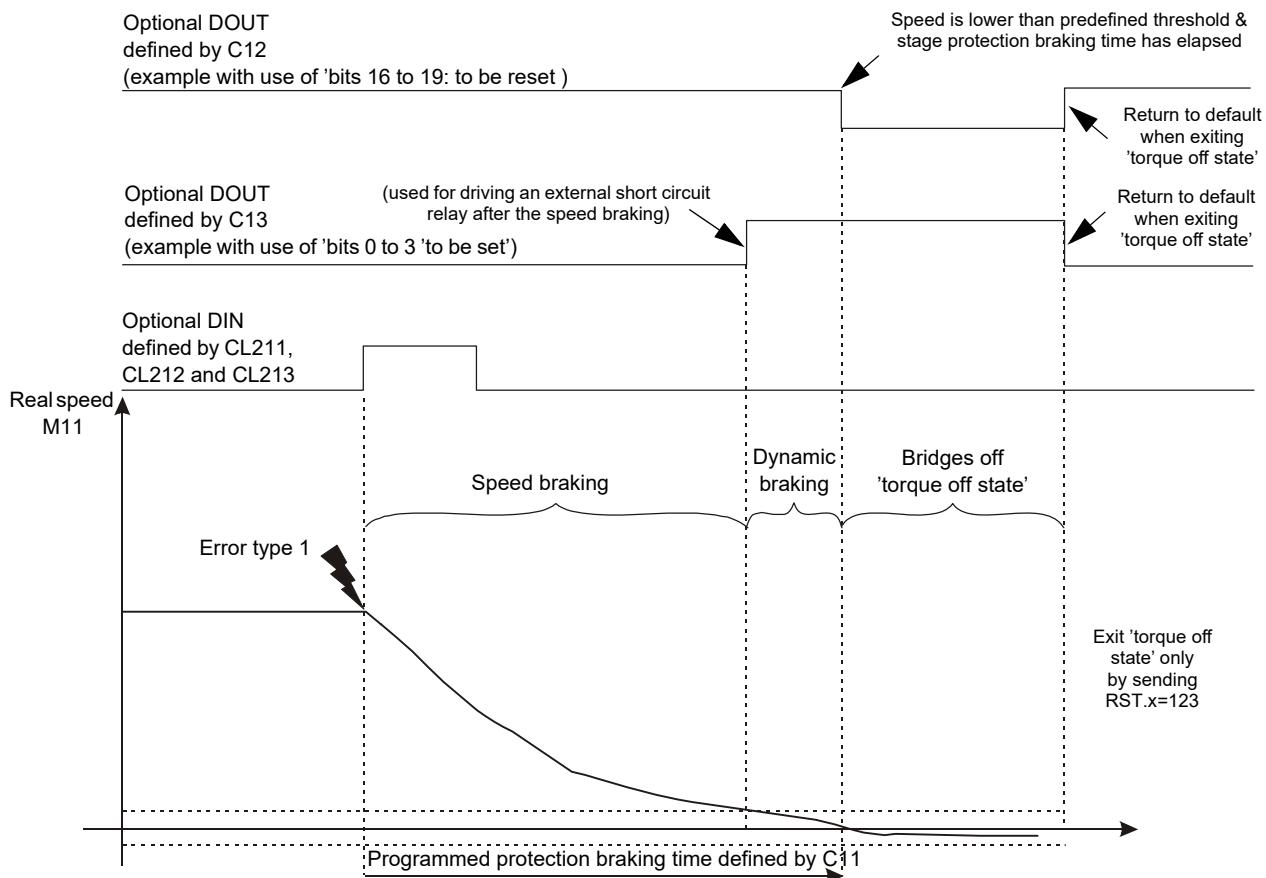
Command	Comment
RST.<axis>[=<P1>]	Exit the 'torque off mode' in stage protection mode with the parameter <P1>=123

9.5 Error cases

The next paragraphs describes several situations with error types 1, 2 or 3. Refer to [§21](#) to know the error type level associated to each error.

9.5.1 Example with speed braking due to an error type 1

- The user opens the cabinet of the machine (error type 1). The AccurET receives an external information on a digital input and goes to **CFG EXTx ERR** error (M64=211, 212 or 213).
- The AccurET software brakes by switching on the speed loop with a speed reference set to 0m/s (therefore, the speed is decreased by a current which is limited by KF60).
- When the speed is lower than the predefined speed (given by K380), then the hardware module (FPGA) activates the dynamic braking and a DOUT (optionally set by a mask) to drive an external motor phase short circuit relay.
- After the pre-programmed protection braking time, the hardware module (FPGA) switches off the power bridges (K33 bit#2 is not set in this example).
- To exit the 'torque off state' and to reset the errors, the RST.!=123 command must be sent to the AccurET (the RST command used without this value will not unlock the 'torque off state').



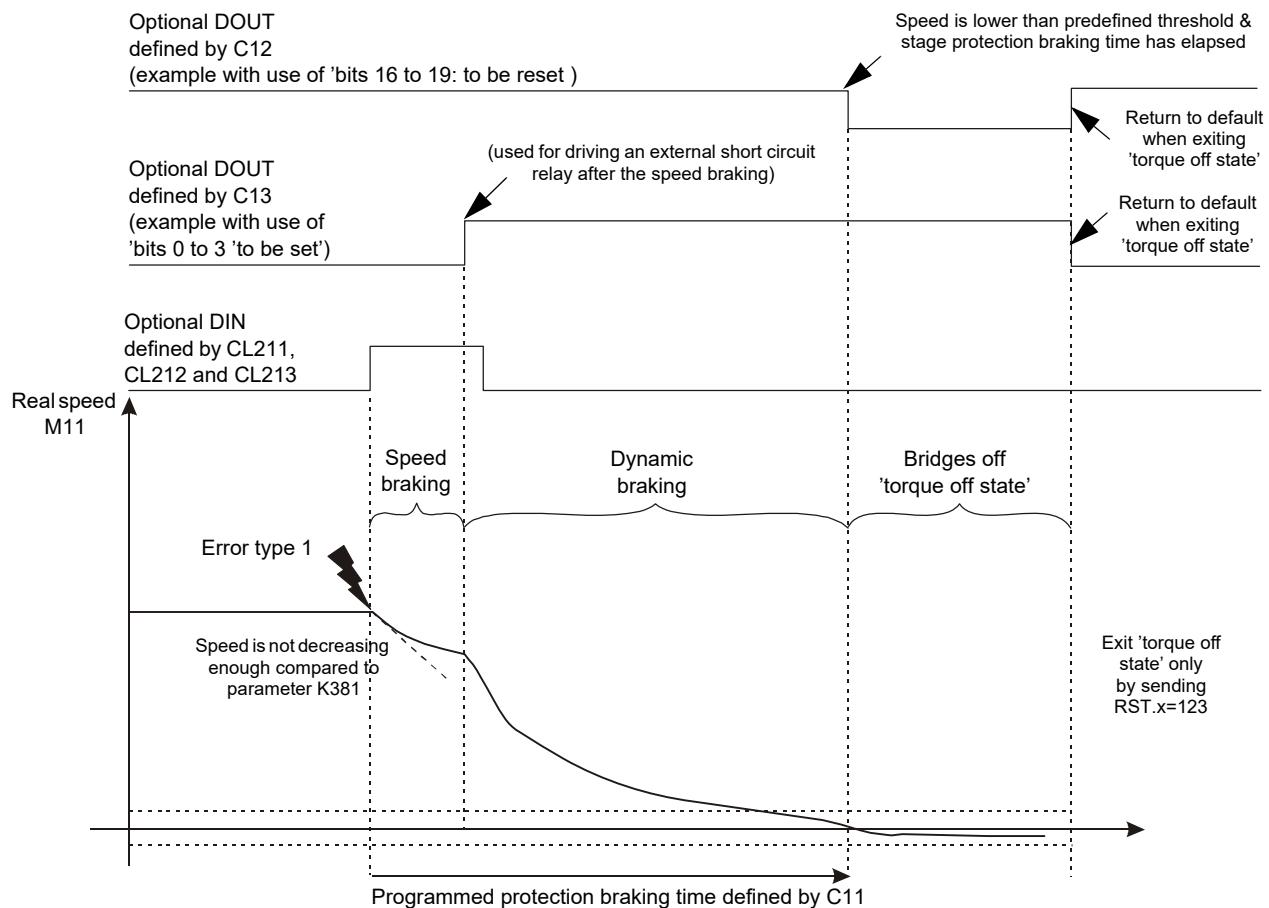
Remark: The DOUT which are optional and set through the common parameters C12 and C13, are managed by the hardware inside the hardware module (FPGA) for a maximum of reliability (without needing any processing in the program).

The three DINs which can be optionally set through the common parameters CL211, CL212 and CL213 are used to trigger an error type 1 (for example: low pressure error, vacuum error,...).

The user must correctly set the values of the parameters KF14 and KF15. To do so, a movement and an error must be generated. Thanks to the Scope of ComET, the monitorings M11 (real speed) and MF32 (current reference) should be monitored together to see the effect of the speed loop setting on the current. An easy solution to deliberately generate an error type 1 is to send the ERR.x command through the Terminal of ComET.

9.5.2 Example with speed braking and dynamic braking due to an error type 1

- The user opens the cabinet of the machine (error type 1). The AccurET receives an external information on a digital input and goes to **CFG EXTx ERR** error (M64=211, 212 or 213).
- The software brakes by switching on the speed loop with a speed reference set to 0m/s (therefore, the speed is decreased by a current which is limited by KF60).
- AccurET checks if the real speed (M11) is decreasing fast enough compared to the value of the parameter K381. If it is not the case, the dynamic braking is activated.
- After the pre-programmed protection braking time, the hardware module (FPGA) switches off the power bridges (K33 bit#2 is not set in this example).
- To exit the 'torque off state' and to reset the errors, the RST.!=123 command must be sent to the AccurET (the RST command used without this value will not unlock the 'torque off state').



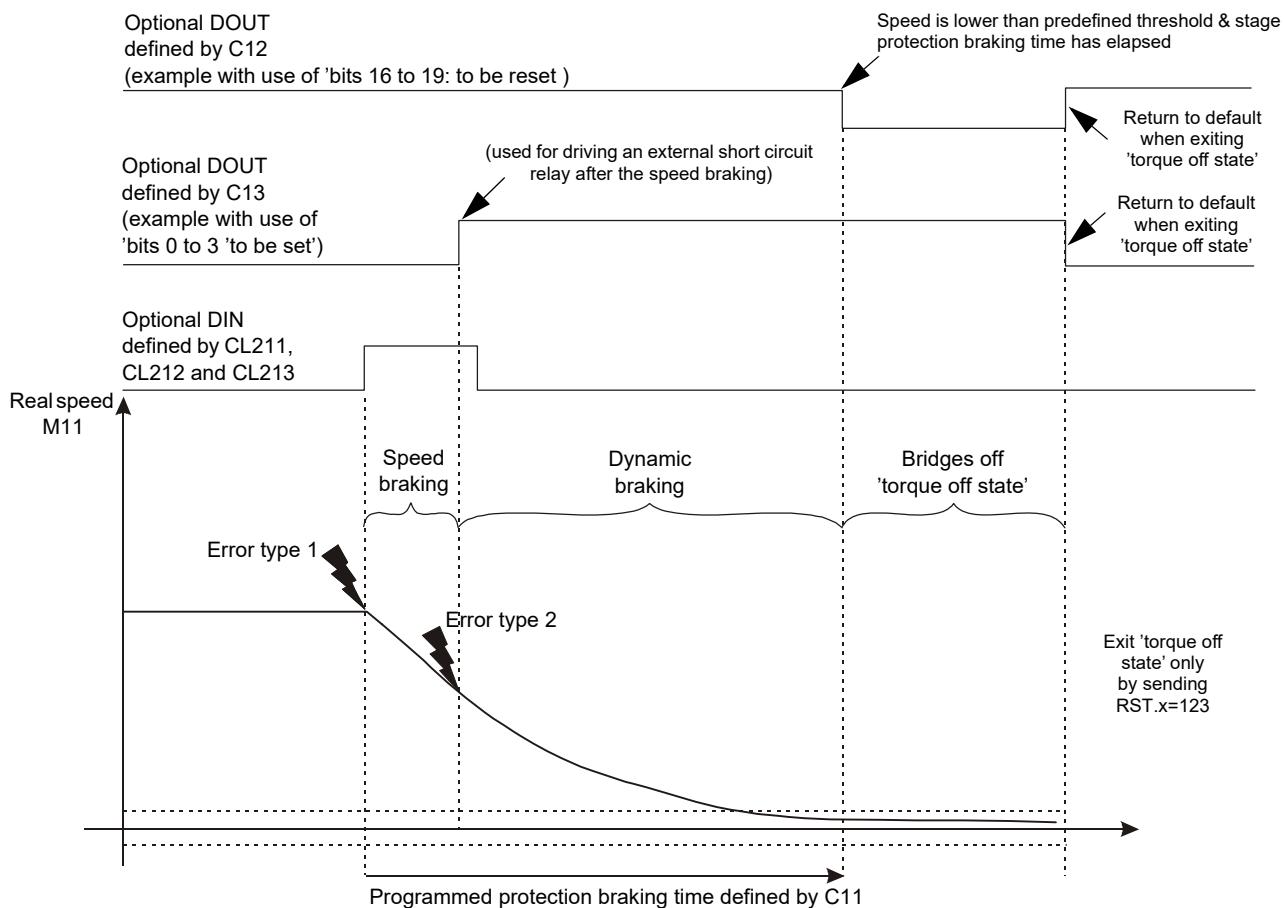
Remark: The DOUT which are optional and set through the common parameters C12 and C13, are managed by an hardware module (FPGA) for a maximum of reliability (without needing any processing in the program).

The three DINs which can be optionally set through the common parameters CL211, CL212 and CL213 are used to trigger an error type 1 (for example: low pressure error, vacuum error,...).

The user must correctly set the values of the parameters KF14 and KF15. To do so, a movement and an error must be generated. Thanks to the Scope of ComET, the monitorings M11 (real speed) and MF32 (current reference) should be monitored together to see the effect of the speed loop setting on the current. An easy solution to deliberately generate an error type 1 is to send the ERR.x command through the Terminal of ComET.

9.5.3 Example with speed braking and dynamic braking due to an error type 1 and 2

- The user opens the cabinet of the machine (error type 1). The AccurET receives an external information on a digital input and goes to **CFG EXTx ERR** error (M64=211, 212 or 213).
- The software brakes by switching on the speed loop with a speed reference set to 0m/s (therefore, the speed is decreased by a current which is limited by KF60).
- An **ENCODER POS LOST** error (M64=21) is detected (error type 2), therefore the dynamic braking is activated.
- After the pre-programmed protection braking time, the hardware module (FPGA) switches off the power bridges (K33 bit#2 is not set in this example).
- To exit the 'torque off state' and to reset the errors, the RST.!=123 command must be sent to the AccurET (the RST command used without this value will not unlock the 'torque off state').



Remark: When an **ENCODER POS LOST** error (M64=21) occurs, the AccurET must be rebooted because the position is lost.

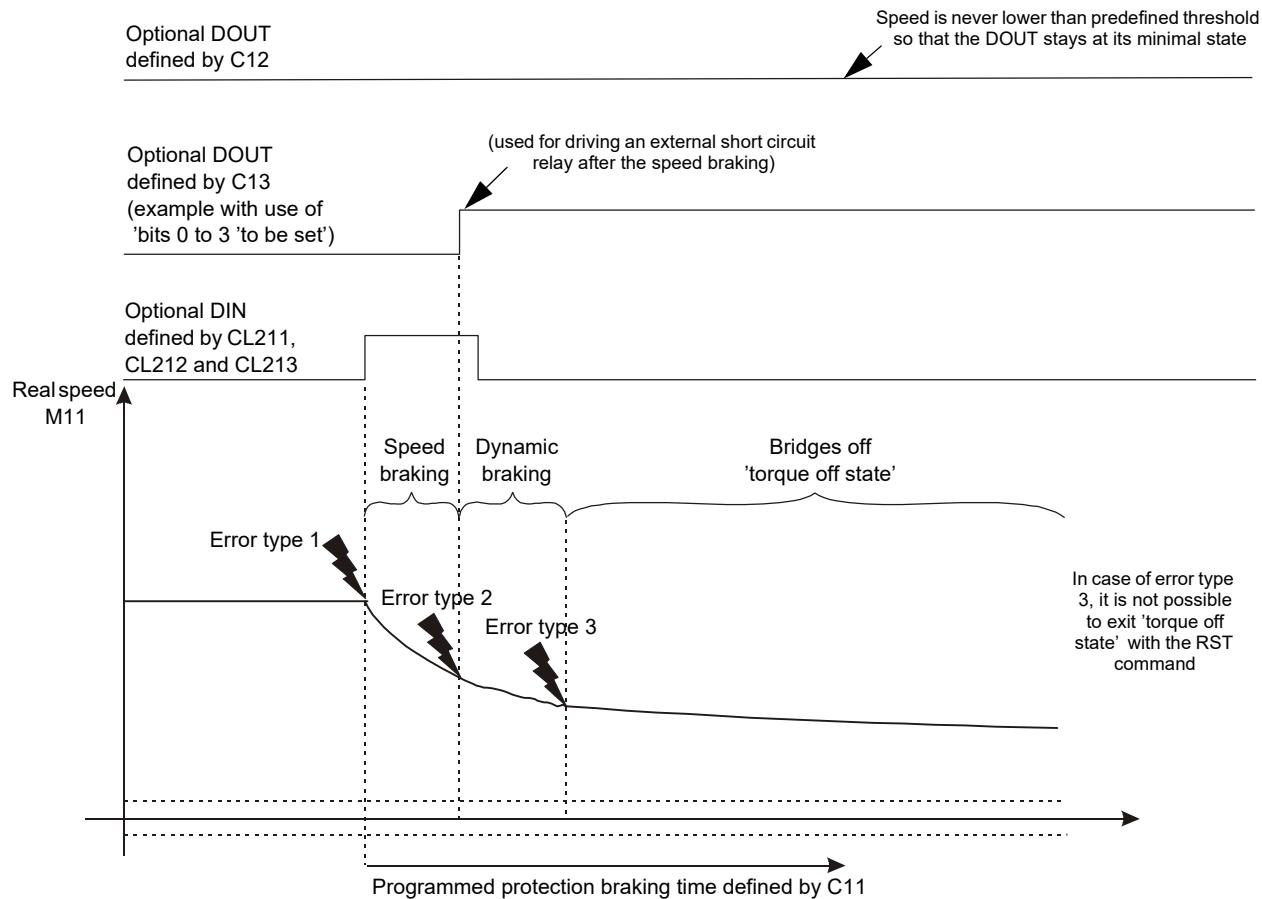
The DOUT which are optional and set through the common parameters C12 and C13, are managed by the hardware inside the hardware module (FPGA) for a maximum of reliability (without needing any processing in the program).

The three DINs which can be optionally set through the common parameters CL211, CL212 and CL213 are used to trigger an error type 1 (for example: low pressure error, vacuum error,...).

The user must correctly set the values of the parameters KF14 and KF15. To do so, a movement and an error must be generated. Thanks to the Scope of ComET, the monitorings M11 (real speed) and MF32 (current reference) should be monitored together to see the effect of the speed loop setting on the current. An easy solution to deliberately generate an error type 1 is to send the ERR.x command through the Terminal of ComET.

9.5.4 Example with speed braking and dynamic braking due to an error type 1, 2 and 3

- The user opens the cabinet of the machine (error type 1). The AccurET receives an external information on a digital input and goes to **CFG EXTx ERR** error (M64=211, 212 or 213).
- The software brakes by switching on the speed loop with a speed reference set to 0m/s (therefore, the speed is decreased by a current which is limited by KF60).
- An **ENCODER POS LOST** error (M64=21) is detected (error type 2), therefore the dynamic braking is activated.
- A Hardware overcurrent error is detected (error type 3), therefore the power bridge is immediately switched off without waiting for the end of the 200ms! The AccurET is then in 'torque off state'.
- With an error type 3, it is not possible to reset the error nor to unlock the 'torque off state'. The AccurET must be rebooted



Remark: The DOUT which are optional and set through the common parameters C12 and C13, are managed by the hardware inside the hardware module (FPGA) for a maximum of reliability (without needing any processing in the program).

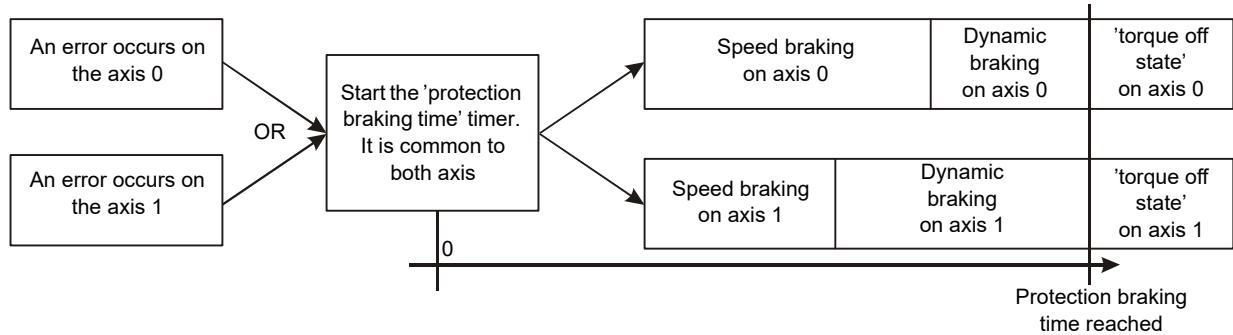
The three DINs which can be optionally set through the common parameters CL211, CL212 and CL213 are used to trigger an error type 1 (for example: low pressure error, vacuum error,...).

The user must correctly set the values of the parameters KF14 and KF15. To do so, a movement and an error must be generated. Thanks to the Scope of ComET, the monitorings M11 (real speed) and MF32 (current reference) should be monitored together to see the effect of the speed loop setting on the current. An easy solution to deliberately generate an error type 1 is to send the ERR.x command through the Terminal of ComET.

9.6 Error management in protection mode

On an AccurET, the stage protection is activated **for both axes at the same time** (through the common parameter C10). Therefore, if one of the two axes detects an error, it will start braking on both axes of the AccurET. In practice, the axis where the error occurs, will generate a **STG PROT OTHERAXI** error (M64=214) on the other axis. The second axis will then brake as well.

The start of the braking is simultaneous on both motors, but the monitoring of braking phases is done separately on each axis.



Remark: Even if one of the axes is in power off, the braking will be activated by powering on the power bridges of both axes!
 Axis 0 corresponds to the first axis and axis 1 to the second one of the controller.
 In gantry mode (C245=1 or 2), to apply simultaneously the same braking forces on both axes, the dynamic braking occurs **at the same time on both axes** of the gantry.

10. Real-time channels on TransnET

10.1 Principle

In some application it is necessary to transmit data between the ETEL devices. There are two ways to share the data between the AccurET, the UltimET and the PC:

- by sending commands from an UltimET sequence or in a PC application, that reads a controller register and writes its value in a register of another controller.
- by continuously sharing data through the TransnET with specific commands introduced in this chapter.

This continuous data sharing, also called Real-Time Values (RTV), can be done between:

- the UltimET and AccurET controllers
- the controllers themselves
- the application PC and the controllers (with UltimET PCI only).

The TransnET bus works with time cycles of 100µs. Each 100µs, a TransnET frame starts from the UltimET (TransnET bus master) and is shared between the different connected controllers. A part of the TransnET frame is made up of slots (a slot is 32-bit data word) used for cyclic data transfer at each TransnET cycle.

The following information may be present on this cyclic data range:

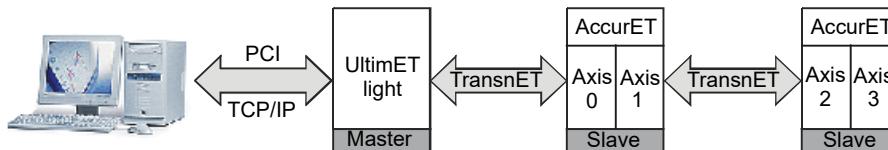
- Axes status information (given by the monitoring M63)
- Monitoring requests (AccurET to UltimET, 4 slots per axis)
- User defined Real-Time Values (RTV)
- ...

There are two types of RTV:

- RTV0: they are managed by the UltimET and described in this chapter (refer to [§10.3.1](#)).
- RTV1: for future use

The number of RTV0 slots depends on the number of present axes: 298 slots – (4 x number of axes (for monitoring request)) - (2 x number of interpolated axes).

Example:



The number of RTV0 is:

- Without interpolation: $298 - (4 \times 4) = 282$
- With interpolation on axes 2 and 3: $298 - (4 \times 4) - (2 \times 2) = 278$

10.2 Data sharing

10.2.1 Between controllers

The read or write on these cyclic data slots is done through sharing data functions (refer to [§10.3.2](#)). The user has to request UltimET for free RTV0 slots before using these data sharing functions. This kind of functions allows one controller to have access to another controller registers in real time.

The EDI interface and sequences provide access to all these functions. The request of RTV0 slots has to be done either on the PC application or from an UltimET sequence. Refer to the '**EDI User's Manual**' for more information about the complete list of the DSA functions and example describing how to use RTV using EDI functions.

10.2.2 Between controllers and the PC application (UltimET Light PCI only)

When using real-time operating systems (for those where EDI is supported or without EDI), this is possible to have access to the TransnET cyclic data slots at the PCI interface. EDI offers a set of functions allowing the sharing of real-time cyclic data with the controllers. Refer to the '**EDI User's Manual**' for more information about the complete list of the DSA functions and example describing how to use RTV using EDI functions. A user defined handler can be called at each TransnET cycle (or multiples) to execute cyclic tasks. Watchdog functionality is provided to warn the user if the user handler execution cannot be processed at the correct frequency (e.g. too much processing load in the handler).

10.2.3 Disappearing of TransnET

Each time a TransnET connection is established, the configuration of the TransnET slots is initialized. Depending on the number of axes, the RTV0 range will be automatically updated. This also means that each time the TransnET disappears and appears again, the assignment of slots is cleared in the UltimET and in the controllers. The configuration of the cyclic data slots has to be redone by the application and/or in the UltimET sequence.

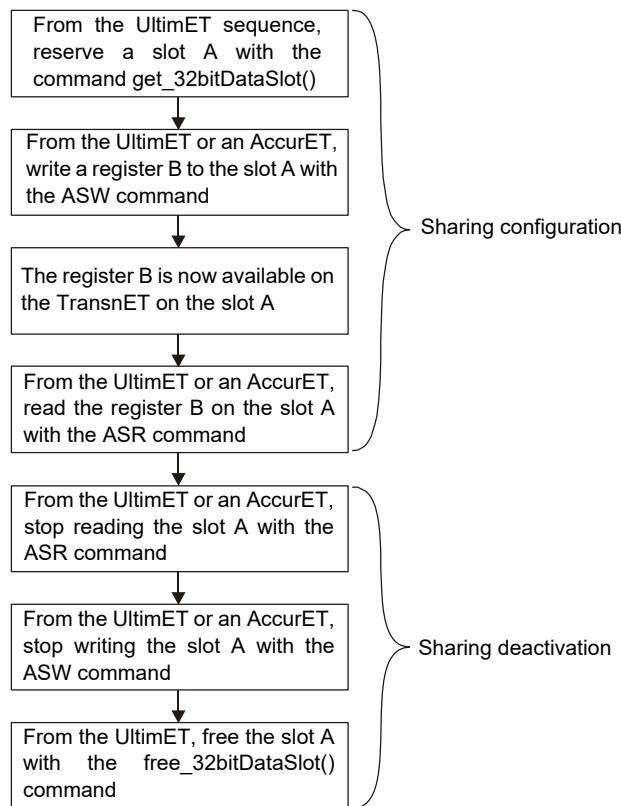
To provide an event to the user, bit#30 of the monitoring *M63 is set as soon as the TransnET connection is interrupted. This status bit is cleared when the TransnET connection is established and when the UltimET reset error command (*RST) has been executed. It is thus necessary to check this event at the application level, and/or in the sequence of the UltimET.

10.3 Interface description

The interface description is presented in three parts:

- Commands related to the management of RTV0 slots
- Commands related to the sharing of the data on the TransnET
- A special UltimET command allowing any command to be forwarded to a mask of axes

Here is a summary on the data sharing principle:



The quantity of slots to be shared in 32-bit integer depends on the parameter **K460** (from firmware 3.14A).

K	Name	Value	Comment
K460	Number of RTV slots	0 1	The maximum number of read slots is 8 and written slots is 7 (K460=0 is the default value) The maximum number of read slots is 16 and written slots is 15

	Read from TransnET	Write on TransnET
AccurET	8 or 16	7 or 15 (*)
UltimET	16	8
PC	All available slots (PC dependent)	All available slots (PC dependent)

(*): 1 slot is already used for sharing the monitoring M63 (controller status) of each controller.

Remark: The new value of the parameter K460 is taken into account once TransnET is reset.

10.3.1 Slot management

The following UltimET sequence commands are used to manage the slots. These commands cannot be used in the 'Terminal' tool of ComET.

- **int get_32bitDataSlot()**
 - get a free TransnET slot
 - no parameter needed
 - returns the value of the slot
 - returns the value -1 if no more slot is available

- **void free_32bitDataSlot(int slot_number)**
 - free the mentioned slot number

The following UltimET monitoring are used to manage the slots:

M	Name	Comment
*M524	Number of free RTV0 slots	Gives the number of free RTV0 slots.
*M525	First RTV0 slot number	Gives the number of the first RTV0 slot.
*M526	Number of RTV0 slots	Gives the number of available RTV0 slots.

Each time the 'get_32bitDataSlot()' command is executed, a new slot is allocated. The following example illustrates what may happen if slots are not properly managed and how to restore the initial situation.

Example:

in this example, the system is made of an UltimET and one AccurET controller (axes 0 and 1). The initial number of RTV0 slots is: *M524 = 298 - (2 x 4) = 290 free RTV0 slots.

```
int slot_axis0;                                // variable that will store the allocated slot
                                                // number

slot_axis0 = get_32bitDataSlot();    // *M524 is now equal to 289
slot_axis0 = get_32bitDataSlot();    // Error: this second request of slot
                                                // allocation must not be done using the same
                                                // variable. *M524 is now equal to 288
```

At this state, the first slot number previously stored in slot_axis0 is no more available.

```
free_32bitDataSlot(slot_axis0);    // One slot, whose value is stored in
                                                // slot_axis0, is free. *M524 is now equal to
                                                // 289
```

The first requested slot cannot be free with the 'free_32bitDataSlot' command. There are two possibilities to free all the RTV0 slots:

- Making the TransnET disappeared, either by powering off/on the UltimET or by sending a RSD.!!=255 command
- Freeing all slots, even those that are not allocated. The following sequence loop does this task:

```
for (i=*M525;i<(*M525+*M526);i++) // free all slots from the first RTV0 slot
                                                // (*M525) to the last (*M525+*M526-1)
{
    free_32bitDataSlot(i);
}
```

10.3.2 UltimET – AccurET functions for sharing data

Sharing data means the exchange of data between controllers and the access of TransnET cyclic data. To do so, the *ASR and *ASW commands (in urgent and normal record formats) are used. With these commands, the request of a free RTV0 slot to the UltimET is mandatory.

10.3.2.1 ASW (ASsign for Write) command

Syntax:

ASW[.<axis>]=register_to_share, slot_lsl [,slot_msl]

Parameter	Comment
register_to_share	Defines the register to be written by the controller (RTV) in the TransnET cyclic data. If this parameter = 0, the register is not written anymore in the TransnET cyclic data
slot_lsl	Defines the slot in the TransnET cyclic data where the controller writes the register, lsl part (RTV)
[slot_msl]	Defines the slot in the TransnET cyclic data where the controller writes the register, msl part (RTV). Only available for 64-bit registers

To stop this continuous write, the ASW=0,slot_lsl [,slot_msl] command must be sent.

Remark: 'lsl' means 'last significant long' and 'msl' means 'most significant long'.

The number of slots already used by the ASW command is given by the monitoring M449. There are 7 or 15 slots reserved for this function for the AccurET (depending on the value of K460). This maximum number of slots for ASW command is available in monitoring M443.

M	Name	Comment
M443	Maximum number of RTV for ASW	Gives the maximum number of RTV slot available for the ASW command.
M448	Used RTV slots by ASW	Gives the slot number associated to a register (the depth corresponds to each of the possible reserved slots)
M449	Counter of written RTV	Gives the number of written RTV by ASW command. By default the monitoring M63 is already put on RTV and can be read through UltimET (*M520:axis_number)

If the ASW command does not have the right format (too many/less arguments, slot number out of range,...), the following errors can occur: **BAD SLOT TRANSNET** (M64=58) for AccurET and **ERROR_SLOT_ASW_BAD_PARAM** (M64=1210), **ERROR_SLOT_NO_FREE_ASW_SLOTS** (M64=1211) **ERROR_SLOT_ALREADY_IN_USE** (M64=1202) for UltimET.

The status of all axes is automatically updated in the UltimET by using cyclic data slots. The nodes status (given by the monitoring M63 of each axis) are available through the monitoring M520.

M	Name	Comment
M520	Copy of the monitoring M63	Copy of the monitoring M63 of all the nodes present on the TransnET. The depths correspond to the node number.

Remark: One slot of each controller is used for transmitting the status information to the UltimET. It means that the number of available slots is reduced by one in each axis for the ASW command. The monitoring M449 is thus by default equal to 1.

10.3.2.2 ASR (ASsign for Read) command

Syntax:

ASR[.<axis>]=monitoring_register, slot_lsl [,slot_msl]

Parameter	Comment
monitoring_register	Defines the monitoring register that is read as a RTV register in the TransnET cyclic data: M450 to M465, or MF450 to MF465, or ML450 to ML457
slot_lsl	Defines the slot number where the data or the lsl part of the data is stored in TransnET
[slot_msl]	Defines the slot number where, only for 64-bit data, the msl part is stored in TransnET

There are 8 or 16 slots reserved for the ASR command for the AccurET (depending on the value of K460). This maximum number of slots is available in monitoring M447.

M	Name	Comment
M447	Maximum number of RTV for ASR	Gives the maximum number of RTV slot available for the ASR command.

Remark: Refer to [§10.3.2.1](#) to have an example.

It is not possible to use at the same time M450 for a slot and MF450 for another slot (same memory range used).

Offset of memory range	M or MF	ML
0	450	450 (msl)
1	451	450 (isl)
2	452	451 (msl)
3	453	451 (isl)
...
14	464	457 (msl)
15	465	457 (isl)

To stop reading a register, the following command must be sent: ASR=monitoring_register, -1, [-1]. If a register reading is not necessary, this command is used to save process time of the PLTI interrupt of the AccurET or the high priority interrupt in UltimET.

If the ASR command does not have the right format (too many/less arguments, slot number out of range,...), the following errors can occur: **BAD SLOT TRANSENTER** (M64=58) for AccurET and **ERROR_SLOT_ASER_BAD_PARAM** (M64=1220) for UltimET.

ASW and ASR commands example:

Sharing and unsharing the real position of the axis 0 to axes 1, 2 and 3 and UltimET. Such a sequence must be used in the UltimET controller only.

```
// Sharing and unsharing of a 64-bit position from axis 0 to axes 1, 2 and UltimET.
// This sequence must be executed in UltimET only.

int axis0_real_position_slot_lsl; // RTV slot for lower 32-bit of position ML1
int axis0_real_position_slot_msl; // RTV slot for upper 32-bit of position ML1

// RTV start
void func10(void)
{
    // Get RTV slots
    axis0_real_position_slot_lsl = get_32bitDataSlot();
    axis0_real_position_slot_msl = get_32bitDataSlot();
    // Axis 0 writes its real position ML1 on TransnET
    ASW.0 = ML1, axis0_real_position_slot_lsl, axis0_real_position_slot_msl;
    // Axes 1, 2 and UltimET start to get ML1 of axis 0 through their own ML453
    *ASR = *ML453, axis0_real_position_slot_lsl, axis0_real_position_slot_msl;
    ASR.(1,2) = ML453.!, -1, -1;
}

// RTV stop
void func20(void)
{
    // Axes 1, 2 and UltimET stop reading the 2 RTV slots and free their own ML453
    *ASR = *ML453, -1, -1;
    ASR.(1,2) = ML453.!, -1, -1;
    // Axis 0 stops writing its real position ML1 on TransnET
```

```

ASW.0 = 0, axis0_real_position_slot_lsl, axis0_real_position_slot_msl;
// Free RTV slots
free_32bitDataSlot(axis0_real_position_slot_lsl);
free_32bitDataSlot(axis0_real_position_slot_msl);
}

```

10.3.3 Real-time channels timing

Here are the different delays and update rates:

Configuration	Delay	Update rate
AccurET <=> AccurET	150 µs (worst case)	100 µs
AccurET <=> UltimET	200 µs (worst case)	100 µs
AccurET (via UltimET PCI) <=> PC	PC dependant	100 µs
UltimET (PCI) <=> PC	PC dependant	100 µs

11. Stage error mapping

11.1 Introduction

This function allows the user to apply an error mapping compensation to the concerned axes. To measure an error, two kinds of information are needed: information about where the stage should be (defined by the target position defined by the customer and measured by the encoders) and information about where the stage really is (measured by an external measurement device such as interferometer measurement, camera...).

Considering an XY stage, errors in X and Y can be measured with a camera and a glass plate marked with patterns (reference grid whose precision will define which measurement precision can be reached). By moving in X and Y, positions given by the axes encoders are compared to positions measured by the camera (pattern recognition and center of gravity measurement). By measuring these errors at various positions, the stage errors are mapped. The resulting data will be two maps (one for X and one for Y errors) of errors that will provide error cartography of the stage.

Two types of errors may be measured: linear and non linear errors. A linear error may be due to a linear deformation of the axis, or to axes which are not orthogonal to each other. A non linear error may be due to the stage mechanical tolerances (components or assembly). Axial linear errors can already be compensated at the encoder scale level (refer to [§8.11](#)).

When measuring these errors, different considerations have to be taken into account:

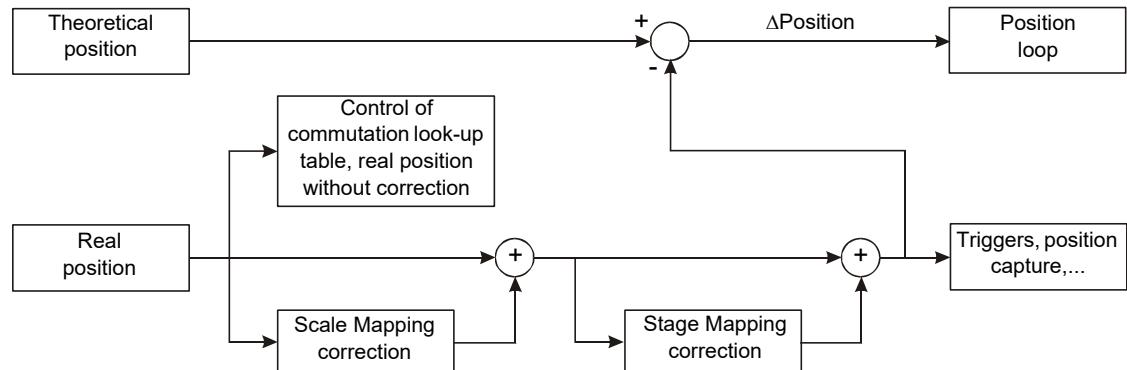
- measurement conditions such as thermal, hygrometry, variations of them... (are they similar to those encountered during the use of the machine?)
- the measurement has to be done at the machine used place
- a burn-in procedure has to be done to warm-up the machine and components
- are the measurements repeatable?
- ...

The stage error compensation is done inside the AccurET controllers. Each axis that has to apply stage error compensation in its direction will contain the stage error mapping table and information to determine and apply this correction (e.g. origin coordinates of the mapped area, step size, number of steps in each dimension...). The information about the position of the other axes is provided through the TransnET.

This mapping of errors can be done on up to 3 dimensions. This functionality is mainly done for a cartesian configuration (linear axes).

11.2 Applied correction

The stage error mapping compensation allows the user to improve the accuracy of his system. The stage error mapping corrections are applied on the real position information (given by the encoder).



The real position after corrections is used as either one of the position loop inputs, or information provided to the user or as an input for a function like 'position trigger' and 'position capture'.

11.3 Stage Mapping configuration

The following information are needed to define a stage error mapping configuration, the following information is requested:

- on which axes a correction must be applied
- according to which axes are these corrections applied
- from which information (registers) are these corrections taken
- the correction table for each corrected axis

Remark: This information is either shared through TransnET or internally if both axes are on the same controller. Each axis calculates its correction based on the source positions and on the data present in the mapping table.

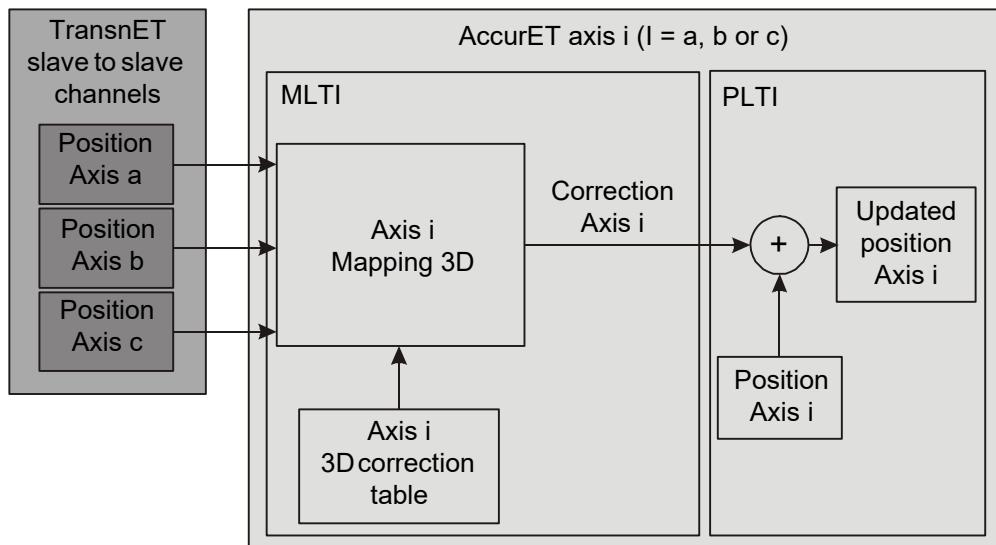
Example with an XY table (2D mapping):

- the user wants to compensate the errors on X and Y
- these errors are compensated according to X and Y ($\Delta X = f_x(X, Y)$ and $\Delta Y = f_y(X, Y)$)
- the source information comes from axes X and Y, and the source position is taken from the monitoring ML17 or M17

11.3.1 Schematics of different configurations

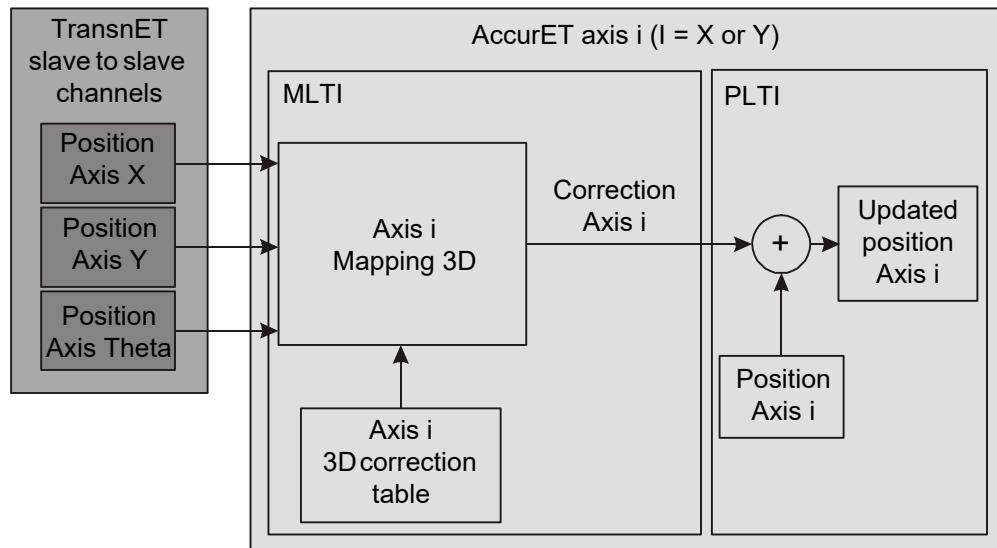
11.3.1.1 3D stage error mapping

- The 3 axes provide their position through TransnET
- The 3 axes correct their real position according to the real positions provided through TransnET



11.3.1.2 X-Y-Theta 3D stage error mapping

- Theta is a rotary axis and its position is considered modulo 1 turn
- The 3 axes provide their position through TransnET
- X and Y axes correct their real position according to the real positions provided through TransnET

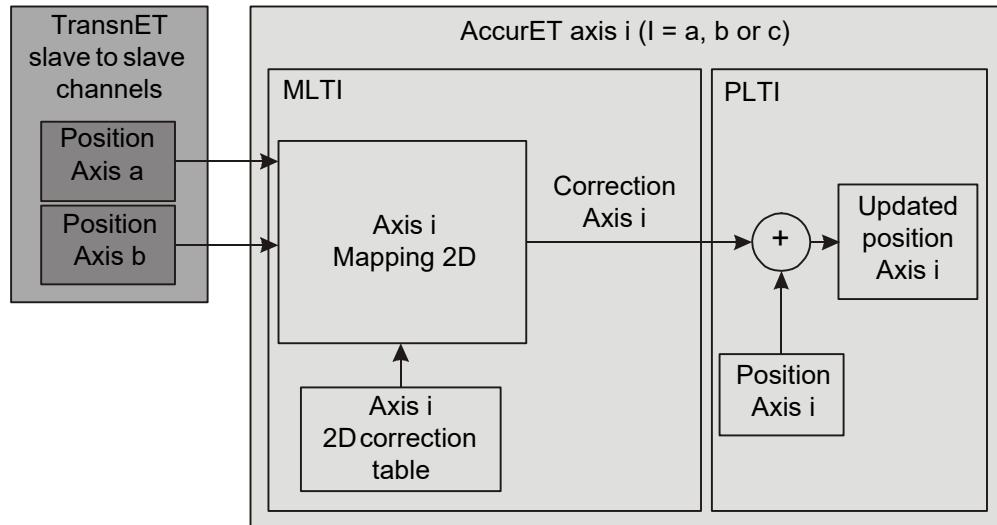


In the mapping table dimension relative to the source cyclic axis, the following rules must be applied:

- The source axis origin is the 0 position
- The error mapping must be done on 1 turn
- The error measurement must start at the 0 position and stop 1 step before 1 turn (e.g. 36 points for a mapping measurement each 10 degree, thus the last measurement is done at 350 degrees). If not, an error will occur (bad Stage Mapping configuration).

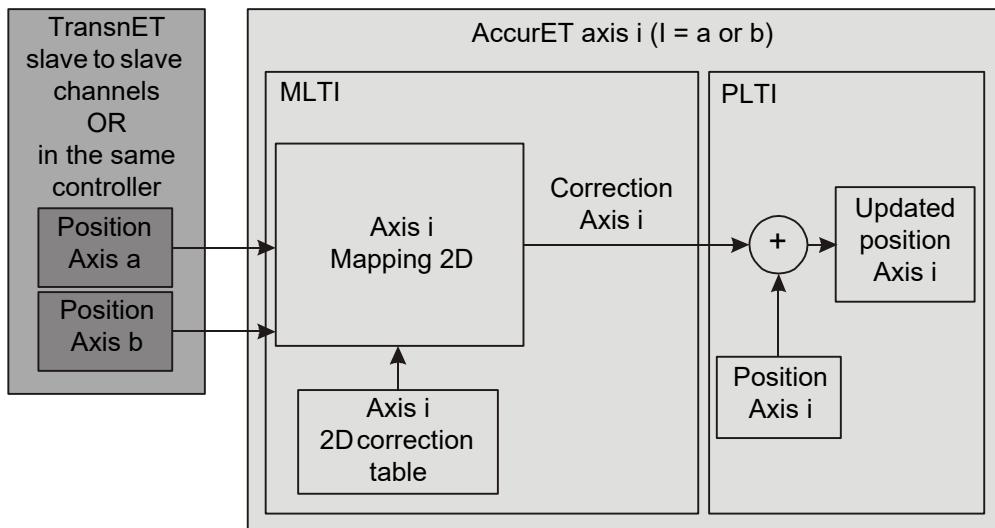
11.3.1.3 2.5D stage error mapping:

- The 2 axes provide their position through TransnET
- 3 axes correct their real position according to the 2 real positions provided through TransnET



11.3.1.4 2D stage error mapping:

- The 2 axes provide their position through TransnET, or not if both axes are on the same controller
- The 2 axes correct their real position according to the real positions provided through TransnET or internally.



Remark: These standard configurations that can be done using the stage error mapping compensation present on AccurET. Other configurations may be done, based on this principle. The stage error mapping is done on a set of coordinates. Between these coordinates, a linear interpolation is done for 1D, 2D and 3D mapping.

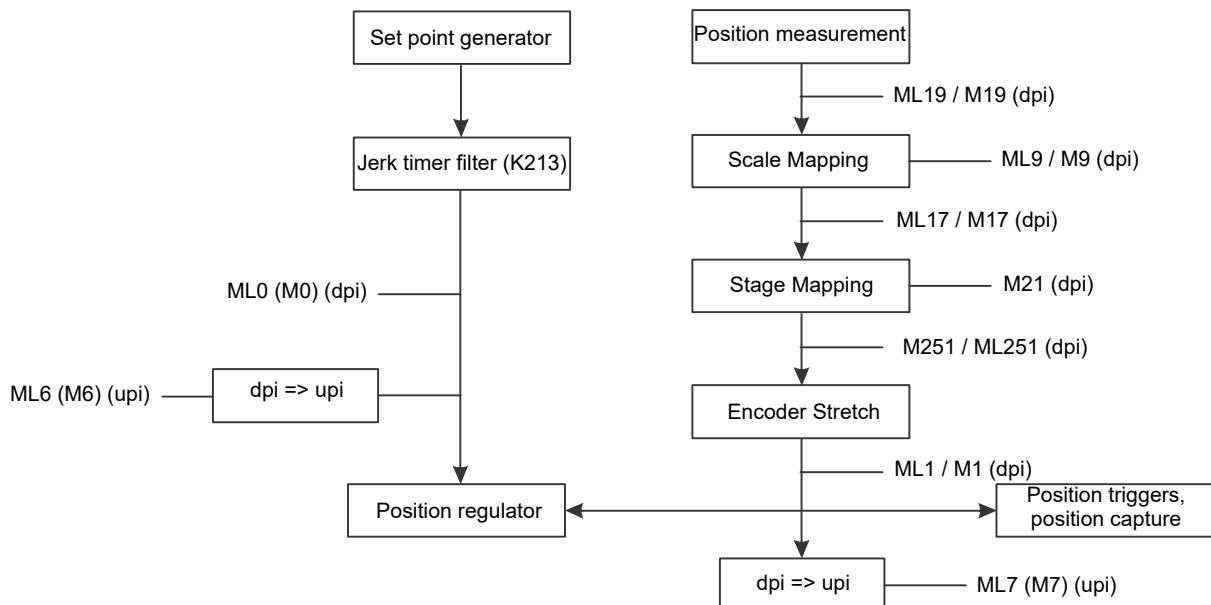
11.4 Activation / deactivation of the stage error mapping

11.4.1 Introduction

The stage error mapping configuration (stored in a configuration file) is downloaded into the controllers through the ComET tool or the EDI function.

Remark: A complete example for downloading and activating a Stage Mapping configuration is available in the EDI package.

The following diagram describes the position of the stage error mapping compensation module as well as the different monitorings which can be used by the user.



The correction is applied after the scale error mapping compensation (refer to §8.11). The measured position (real position) before the stage error mapping correction, can be monitored through the monitoring **M17 (ML17)**. It is used as the source position reference to the stage error mapping. The monitoring **M21** gives the applied correction value and the monitoring **M1 (ML1)** gives the axis real position with the scale and stage error

mapping compensations. This real position is used as an input for the axis position regulator, the position trigger functionality and the position capture functions.

M	Name	Comment	Units
M17 ML17	Real position after Scale Mapping	Gives the real position after Scale Mapping before Stage Mapping. The monitoring M17 corresponds to the LSL part of the monitoring ML17.	[dpi] [rdpi]
M21	Stage Mapping compensation value	Gives the Stage Mapping compensation value.	[dpi] [rdpi]
M1 ML1	Real position	Real position X. Takes the mapping corrections into account, but does not take care about SET command. The monitoring M1 corresponds to the LSL part of the monitoring ML1.	[dpi] [rdpi]

The state of the stage error mapping is given by the monitoring **M330**:

M	Name	Value	bit#	Comment
M330	Stage error mapping state	16 32 64	4 5 6	Current source positions are in the mapping area Stage error mapping configuration is local to this AccurET controller Stage error mapping correction is enabled

The table containing the stage error mapping information (measured corrections) is stored in P registers. The correction values are stored in integer 32-bit format and the unit is in [dpi]. There are 16352 P registers and each P register has 8 depths (0 to 7). Thus 130816 correction values can be stored in each axis. The table is filled as follows: P[index]:[depth]:

P0:0=1st correction	P0:1=2nd correction	...
P1:0=9th correction	P1:1=10th correction	...
		...
		P16351:7=130'816thcorrection

This allows a table size of:

- 3D table: 50 x 50 x 50 correction values (1st dimension x 2nd dimension x 3rd dimension)
- 3D table: 255 x 255 x 2 correction values
- 2D table: 361 x 361 correction values

For a 2D correction table (e.g. 3x2 table), the P registers are filled as follows:

Y/X dimensions	X [0]	X [1]	X [2]
Y [0]	Corr ₀₀	Corr ₁₀	Corr ₂₀
Y [1]	Corr ₀₁	Corr ₁₁	Corr ₂₁

P0:0=Corr00, P0:1=Corr10, P0:2=Corr20, P0:3=Corr01, P0:4=Corr11, P0:5=Corr21

All the data related to the stage error mapping is saved in a specific area of the flash memory. This includes P registers and information related to the mapping configuration. To save and restore this configuration, the SAV and RES commands must be used (refer to [S6](#) for more information).

The activation / deactivation of the stage error mapping can be done in two different ways:

- Full activation / deactivation: this is the case at the start or at the stop of the machine.
- Activation / deactivation correction: this is done when the machine is already in use and the user wants to temporarily deactivate / activate the stage error mapping compensation.

11.4.2 Full activation / deactivation

11.4.2.1 Full activation

The full activation of a stage error mapping configuration is done through the sequence or with the ComET tool (Tools/Advanced/Stage Mapping):

- The stage error mapping configuration is downloaded into the controllers through the ComET tool.
- All information related to the stage error mapping configuration is present in each axis that needs to be corrected. If SAV=9, the parameters of the mapping configuration are downloaded at the power on of the controller
- The following information are needed to activate the compensation on the concerned axes (it is automatically done by clicking on the 'Activate' button of the ComET tool). Activating means:
 - Requesting to UltimET which slots are free to be used for sharing source positions on TransnET:
`dim1_source_position_slot_lsl = get_32bitDataSlot();`
`dim1_source_position_slot_msl = get_32bitDataSlot();`
`...`
 - Sharing, if needed, the source positions on TransnET. The sharing is done using the **ASW** command. It sets slot (offset in the TransnET cyclic data) where the controller writes cyclic data RTV (refer to [§10](#)):

Command	<P>	Comment
ASW.<AXIS>=<P1>,<P2>,[<P3>]	<P1> <P2> <P3>	Defines the register to be written by the controller (RTV) in the TransnET cyclic data. If <P1>=0, the register is not written anymore in the TransnET cyclic data Defines the slot in the TransnET cyclic data where the controller writes the register, LSL part (RTV) Defines the slot in the TransnET cyclic data where the controller writes the register, MSL part (RTV). Only available for 64-bit#registers

`ASW.dim1_source_axis=ML17, dim1_source_position_slot_lsl, dim1_source_position_slot_msl;`

Where `dim1_source_position_slot_lsl` and `dim1_source_position_slot_msl` slots are obtained from `get_32bitDataSlot` (compiled sequences) or `get_32bit_rtv_slot` (EDI) command.

- Sending K365=0 to all corrected axes of the stage error mapping configuration.
- Telling the corrected axes in which TransnET slots (or in local) they will receive the source positions. This is done through the **ASP** command. This command initializes also the stage error mapping configuration on the axes. All axes needing a correction must get this command:

Command	<P>	Comment
ASP.<AXIS>=<P1>,<P2>,<P3>,<P4>,<P5>,<P6>	<P1> <P2> <P3> <P4> <P5> <P6>	Defines the offset in the TransnET DPRAM where the controller reads the LSL of first source of Stage Mapping Defines the offset in the TransnET DPRAM where the controller reads the MSL of first source of Stage Mapping Defines the offset in the TransnET DPRAM where the controller reads the LSL of second source of Stage Mapping Defines the offset in the TransnET DPRAM where the controller reads the MSL of second source of Stage Mapping Defines the offset in the TransnET DPRAM where the controller reads the LSL of third source of Stage Mapping Defines the offset in the TransnET DPRAM where the controller reads the MSL of third source of Stage Mapping

`ASP.corrected_axis=dim1_source_position_slot_lsl, dim1_source_position_slot_msl,
dim2_source_position_slot_lsl, dim2_source_position_slot_msl,
dim3_source_position_slot_lsl, dim3_source_position_slot_msl;`

Where the pairs of the rtv slots provide the address of the dimension 1, 2 and 3 source data.

- If the source data is a 32-bit#data, msl slot value is set to -1.
- If the mapping configuration is local (2D mapping on the same AccurET), all the parameters of the ASP command are set to -1.
- When the mapping configuration is 1D or 2D external, the unused dimensions are to set to -1.
- Sending K365=1 to all corrected axes of the stage error mapping configuration:

K	Name	Value	Comment
K365	Stage error mapping activation	0	Deactivates the stage error mapping
		1	Activates the stage error mapping

Remark: The stage error mapping activation induces an update of the axis measured position (ML1). The theoretical position (ML0) is updated as well (jump of both values), however the motor does not physically move.

11.4.2.2 Full deactivation

The full deactivation (manage by the sequence or the 'Deactivate' button of the ComET tool) of a stage error mapping configuration consists in:

- Sending K365=0 to all corrected axes of the stage error mapping configuration
- Sending the ASP command to all corrected axes with only the first parameter (<P1>) equal to 0 (ASP.<axis>=0) to stop them watching at mapping RTV slots (this also reduces the processor load). In this case the parameters <P2> to <P6> must not be used.
- The TransnET shared data (through the ASW command) is not released. The function cannot ensure that this data was shared only for the stage error mapping.

11.4.3 Activation / deactivation correction

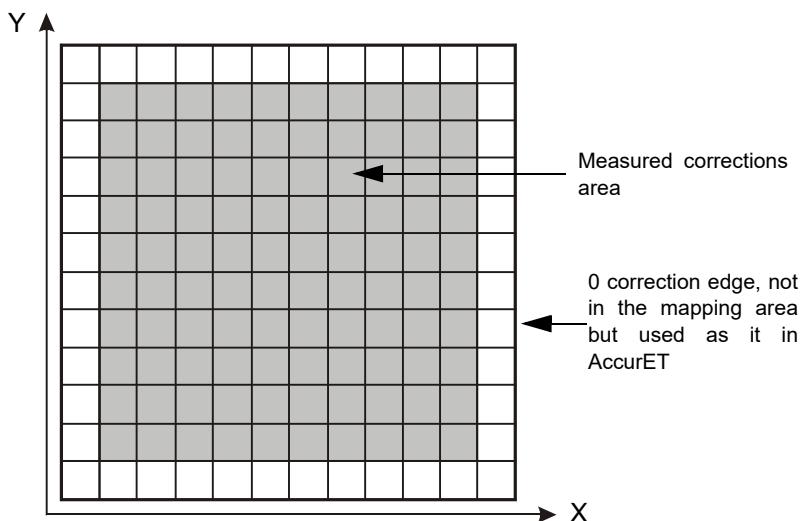
The activation / deactivation correction is done when the machine is already in use and the user wants to temporarily deactivate / activate the mapping compensation. This correction is done through the parameter K365 (refer to [§11.4.2.1](#)).

- This activation / deactivation through the parameter K365 is not related to the sharing of source positions.
- The stage error mapping state is accessible through the monitoring **M330**.

M	Name	Value	bit#	Comment
M330	Stage error mapping state	0x10 0x20 0x40	4 5 6	Current source positions are in the mapping area Stage error mapping configuration is local to this AccurET controller Stage error mapping correction is enabled

11.4.4 Mapping area

The stage error mapping configuration is measured and defined by the user. The border between mapped and not mapped areas should be smoothed by the user to have no correction in the not mapped areas. Anyway, AccurET controllers automatically add a zero correction range all around the mapping area to apply a transition between areas. This transition is done on a one step size in each dimension.



11.4.5 Error management

The following errors (given by M64) may occur related to the stage error mapping function:

- **STAGEMAP ACTIVE** error (M64=75): this error occurs when mapping configuration commands (like ASP,...) or IND command are executed when the stage error mapping compensation is active.
- **STAGEMAP CFG** error (M64=76): this error occurs when the number of mapping steps is wrongly defined (less than 1 step) or when the number of mapping dimensions is incorrect (must be either 1, 2 or 3).
- **STAGEMAP STEP ERR** error (M64=77): this error occurs if one source axis moves more than the step size in 1 MLTI sampling time (400us), for example if a step size is 5mm, the axis velocity must be kept smaller than 12.5m/s.

For all these errors, the mapping compensation is disabled and the full activation must be done again. If the TransnET link is disconnected, the mapping compensation is disabled and the full activation must be done again, except for a 1D and 2D Stage Mapping configuration located in a single controller (both axes in the same controller). In this particular configuration, the mapping compensation is kept active.

11.5 Activation example

Here is an example describing how to manually activate the stage error mapping function. To do so, the following procedure has to be done:

- The stage error mapping configuration is downloaded (and possibly saved) into the controllers through ComET.
- The activation part is done in a sequence:
 - Step 1: check that the controllers concerned by the stage error mapping (source and corrected axes) have their homing done.
 - Step 2: get free slots from the UlimET to share source data between axes.
 - Step 3: share the source registers (given by M321, M322 and M323 for register type, index and depth) from the source axes (given by M320), using the ASW command.
 - Step 4: provide to the corrected axes in which slots they will find the source information (defined in the previous step). The ASP command is used for this purpose.
 - Step 5: final activation of the stage error mapping compensation. This is done by setting the parameter K365 of the corrected axes to 1.
 - Step 5: check that bit# 6 of M330 is set (mapping is properly enabled).

Here is an example showing how to activate a 3D stage error mapping. The concerned axes are:

- Axis X, 0 (axis X calculates its correction based on ML17 of X, Y1 and Z)
- Axis Y1, 2 (axis Y1 calculates its correction based on ML17 of X, Y1 and Z)

- Axis Y2, 3 (axis Y2 calculates its correction based on ML17 of X, Y2 and Z)
- Axis Z, 1 (axis Z calculates its correction based on ML17 of X, Y1 and Z)

Thus the monitoring ML17 of X, Y1, Y2 and Z axes needs to be shared on TransnET. X, Y1, Y2 and Z axes also require the slots for their 3D source positions (ASP command). Then the parameter K365 of all axes is set. To do so, the UltimET sequence code will be (variables are already defined):

```
// Get new free slots from UltimET
slot_X_lsl = get_32bitDataSlot();
slot_X_msl = get_32bitDataSlot();
slot_Y1_lsl = get_32bitDataSlot();
slot_Y1_msl = get_32bitDataSlot();
slot_Y2_lsl = get_32bitDataSlot();
slot_Y2_msl = get_32bitDataSlot();
slot_Z_lsl = get_32bitDataSlot();
slot_Z_msl = get_32bitDataSlot();

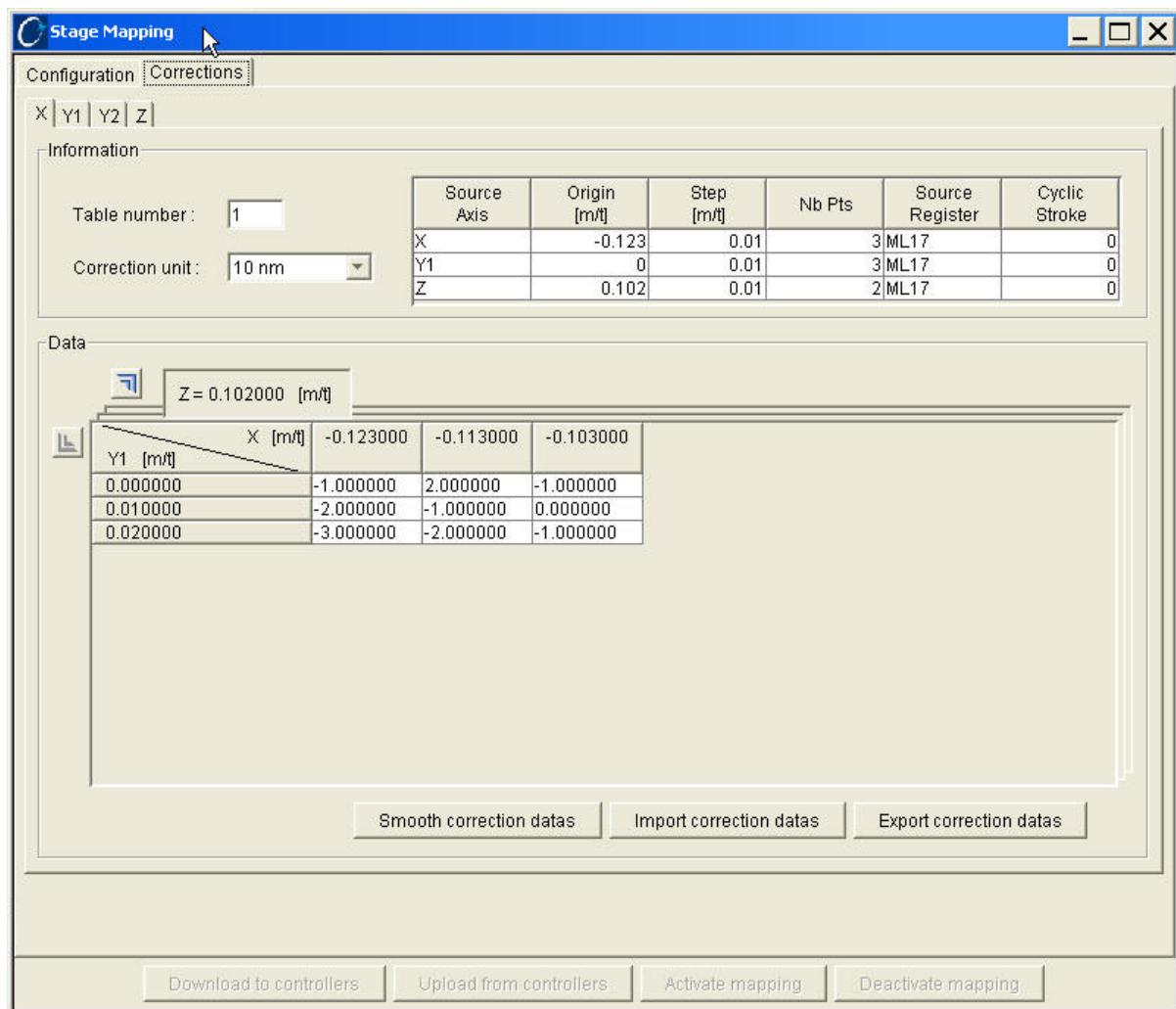
// ask axes to write ML17 to mentioned slots
asw.0=ML17,slot_X_lsl,slot_X_msl;
asw.2=ML17,slot_Y1_lsl,slot_Y1_msl;
asw.3=ML17,slot_Y2_lsl,slot_Y2_msl;
asw.1=ML17,slot_Z_lsl,slot_Z_msl;

// provide slots addresses source positions to each axis
asp.0=slot_X_lsl,slot_X_msl,slot_Y1_lsl,slot_Y1_msl,slot_Z_lsl, slot_Z_msl;
asp.2=slot_X_lsl,slot_X_msl,slot_Y1_lsl,slot_Y1_msl,slot_Z_lsl, slot_Z_msl;
asp.3=slot_X_lsl,slot_X_msl,slot_Y2_lsl,slot_Y2_msl,slot_Z_lsl, slot_Z_msl;
asp.1=slot_X_lsl,slot_X_msl,slot_Y1_lsl,slot_Y1_msl,slot_Z_lsl, slot_Z_msl;

// Activates the stage error mapping compensation on all concerned axes
K365.(0,1,2,3)=1
```

Then, if desired, bit# 6 of the monitoring M330 can be tested to ensure that the correction is applied.

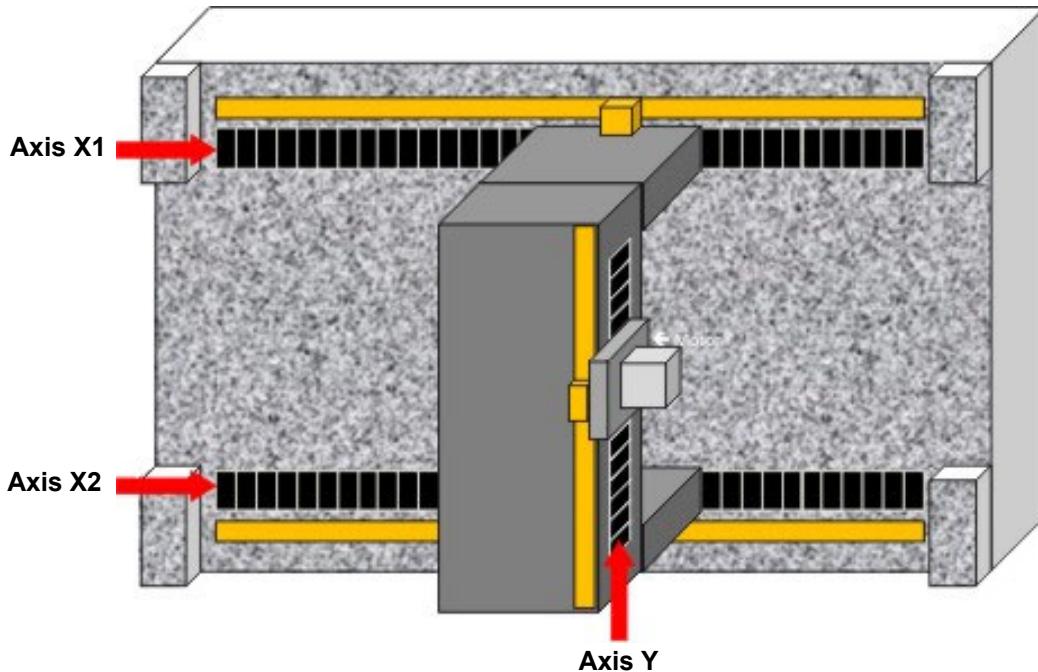
Here is a print corresponding to the 'Stage Mapping' tool of ComET which can automatically generate the above-mentioned sequence code:



12. Gantry control

12.1 Introduction

This function allows the user to manage gantry configurations. To simplify the descriptions of this function, the axes of the gantry are called X1 & X2 and the axis on the beam is the axis Y.



The gantry rules used for this function are as follows:

- The two X axes of a gantry are always connected to the same position controller. The first axis is the master of the gantry (X1) and the second one is the slave (X2).
- The two X axes of a gantry use the same type of encoders and the same type of motors.
- UltimET and TransnET are useful only if advanced gantry feedforward is used.

12.2 Gantry levels

Three different gantry levels have been developed to manage a gantry:

- **Level 0:** X1 and X2 axes have no link between them and no protection is set. It is the state where gantry is not activated. It is the level where the basic setting can be made individually on the axes.
- **Level 1:** X1 and X2 axes have protections enabled. It is useful for the final setting for the gantry configuration.
- **Level 2:** Same as level 1, except that only the master is visible by the application side. At this level, it is not possible to make any setting.

12.2.1 Visibility aspect

The visibility aspect is on two sides:

- Application side: which axes are visible on the customer application side?
- TransnET side: which axes are visible on the TransnET?

- **Level 0**

- On the application side, axes X1 and X2 are visible. The commands could be sent on each axis individually or on both axes together. The gantry mode is not enabled.
- On the TransnET side, both axes X1 and X2 are visible.

- **Level 1**

- On the application side, axes X1 and X2 are visible, with protections enabled between them. The commands must be sent on both axes ($MVE.(X1,X2)=0L;$)
- On the TransnET side, axes X1 and X2 are visible.

- **Level 2**

- On the application side, only the master is visible. The commands must be sent only on the master axis (X1).
- On the TransnET side, axes X1 and X2 are visible.
- It is possible to connect ComET on both axes and to have access to the slave with the scope.
- Some commands are not allowed (on the slave and/or on the master). They can generate an error if they are sent to the position controller.

12.2.2 Level configuration parameters

The common parameter **C245** allows the user to define the gantry level configuration.

C	Name	Comment
C245	Gantry level	Gives the level for the gantry functionality. 0 => no gantry, 1=> gantry protection active, 2=> only 1 axis visible for HMI.

The monitoring **M289** allows the user to know the current gantry level.

M	Name	Comment
M289	Gantry current Level	Gives the current gantry level

The following table shows the transition between the different levels:

	Current level=0	Current level=1	Current level=2
Go to level 0	-	On-line	Need to be saved and rebooted
Go to level 1	On-line	-	Need to be saved and rebooted
Go to level 2	Need to be saved and rebooted	Need to be saved and rebooted	-

12.3 Gantry axes configuration

The parameter **K312** allows the user to set the axis number for the master (X1) of the gantry. This number is always the first axis of an AccurET controller. In level 1 and 2, if K312 is not equal to the first axis number of the controller or -1 (default value), a **GTRY ERR PARAM** error (M64=122) occurs. This parameter must be set in level 0 on both axes of the gantry (X1 and X2).

K	Name	Comment
K312	Master gantry axis number	Gives the number of the master gantry axis

The parameter **KL313** allows the user to set the mask of the axes concerned by the gantry. This mask must contain the axes X1 and X2 of the gantry and the Y axis for the beam if the user wants to use an advanced gantry feedforward. In level 1 and 2, if KL313 does not contain the value given by the monitoring ML83, a **GTRY ERR PARAM** error (M64=122) occurs. If more than 3 axes are in the mask when setting KL313, a **BAD CMD PARAM** error (M64=70) occurs. This parameter must be set in level 0 on both axes of the gantry (X1 and X2).

K	Name	Comment
KL313	Gantry axes mask	Gives the mask of the gantry axes

The parameter **K314** allows the user to set the axis number for the beam (Y) of the gantry. This information is needed if the user wants to use an advanced gantry feedforward. This parameter must be set in level 0 on both axes of the gantry (X1 and X2).

K	Name	Comment
K314	Beam gantry axis number	Gives the axis number of the axis present on the beam of the gantry

12.4 Gantry protections and errors

The parameter C245 (refer to [§12.2.2](#)) must be different than 0 to allow the user to activate the gantry protections functionalities. These functionalities are a protection control inside the controller between the 2 axes of the gantry. Refer to [§21.](#) to have the complete list of the error messages.

12.4.1 Error management between the axes on the gantry

When an error is present on one of the gantry axis, the other axis goes immediately in **GANTRY ERROR** error (M64=118).

12.4.2 Tracking error management between the axes on the gantry

The parameter K30:1 allows the user to define a gantry tracking error different than the tracking error on a single axis. If K30:1.<axis>=0, the internal value for the gantry tracking error is defined by K30:0.<axis>. The gantry tracking error is checked only with K30:1.'first' axis of the controller. When the delta position (measured position on 48 bits) is greater than K30:1.'first' axis, the axis enters immediately in **GANTRY TRACKING** error (M64=119).

12.4.3 'Power off' protection management between the axes on the gantry

When an axis goes in 'power off', both axes enters immediately in **GANTRY PWR ERR** error (M64=120).

12.4.4 Parameter check

Some parameters must have the same value on both axes of the gantry (refer to [§17.](#) for the complete list). If a difference appears between the axes, the two gantry axes enter immediately in error **GTRY ERR PARAM** error (M64=122). The strategy regarding the regulation is that the current loop must be the same while the position loop can be different. With the monitoring **M265**, it is possible to know the number of parameters concerned by this error.

M	Name	Comment
M265	Gantry parameter error	Gives the gantry parameter error. Depth 0: parameter number, depth 1: parameter type (refer to §2.2.1.1), depth 2: parameter depth

Remark: This check is executed on the background side.

12.5 Management in gantry level 2

12.5.1 Commands management

In gantry level 2 (C245.<axis>=2), the command management is different that on level 1 or 0. As only one axis (the master) can be visible, the command must be treated in a different way.

12.5.1.1 Master executes for itself; Slave enters in error

Some commands are only executed on the master side. If the slave receives the command, the slave enters immediately in **GTRY BADCMD LVL2** error (M64=121). The commands managed that way (action 1) are listed in [§16..](#)

An exception is the command NEW. It is not allowed in gantry level 2 to make a NEW.<axis>=0, 2, 3 or 4 (in

this case, both axes enter immediately in **GTRY BADCMD LVL2** error (M64=121). Refer to [§6.](#) for more information about the NEW command.

12.5.1.2 Master executes for himself and for the slave; Slave enters in error

For some commands, if the master receives a command, this command is duplicated on the slave side and executed simultaneously on both axes. If the slave receive the command, the slave enters immediately in **GTRY BADCMD LVL2** error (M64=121). The commands managed that way (action 2) are listed in [§16..](#)

12.5.1.3 Master executes for himself; Slave executes for himself

Some commands can be executed by the master or by the slave. If the master or the slave receives the command, this command is executed. The commands managed that way (action 3) are listed in [§16..](#)

12.5.1.4 Master or slave enters in error

Some commands could not be executed by the master or by the slave. If the master or the slave receives the command, the master or the slave enters immediately in **GTRY BADCMD LVL2** error (M64=121). The commands managed that way (action 4) are listed in [§16..](#)

12.5.1.5 Master executes for himself and for the slave; Slave ignore

For some commands, if the master receives the command, this command is duplicated on the slave side and executed simultaneously on both axes. If the slave receives the command, the slave ignores this command. The commands managed that way (action 6) are listed in [§16..](#)

12.5.2 Parameters management

In gantry level 2 (C245.<axis>=2), the parameters management is different than on level 1 or 0. As only one axis (the master) can be visible, the parameters management must be treated in a different way.

12.5.2.1 Master executes for himself; Slave enters in error

Some parameters are only executed on the master side. If the slave receives the parameters, the slave enters immediately in **GTRY BADCMD LVL2** error (M64=121). The parameters managed that way (action 1) are listed in [§17..](#)

12.5.2.2 Master executes for himself and for the slave; Slave enters in error

For some parameters, if the master receives the parameter, this parameter is duplicated on the slave side and executed simultaneously on both axes. If the slave receives the parameter, the slave enters immediately in **GTRY BADCMD LVL2** error (M64=121). The parameters managed that way (action 2) are listed in [§17..](#)

12.5.2.3 Master executes for himself; Slave executes for himself

Some parameters can be executed by the master or by the slave. If the master or the slave receives the command, this command is executed. The parameters managed that way (action 3) are listed in [§17..](#)

12.5.2.4 Master or slave goes in error

For all the other parameters, they could not be executed by the master or by the slave. If the master or the slave receives the command, the master or the slave enters immediately in **GTRY BADCMD LVL2** error (M64=121).

12.5.3 Monitorings management

In gantry level 2 (C245.<axis>=2), the monitorings management is different than on level 1 or 0. As only one axis (the master) can be visible, the monitorings management must be treated in a different way.

12.5.3.1 Depth 0 shows combination between master & slave, depth 1 shows master, depth 2 shows slave

For some monitoring, the depth 0 shows a combination between the master and the slave axes, depth 1 shows the master value (X1) and depth 2 shows the slave value (X2). The monitorings managed that way (feedback 4) are listed in [\\$19](#).

Remark: For M60: axis X1 and axis X2 must be set to have the combined bit#set to 1. It is the case for 'Power ON', 'First INIT done' and 'In Windows'.

For M61:axis X1 or axis X2 must be set to have the combined bit#set. It is the case for 'Motor is moving', 'Fatal error', 'Warning' and 'Position captured'.

For M63: The combined bit#is equal to the bit#of axis X1 (master). It is the case for 'Trace busy', 'Sequence on', 'Sequence error label pending' and 'User bits' (K177.X1).

12.5.3.2 Depth 0 shows gantry, depth 1 shows master, depth 2 shows slave

For some monitoring, the depth 0 shows the gantry value (X), depth 1 shows the master value (X1) and depth 2 shows the slave value (X2). The monitorings managed that way (feedback 3) are listed in [\\$19](#).

12.5.3.3 Depth 0 shows the value of X1

For all the other monitorings, the depth 0 shows the value of X1.

12.5.4 Sequence management in gantry level 2

In gantry level 2 (C245.<axis>=2), the sequence is only available on the master side. If a sequence is present on the flash for the slave, the sequence is not restored when the gantry is in level 2.

12.6 Homing mode in gantry level

The parameter **KL49** allows the user to set the minimum and maximum delta value allowed between KL45.<master> and KL45.<slave>.

K	Name	Comment
KL49	Delta offset absolute position in gantry mode	Gives the delta offset absolute position in gantry mode. Depth 0 represents the minimum value for delta KL45 and depth 1 represents the maximum value for delta KL45.

If delta KL45 is out of the limit determined by KL49:0.<master> or KL49:1.<master>, the **GTRY ERR HM OFFST** error (M64=124) is set on both axes of the gantry.

If the user tries to change the value of parameter KL49 when the gantry is in level 2 (C245.<axis>=2), the **GTRY BAD CMD LVL2** error (M64=121) is set on both axes of the gantry.

Remark: (KL49:0.<master>) < (KL45.<master> - KL45.<slave>) < (KL49:1.<master>) If both value for KL49:0<master> and KL49:1<master> are set to 0, the functionality is disabled. As soon as one of the depth of KL49 is different than 0, the functionality is enabled.

12.6.1 Homing in gantry level 1

In gantry level 1 (C245.<axis>=1), an UltimET is needed to perform a synchronized homing with the command *IND=<axis mask>. The following homing modes are available:

- K40.<X1,X2>=8 or 9 Homing on mono-reference mark with mechanical end stop.
- K40.<X1,X2>=10 or 11 Homing on mono-reference mark with limit-switch.
- K40.<X1,X2>=12 or 13 Homing on a multi-reference mark with mechanical end stop.
- K40.<X1,X2>=38 or 39 Homing on a mono-reference mark but only after having found the mechanical end stop.
- K40.<X1,X2>=44 or 45 Homing on a mono-reference mark but only after having found the limit switch.

- K40.<X1,X2>=46 Homing without effect on the position.

In gantry level 1, at the end of the homing process, the slave axis moves to the master position. During this movement, the tracking error management between the axes of the gantry is activated. A bad value in KL45 generates an error without movement.

In gantry level 1, it is not allowed to have EnDat2.2 incremental encoders.

12.6.2 Homing in gantry level 2

In gantry level 2 (C245.<axis>=2), the homing is controlled by the master and only some mode are available:

- K40.<X1,X2>=8 or 9 Homing on mono-reference mark with mechanical end stop.
- K40.<X1,X2>=10 or 11 Homing on mono-reference mark with limit-switch.
- K40.<X1,X2>=22 Immediate homing.
- K40.<X1,X2>=46 Homing without effect on the position.

For all the other values of K40, the **HOME NOT POSSIBLE** error (M64=69) occurs if the IND command is sent to the master.

In gantry level 2, at the end of the homing process, the slave axis moves to the master position. During this movement, the tracking error management between the axes of the gantry is activated. A bad value in KL45 generates a movement and could stuck the gantry.

In gantry level 2, it is not allowed to have EnDat2.2 incremental encoders.

12.7 How to make the setting of a gantry

12.7.1 Setting of axes X1 and X2

- Perform the drive setting of the axis X1 with the axis X2 in power off and C245.<X1 or X2>=0.
- Copy the parameters of axis X1 on axis X2.
- Send the command AUT.<X2>=2 to check the motor phases. Axis X1 must be switched off.
- Send the command IND.<X2> to check if X2 moves in the same direction as X1. Axis X1 must be switched off.
- Adjust the parameter K68.<X2> to have X1 and X2 moving in the same direction. Execute again AUT.<X2>=2 with X1 switched off if K68.<X2> has been changed.
- Set the parameters K312.<X1,X2> and KL313.<X1,X2> if no advanced gantry feedforward is needed or K312.<X1,X2>, KL313.<X1,X2,Y> and K314.<X1,X2,Y> if an advanced gantry feedforward is needed.
- Switch on axis X1 and set the parameter C245.<X1 or X2>=1 to have protections between X1 and X2 for the rest of the setting.
- Adjust the parameter KL45.<X2> to have the minimum current consumption on X1 and X2. Make small movements on axis X2 and check MF31.<X1 & X2> to find the smallest current. Multiply the distance from your initial position by -1 and copy it into KL45.
- Adjust on both axes the position loop parameter if necessary.
- If wanted, set the advanced gantry feedforward (Y must be already set).
- Set C245.<X1 or X2>=2 to get only one gantry axis visible on the application side.
- Save all the parameters on X1 and X2.
- If C245.<X1 or X2>=2, restart the controllers.

12.7.2 Setting of the beam (axis Y)

- Perform the drive setting of axis Y with ComET.

12.8 Command for slot assignment (if advanced gantry feedforward needed)

The setting of the Real-Time Value (RTV) must be done before powering-up the axes of the gantry. If an advanced gantry feedforward is needed (only possible with TransnET), it is necessary to set the RTV between the beam and the Gantry (refer to [§10](#)).

The axis of the beam must send the following information to the master of the gantry:

- Position (ML0<Y> or M0<Y>)

- Status (M60.<Y>)

The commands on the beam side are:

- slot_ML0_lsl=get_32bitDataSlot()
- slot_ML0_msl=get_32bitDataSlot()
- slot_M60=get_32bitDataSlot()
- ASW.<Y>=ML0.<Y>, <slot_ML0_lsl>, <slot_ML0_msl> or ASW.<Y>=M0.<Y>, <slot_ML0_lsl>
- ASW.<Y>=M60.<Y>, <slot_M60>

The command **ASG** (Assign Slot for Gantry) is used to assign slots for getting the position and status sources for the gantry algorithm.

Command	<P>	Comment
ASG.<AXIS>=<P1>,<P2>,<P3>	<P1> <P2> <P3>	Defines the offset in the TransnET DPRAM where the controller reads the LSL part of ML0 for gantry algorithm Defines the offset in the TransnET DPRAM where the controller reads the MSL part of ML0 for gantry algorithm Defines the offset in the TransnET DPRAM where the controller reads the axis status for gantry algorithm

ASG.<X1>=slot_ML0_lsl, slot_ML0_msl, slot_M60

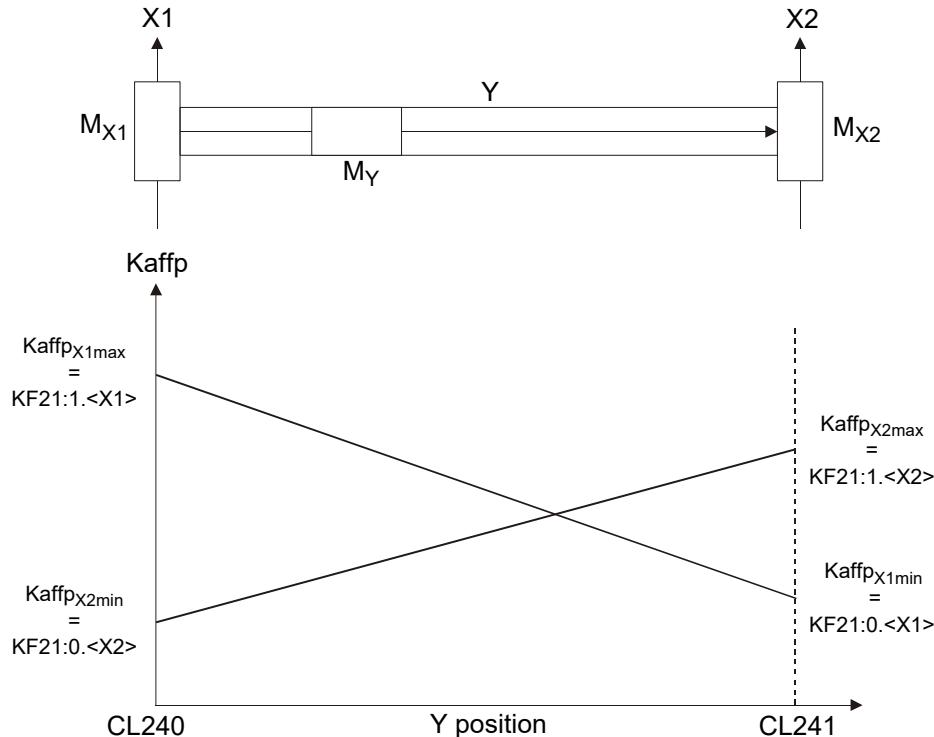
If the position send by the axis of the beam is M0, the second parameter of the command must be -1.

The command ASG returns the following error in case of bad configuration:

- If less or more than 3 parameters are used, the **BAD CMD PARAM** error (M64=70) occurs.
- If <P1>=-1, if <P2> or <P3>!= -1, if only the slot_ML0_msl is defined, If the slot number is greater than 511, If the slot_M60 is equal to -1, the **BAD SLOTTRANSNET** error (M64=58) occurs.

12.9 Advanced gantry feedforward

The feedforward given by the parameter KF21 of X1 and X2 depends on the position of the axis on the beam.



The parameter **C246** allows the user to enable the advanced gantry feedforward.

C	Name	Comment
C246	Advanced gantry feedforward	Determines the advanced gantry feedforward. 0 => no advanced gantry feedforward, 1=> advanced gantry feedforward

This advanced gantry feedforward requires the following parameters:

- A second depth for parameter KF21.
- depth 0 of axis X1 is used when the axis Y is on the CL241 position.
- depth 1 of axis X1 is used when the axis Y is on the CL240 position.
- depth 0 of axis X2 is used when the axis Y is on the CL240 position.
- depth 1 of axis X2 is used when the axis Y is on the CL241 position.

C	Name	Comment
CL240	Gantry min beam position	Gives the gantry minimum beam position (Y axis) when positioned on the master
CL241	Gantry max beam position	Gives the gantry maximum beam position (Y axis) when positioned on the slave

The monitoring **MF121** allows the user to know the current value of KF21.

M	Name	Comment
MF121	position loop acceleration feedforward gain	Gives the position loop acceleration feedforward gain value

12.9.1 Management of the status beam axis (Y) on gantry side

If the Y axis is not referenced, (bit#‘homing done’=0), the parameter KF21 used on X1 and X2 axes has the value given in depth 0.

12.9.2 Setting of the advanced gantry feedforward

- Set C245.<X1 or X2>=0.
- Save all parameters and restart the controllers.
- Set K314.<X1,X2,Y> to define the beam axis.
- Set CL240.<X1 or X2>, the maximum position of Y towards the master X1.
- Set CL241.<X1 or X2>, the maximum position of Y towards the slave X2.
- Perform typical movement on the gantry with Y on position CL240 and set KF21:1.<X1>. Set C245<X1,X2>=1
- Perform typical movement on the gantry with Y on position CL240 and set KF21:0.<X2>.
- Perform typical movement on X1 with Y on position CL241 and set KF21:0.<X1>.
- Perform typical movement on X2 with Y on position CL241 and set KF21:1.<X2>.
- Set C246.<X1 or X2>=1 to enable the algorithm.
- Set C245.<X1 or X2>=1 or 2.
- Save all parameters and restart the controllers if needed.
- Program on the UltimET the shared information between the beam and the gantry:
 - Get a slot number on TransnET to share the beam position:
 - slot_ML0_lsl=get_32bitDataSlot()
 - slot_ML0_msl=get_32bitDataSlot()
 - slot_M60=get_32bitDataSlot()
 - Send the two commands ASW to the beam axis to share the position and status:
 - ASW.<Y>=ML0.<Y>, <slot_ML0_lsl.<Y>>, <slot_ML0_msl.<Y>>
 - ASW.<Y>=M60.<Y>, <slot_M60.<Y>>
 - Send the command ASG (Assign Slot for Gantry) to the axis X1 to get the beam position and status:
 - ASG.<X1>=<slot_ML0_lsl.<Y>>, <slot_ML0_msl.<Y>>, <slot_M60.<Y>>

13. Force control mode

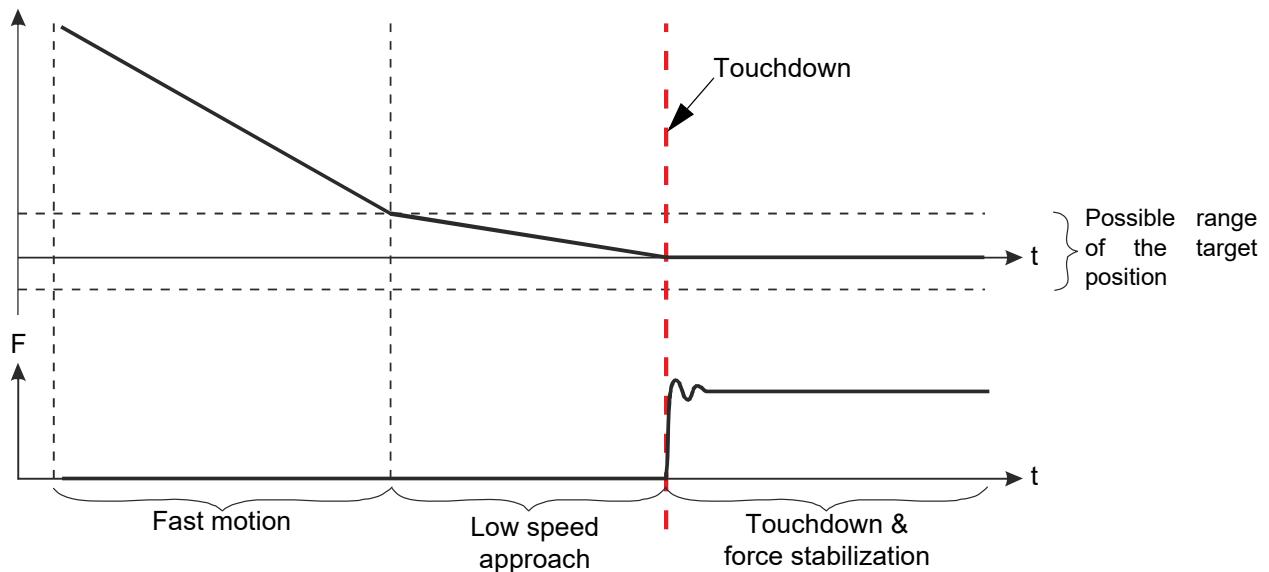
13.1 Single axis force control

13.1.1 Introduction

Nowadays, many processes need to do a motion in a position controlled way and at the end of it they need to control accurately the force applied. It means that the position controller should be able to switch from position control to force control as fast as possible to ensure a better throughput. For this purpose, ETEL has developed a force control mode. This feature allows the user to start a motion in a controlled position and to switch in a force controlled mode without stopping the motion in between.

For the force control, there are three main phases: a fast motion, a low speed approach and a touchdown as shown below.

Position



The contact force can be detected by using an external force sensor or an internal force estimator. It is possible to set a contact force window to set a bit in the status of the position controller when the force is in a defined force window during a defined amount of time. Many applications need also to be able to change the contact force reference when they are in force control mode.

13.1.2 Contact force definition

There are two main ways to define the contact force, depending of the application requirements:

- Through an external force sensor (connected to the AccurET I/O board for example): high accuracy but additional components are needed (increase space and cost).
- Through an internal estimator: cost and space effective solution but a calibration is needed.

13.1.3 Commands

The following commands are needed to set the force control function.

13.1.3.1 SETFC command

The **SETFC** (**S**ET **F**orce **C**ontrol) command is used to start the motion and to switch in force mode.

Command	<P>	Comment	Unit
SETFC.<axis>=<PF1>,<P2>,<PL3>,[<PF4>],[<P5>],[<PL6>],[<PL7>]	<PF1> <P2> <PL3> <PF4> <P5> <PL6> <PL7>	Force reference (F_{REF}) Mode for the detection and force loop (Fctrl_mode) Absolute position switch to force control for the axis (X_{ABS}) Force range of the window, optional ($\Delta F_{WINDOWS}$) Duration of the force window, optional ($T_{WINDOWS}$) Absolute speed for the axis, optional (V_{ABS}) Absolute acceleration for the axis, optional (A_{ABS})	[fref_fctrl] - [upi] [fref_fctrl] [plti] [usi] [uai]

The different values for the Fctrl_mode are as follows:

<P>	Value	Comment
<P2>=Fctrl_mode	0 1	Standard mode allowing a good control of overshoot at landing. In this mode, the feedforward remains active (KF21, KF20, KF22, KF247, KF248 and K119). Improves the force stability after touchdown when the encoder resolution is poor. The feedforward (KF21, KF20, KF22, KF247 and KF248) are active during the touchdown and inactive after touchdown.

For $\Delta F_{WINDOWS}$ and $T_{WINDOWS}$, the command affects the internal variable but has no impact on KF191 and K190. The monitoring MF395.<axis> ($\Delta F_{WINDOWS}$) and M395.<axis> ($T_{WINDOWS}$) allow the user to see the force windows configuration.

The SETFC command is executed if:

- Bit#0 of K302.<axis>=1 Force control mode selected
- K61.<axis>=1 Standard mode with set point generator movement profiles
- K63.<axis>=0 Force/torque reference is coming from position regulator
- M110.<axis>=0 Not in interpolation mode (ITP)

The SETFC command waits before being executed that there is no trajectory active on the axis (bit#20 of M63.<axis>=0).

This command could generate the followings errors:

- **MVT NOT POSSIBLE** (M64=67) if the axis is not powered when the command is send
- **BAD CMD PARAM** (M64=70) if the command is with less than two parameters
- **BAD CMD PARAM** (M64=70) if a parameter of the command is with a wrong format
- **HOMING NOT DONE** (M64=71) if the axis is not referenced when the command is send
- **FORCE CTRL CFG** (M64=90) if the command is sent with bit#0 of K302.<axis>=0
- **MVT NOT ALLOWED** (M64=92) if the axis is in interpolation mode when the command is send
- **MVT NOT ALLOWED** (M64=93) if the axis is with K61≠1 or K63≠0 when the command is send
- **GTRY BADCMD LVL2** (M64=121) if the command is sent in gantry mode 2 (C245.<axis>=2)

Remark: Now if a MVE, RMVE, STE_ABS, STE_ADD, STE_SUB, STA STI, IND, or SLS command is sent after the SETFC command but before a RESETFC command, a **MVT NOT ALLOWED** error (M64=91) occurs.

13.1.3.2 SETFCI command

The **SETFCI** (SET Force Control Immediate) command is used to switch in force mode without movement.

Command	<P>	Comment	Unit
SETFCI.<axis>=<PF1>,<P2>,<P3>,<P4>,[<PF5>],[<P6>]	<PF1> <P2> <P3> <P4> <PF5> <P6>	Force reference (F_{REF}) Mode for the detection and force loop (Fctrl_mode) Gives the direction (sign) for the force reference (Dir) Smooth filter for applying the force (smoothing) Force range of the window, optional ($\Delta F_{WINDOWS}$) Duration of the force window, optional ($T_{WINDOWS}$)	[fref_fctrl] - - [plti] [fref_fctrl] [plti]

The different values for the Fctrl_mode (<P2>) are as follows:

<P>	Value	Comment
<P2>	0	Standard mode allowing a good control of overshoot at landing. In this mode, the feedforward remains active (KF21, KF20, KF22, KF247, KF248 and K119).
	1	Improves the force stability after touchdown when the encoder resolution is poor. The feedforward (KF21, KF20, KF22, KF247 and KF248) are active during the touchdown and inactive after touchdown.

Definition for Dir:

- If Dir is equal to 0 or greater, a positive force will be applied.
- If Dir is with a negative value, a negative force will be applied.

Definition for Smoothing:

This parameter allows the user to smooth the applied force in order to not apply a step.

For the $\Delta F_{\text{WINDOWS}}$ and T_{WINDOWS} , the command affects the internal variable but has no impact on KF191 and K190. The monitoring MF395.<axis> ($\Delta F_{\text{WINDOWS}}$) [fref_fctrl] and M395.<axis> (T_{WINDOWS}) [plti] allow the user to see the force windows configuration.

The command SETFCI is executed if:

- | | |
|--------------------------|---|
| • Bit#0 of K302.<axis>=1 | Force control mode selected. |
| • K61.<axis>=1 | Standard mode with set point generator movement profiles. |
| • K63.<axis>=0 | Force/torque reference is coming from position regulator. |
| • M110.<axis>=0 | Not in interpolation mode (ITP). |

The command SETFCI waits before being executed that there is no trajectory active on the axis (bit#20 of M63.<axis>=0).

The command SETFCI could generate the followings errors:

- **MVT NOT ALLOWED** (M64=67) if the axis is not powered when the command is send
- **BAD CMD PARAM** (M64=70) if the command is with less than four parameters
- **BAD CMD PARAM** (M64=70) if a parameter of the command is with a wrong format
- **HOMING NOT DONE** (M64=71) if the axis is not referenced when the command is send
- **FORCE CTRL CFG** (M64=90) if the command is sent with bit#0 of K302.<axis> = 0
- **MVT NOT ALLOWED** (M64=92) if the axis is in interpolation mode when the command is send
- **MVT NOT ALLOWED** (M64=93) if the axis is with K61≠1 or K63≠0 when the command is send
- **GTRY BADCMD LVL2** (M64=121) if the command is sent in gantry mode 2 (C245.<axis>=2)

13.1.3.3 NEWFC command

The **NEWFC** (NEW Force Control) command allows changing the force reference value currently applied. The force control mode defined in the SETFC command remains unchanged (refer to [§13.1.3.1](#)).

Command	<P>	Comment	Unit
NEWFC.<axis>=<PF1>, <P2>, [<PF3>], [<P4>]	<PF1> <P2> <PF3> <P4>	Force reference (F_{REF}) Time: Duration of the transition between previous and new force reference Force range of the window, optional ($\Delta F_{\text{WINDOWS}}$) Duration of the Force window, optional (T_{WINDOWS})	[fref_fctrl] [plti] [fref_fctrl] [plti]

For $\Delta F_{\text{WINDOWS}}$ and T_{WINDOWS} , the command affect the internal variable but have no impact on KF191 and K190. The monitoring MF395.<axis> ($\Delta F_{\text{WINDOWS}}$) and M395.<axis> (T_{WINDOWS}) allow the user to see the force windows configuration.

If Time=0, the new force reference is applied immediately, otherwise the change from the previous F_{REF} and the new one is linear. The axis will stay in the force control mode. The in-window bit will be set to 0 and the in-window check will start again. If the axis is not in force control mode, the command is acknowledged without any error.

The NEWFC command is executed if:

- Bit#0 of K302.<axis>=1 Force control mode selected
- M392.<axis>=1 The axis is in force control mode

The NEWFC command waits before being executed that there is no trajectory active on the axis (bit#20 M63.<axis>=0).

This command could generate the followings errors on both axes of the controller:

- **MVT NOT POSSIBLE** (M64=67) if the axis is not powered when the command is executed
- **BAD CMD PARAM** (M64=70) if the command is with less than two parameters
- **BAD CMD PARAM** (M64=70) if a parameter of the command is with a wrong format
- **HOMING NOT DONE** (M64=71) if the axis is not referenced when the command is executed
- **FORCE CTRL CFG** (M64=90) if the command is sent with bit#0 of K302.<axis>=0
- **MVT NOT ALLOWED** (M64=92) if the axis is in interpolation mode when the command is send
- **MVT NOT ALLOWED** (M64=93) if the axis is with K61≠1 or K63≠0 when the command is send
- **GTRY BADCMD LVL2** (M64=121) if the command is sent in gantry mode 2 (C245.<axis>=2)

13.1.3.4 RESETFC command

The **RESETFC** (RESET Force Control) command is used to switch off the force mode and to move the axis, or to switch off the force mode only and stay in position.

Command	<P>	Comment	Unit
RESETFC.<axis>=<P1>, [<PL2>], [<PL3>], [<PL4>]	<P1>	Option: 0 = immediate (other value reserved for the future) Option: 1 = collapse detection mode with absolute movement with [<PL2>] Option: 2 = collapse detection mode with relative movement with [<PL2>]	- - -
	<PL2>	Absolute position for the axis, optional (X _{ABS})	[upi]
	<PL3>	Absolute speed for the axis, optional (V _{ABS})	[usi]
	<PL4>	Absolute acceleration for the axis, optional (A _{ABS})	[uai]

- With 'Option'=0 and no optional parameter, the force mode is immediately disabled and the axis stays in position.
With 'Option'=0 and X_{ABS} ([<PL2>]) defined, the force mode is immediately disabled and the axis moves upward to the position X_{ABS}, with the default speed and acceleration or with V_{ABS} ([<PL3>]) and A_{ABS} ([<PL4>]) if they are defined.
- With 'Option'=1 and no optional parameter, the force mode is disabled only after the detection of the collapse, i.e. when the real position ML1 is crossing the collapse threshold and the axis stays at the position where the threshold was detected (M389).
With 'Option'=1 and X_{ABS} ([<PL2>]) defined, the force mode is disabled only after the detection of the collapse, i.e. when the real position ML1 is crossing the collapse threshold and the axis moves upward to the position X_{ABS}, with the default speed and acceleration or with V_{ABS} ([<PL3>]) and A_{ABS} ([<PL4>]) if they are defined.
- With 'Option'=2 and no optional parameter, the force mode is disabled only after the detection of the collapse, i.e. when the real position ML1 is crossing the collapse threshold and the axis stays at the position where the threshold was detected (M389).
With 'Option'=2 and X_{ABS} ([<PL2>]) defined, the force mode is disabled only after the detection of the collapse, i.e. when the real position ML1 is crossing the collapse threshold and the axis moves to the position where the threshold was detected (M389) + X_{ABS} (in that case this is a relative position), with the default speed and acceleration or with V_{ABS} ([<PL3>]) and A_{ABS} ([<PL4>]) if they are defined.

The command RESETFC is executed if:

- Bit#0 of K302.<axis>=1 Force control mode selected
- K61.<axis>=1 Standard mode with set point generator movement profiles
- K63.<axis>=0 Force/torque reference is coming from position regulator
- M110.<axis>=0 Not in interpolation mode (ITP).

The command RESETFC waits before being executed that there is no trajectory active on the axis (bit#20, M63.<axis>=0).

This command could generate the following errors:

- **MVT NOT POSSIBLE** (M64=67) if the axis is not powered when the command is send
- **BAD CMD PARAM** (M64=70) if the command is with less than two parameters
- **BAD CMD PARAM** (M64=70) if a parameter of the command is with a wrong format

- **HOMING NOT DONE** (M64=71) if the axis is not referenced when the command is send
- **FORCE CTRL CFG** (M64=90) if the command is sent with bit#0 of K302.<axis>=0
- **GTRY BADCMD LVL2** (M64=121) if the command is sent in gantry mode 2 (C245.<axis>=2).

Remark: PWR.<axis>=0, HLT.<axis>, HLO.<axis>, HLB.<axis> and RST.<axis> commands switch off the force mode.

• Collapse detection

The parameter **K308** allows the user to define the distance between the real position ML1 and the collapse detection threshold.

K	Name	Comment	Unit
K308	collapse detection threshold	Gives the distance of the collapse detection threshold	[dpi]

As soon as the RESETFC command with the option=1 or 2 is received in the controller, the collapse detection limit is computed as follow: Limit = ML1 + K308.

K308 can have a positive or a negative value, depending on the reading way of the encoder:

- If the positive way is upward, the limit is smaller than the real position ML1 and K308 is negative.
- If the positive way is downward; the limit is bigger than the real position ML1 and K308 is positive.

The collapse detection limit is not fixed forever. It moves with the thermal expansion of the bounded parts. To follow the movement of the threshold, the monitoring **M389** is available.

M	Name	Comment	Unit
M389	threshold for collapse detection	Gives the moving threshold for collapse detection	[dpi]

Remark: In one direction of the real position ML1, the collapse and the thermal expansion in the opposite direction are checked. The collapse detection during a thermal contraction is not possible. To work properly, the noise amplitude of the position feedback must be clearly smaller than the collapse detection limit (at least 3 times), otherwise undesired triggers could occur. If the collapse is never detected, the command stays pending until an urgent command !HLO, !HLB or !HLT stops the process. The user has to manage a time-out error in his EDI application to handle such event.

The user can be informed about the collapse as follow:

- The command RESETFC is acknowledged only when the collapse is detected.
- The bit#14 of M390 (Force mode status) is set when the collapse is detected. As soon as the axis leaves the force control mode, this bit is set to 0 (all bits from M390 are set to 0 when the axis leaves the force control mode). It is not possible to check this bit because it is only present less than 1 MLTI.

For the end of the motion, the 'is_moving' bit present in M63 is used.

Remark: The threshold detection is made on 32 bits. So the distance between the real position and the threshold is limited to 2^{31} increments.

13.1.3.5 WTF command

The **WTF** (WaIT Force window) command allows the user to define a 'force window' around the target force in force control mode (MF392).

Command	Comment
WTF.<axis>	Wait in force control that the current error is around the current reference MF401

This command could generate the following errors:

- **FORCE CTRL CFG** (M64=90) if the command is sent with K302.<axis>=0

The monitoring MF395.<axis> ($\Delta F_{\text{WINDOWS}}$) and M395.<axis> (T_{WINDOWS}) allow the user to see the force windows configuration.

The time windows could be defined by the followings:

- Parameter K190 (duration of the window in force control mode) allows the user to define the duration of the window.
- <P5> of SETFC command
- <P4> of NEWFC command

The force range of the window could be defined by the followings:

- Parameter KF191 (force range of the window in force control mode) allows the user to define force/torque range of the window.
- <PF4> of SETFC command
- <PF3> of NEWFC command

Remark: Bit#17 of M390 as well as bit#17 of M63 are defined to give the 'in_force_window' information.

If parameter M395 and/or MF395.<axis>=0, the command WTF is immediately acknowledged and the 'in_force_windows' bit of M390 and M63 are set to 0.

It is possible to define a 'force window' around the force reference (MF392) when the controller is in force regulator mode (bit#0 monitoring M390 set to 1).

The motor is considered in the 'force window' when the estimated current (MF400) is between [current reference (MF401) - MF395 (converted in [cur])] and [current reference (MF401) + MF395 (converted in [cur])] during a minimum time given by the parameter M395 without interruption. If it is the case, the bit#17 of M390 as well as the bit#17 of M63 are set to 1.

In this mode, if the estimated current (MF400) is not between [current reference (MF401) - MF395 (converted in [cur])] and [current reference (MF401) + MF395 (converted in [cur])], the bit#17 of M395 as well as the bit#17 of M63 are set to 0.

As soon the axis is in force regulation, the windows check is active around the target defined by the current reference (MF401).

13.1.4 Registers

The following registers are needed to set the force control function.

The parameter **K302** allows the user to configure the force control feature on AccurET. The bit#0 must be equal to 1 when the controller boots.

K	Name	Comment	Unit
K302	Force control mode activation & configuration	Bit#0: the force control mode is selected Bit#2: approach with constant speed Bit#3: touchdown detection with speed	-

If this register is set to 0, the SETFC, RESETFC, NEWFC and WTF commands will generate the **FORCE CTRL CFG** error (M64=90).

- **Contact force definition**

Registers	Value	Comment	Unit
MF31	-	Real current Iq measured	[cur]
M109	-	Theoretical speed before advanced filter depth 0	[dsi]
M111	-	Real speed before advanced filter depth 0	[dsi]
M114	-	Real acceleration before advanced filter depth 5	[dai]
M115	-	Real acceleration after advanced filter depth 5	[dai]
MF250	-	Cogging compensation value	[cur]
MF398	-	Estimated current after filtering and smoothing	[cur]

Registers	Value	Comment	Unit
MF400	-	Estimated current after filtering before smoothing	[cur]
MF406	-	Force feedback before smoothing and filtering	[cur]
KF104-KF108 (depth 5)	-	Advanced filter 6 for real acceleration	-
KF104-KF108 (depth 6)	-	Force estimator output filter	-
KF301	-	Acceleration gain for force estimator	-
KF302	-	Current offset for force estimator	-
KF303	-	Speed gain for force estimator	-
KF304	-	Dry friction gain for force estimator	-
KF309	-	Force estimator sensor offset	-
KF310	-	Force estimator sensor gain	-
K311	-	Force estimator sensor definition, depth 0 input type, depth 1 input index, depth 2 input depth, depth 3 axis number	-
K311:0	0 1 2 3 13 33 34 35 45 65 66 67 77	Force estimator Source type is a users variable X Source type is a parameter K Source type is a monitoring M Source type is a common parameter C Source type is a users variable XF Source type is a parameter KF Source type is a monitoring MF Source type is a common parameter CF Source type is a users variable XL Source type is a parameter KL Source type is a monitoring ML Source type is a common parameter CL	-
K311:1	0-511	Register number	-
K311:2	0-7	Register depth	-
K311:3	0 1	Current axis Other axis of the controller	-
KF316	-	Measured current gain for force estimator	-
KF317	-	Cogging compensation gain for force estimator	-
K373	-	Position spring offset for the force estimator	[m]
KF373	-	Force spring factor	[A/m]
K374	-	Maximum compression of the spring position (saturation) for the force estimator (between 0 and K374)	[m]
KF374	-	Damping spring factor	[A/(m/s)]

- Regulators

Registers	Comment	Unit
M391	Position reference added in force regulation mode	[dpi]
MF391	Current reference added in force regulation mode	[cur]
MF392	force reference for force regulator	[fref_ctrl]
MF399	Estimated force for force estimator	[fref_ctrl]
MF401	Reference current for force regulator	[cur]
MF402	Error current for force regulator	[cur]
KF6:1	Force loop integrator limitation	[cur]
K19:4	Speed threshold for touchdown detection with the speed	[dsi]
K191	Delay between the touchdown detection and the change of the force control gain	[PLTI]
K192	Smoothed estimated force for force control	[plti]
KF299	Force threshold for touchdown detection with the force feedback	[fref_ctrl]
KF300	Kt motor for force regulator	[kt_motor]
KF305	Kp gain for force regulator in mode 0	[kpf_ctrl]
KF306	Ki gain for force regulator in mode 0	[kif_ctrl]
K307	Integrator limit for force regulator	[dpi]
KF308	Lever arm for force regulator	-
KF312:1	Kp gain for force regulator in mode 1	-
KF313:1	Ki gain for force regulator in mode 1	[plti_inv]
KF314:1	Kd gain for force regulator in mode 1	[plti]
KL314	Single axis force control: upper limit after touchdown	[dpi]
KF318	Kd gain for force regulator in mode 0	[kdf_ctrl]
KF319	Force feedforward gain for force regulator in mode 1	[float]

- **Wait window force control**

Registers	Comment	Unit
M395	Duration of the window in Force Control mode	[plti]
MF395	Force range of the window in force control mode	[fref_fctrl]
K190	Duration of the window in Force Control mode	[plti]
KF191	Force range of the window in force control mode	[fref_fctrl]

- **Force Control status**

The status for the force control mode is given by the monitoring M390 which is a bit field:

M	Name	Comment
M390	Status for Force Control mode	Gives the position reference added in Force regulation mode

- bit#0: The force control mode is active.
- bit#1: Check the position for the switch in force control mode.
- bit#3: Force control on mode 0 is selected (mode 0 and 1 command SETFC).
- bit#4: Force control on mode 1 is selected (mode 1 command SETFC).
- bit#6: Force control on mode 1 is active.
- bit#12: Integrator limit reached

From release 3.03A, the bit#12 of M390 was set to 1 in Force Control mode 0 when M391 reached the limit defined by K307 or in Force Control mode 1 when MF391 reached the limit defined by KF6:1. The bit#12 remained at 1 until a RST, RESETFC, HLT, HLO or HLB command was executed by the controller.

Now, the bit#12 of M390 is set to 1 when it reaches the limit and cleared when it is below the limit. The behavior regarding the command RST, RESETFC, HLT, HLO or HLB stays the same, they clear the bit#12.
- bit#14: Collapse detected
- bit#17: In windows Force control.
- bit#18: Check rebound

In addition, bit#17 of M63 shows the force in-window bit.

- **Force regulator mode**

The mode selector for the force regulator is given by the monitoring M392 which is a bit field:

- bit#0 The force control regulator is active.

13.1.5 Touchdown detection

In force control mode, there are two possibilities to detect the touchdown: either with the force feedback or with the speed.

13.1.5.1 Detection with the force feedback

In this mode, the touchdown is detected when the force feedback MF397 (force value given by the estimator or by the external sensor) exceeds the threshold value KF299. This mode is selected by default.

13.1.5.2 Detection with the speed

This mode is enabled when the bit#3 of register K302 is set to 1. Then the touchdown is detected as soon as the real speed M11 goes under the threshold value defined by K19:4.

Depending on the force feedback setting, this solution may be faster than the detection based on the force feedback. Thanks to this shorter time, it is thus possible to improve the throughput of demanding applications.

13.1.6 Restrictions

There are some restrictions:

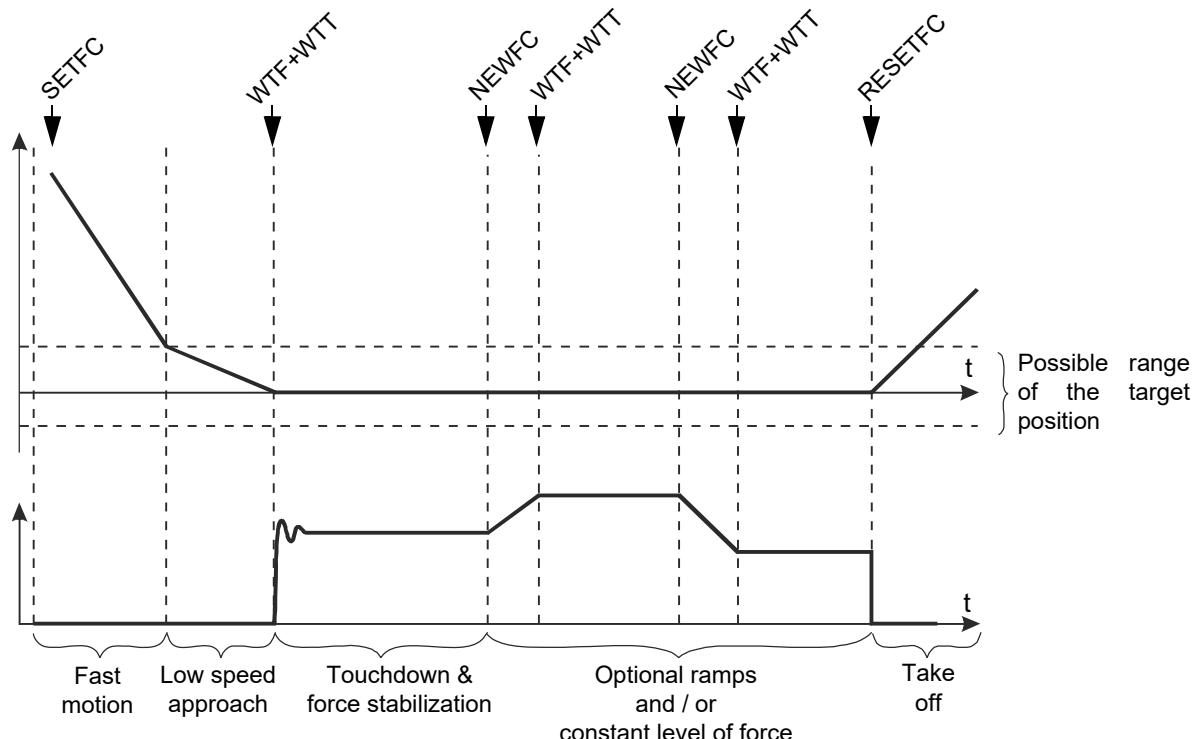
- The force control mode is available in position reference (trajectory generated by the position controller).
- As this feature is time consuming (1 μ sec for the estimator per axis and 1 μ sec for the regulation mode when active per axis), it is important to limit the options when this force control mode is used (Scale Mapping, Stage Mapping, Encoder Stretch, Gain Scheduling, motion delay, etc...).

13.1.7 Hardware protection

The parameter KL314 defines the allowed stroke for the axis in Force Control mode, from the touchdown position to the upper limit. If the measured position becomes greater than this limit during the force control, the **SETTING FCTRL** error (M64=89) is raised. This protection limit is active from the touchdown until the RESETFC command. If KL314=0, the protection limit is not active. KL314 is a fixed value and does not move with the thermal expansion like for the collapse detection. This protection is active both for the mode 1 and mode 0. The monitoring register ML396 allows the visualization of the position of this upper threshold.

M	Name	Comment	Unit
ML396	Theoretical position Force mode	Gives the theoretical position in Force mode	upi

13.1.8 Examples



The sequence for this movement is:

```

// <PF1>      Force reference value      0.4 N
// <P2>        Fctrl mode                0
// [PL3]        Absolute Position         05 m
// [PF4]        Force range of the window  0.02 N
// [P5]         Duration of the Force window 0.015 sec
// [PL6]        Absolute Speed            0.01 m/sec
// [PL7]        Absolute Acceleration     0.2 m/sec2
SETFC.<axis>=0.4,0.05,0.02,0.015,0.01,0.2;
WTF.<axis>;
WTT.<axis>=0.2;
// <FPAR1>    Force reference value      0.6 N

```

```
// <PAR2> Duration of the transition 0.15 sec
// <FPAR3> Force range of the window 0.02 N
// <PAR4> Duration of the Force window 12 msec
NEWFC.<axis>=0.6,0.15,0.02,0.012;
WTF.<axis>;
WTT.<axis>=0.2;
// <FPAR1> Force reference value 0.25 N
// <PAR2> Duration of the transition 0.15 sec
// <FPAR3> Force range of the window 0.02 N
// <PAR4> Duration of the Force window 12 msec
NEWFC.<axis>=0.25,0.15,0.02,0.012;
WTF.<axis>;
WTT.<axis>=0.2;
// <PAR1> Option how to reset the mode 0
// <LPAR2> Absolute Position 00 m
// <LPAR3> Absolute Speed 0.01 m/sec
// <LPAR4> Absolute Acceleration 0.2 m/sec2
RESETFC.<axis>=0,0.0,0.01,0.2;
WTM.<axis>;
```

14. ZxT combined module functions

The ZxT is a chuck handling module able to make a silicon wafer move vertically with static/dynamic level-off correction and rotate around the Z axis. The platform is a Z, Rx, Ry and Theta motion system. Two AccurET controllers are required to drive it.

- The first AccurET (AccurET VHP48) runs the ZxT firmware that controls Z, Rx and Ry. It is registered as the product number 42 (given by the monitoring M70) and introduced from firmware version 3.10A.
- The second AccurET is needed to drive the Theta and the coarse Z axis (if available). This second controller can be any standard AccurET controller.

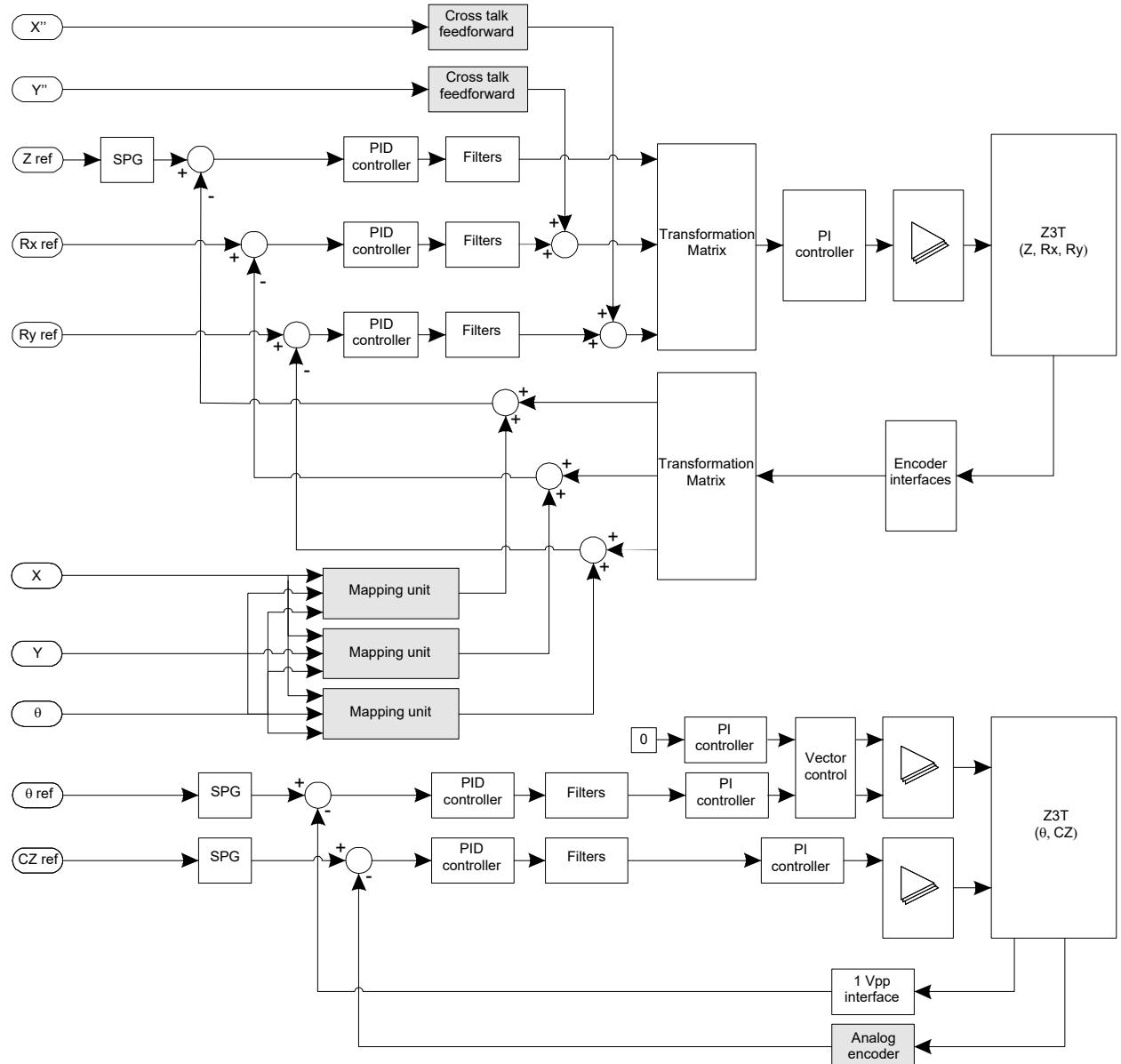
Before using it, the ZxT firmware has to be installed on the first VHP48 connected to the ZxT module. This operation will modify the product number (given by M70) into 42. The operation is reversible and a VHP48 firmware can be restored at any time using EDI or ComET.

Remark: All the registers of the AccurET VHP 48 apply to the ZxT firmware except for those specifically mentioned in this chapter.

The interpolated mode, triggers, the Scale Mapping, the Encoder Stretch, the position capture and the force control have been disabled on the Z axis of the ZxT product.

14.1 Operation principle

In the ZxT firmware, transformations matrices are used to compute Z, Rx and Ry from the linear encoders and to generate the current references to each linear EMF actuators within the module. Here is the overview of the control diagram for the combined module.



Remark: The Theta and coarse Z axes are completely independent and controlled by another AccurET.

14.1.1 Z axis

Except the presence of transformation matrices part which includes the current controllers and the position acquisition (read from the encoders), the control diagram of the Z axis is the same as any other standard AccurET products. It means all the commands, the registers and the monitorings related to the movement work as usual. Therefore, no other technical details will be given for this axis. Please refer to the corresponding AccurET manual (refer to [page 14](#)) for more information.

14.1.2 Tip-tilt (Rx, Ry)

The tip-tilt axes are used for static level-off adjustment. There is no need for complex motion planners (set point generator), only ramps filters were implemented. Originally, the AccurET firmware is designed for driving 2 axes. The Z axis is the first of these two and the tip-tilt is implemented as a dual second axis. In other words, EDI and ComET will not see 3 independent axes, but 2 where the first ZxT axis 0 refers to the Z axis and the second refers to any of Rx and Ry. The monitorings such as M2, M7 or MF233 have two depths: depth 0 refers to Rx and depth 1 is related to Ry.

14.1.3 Transfer matrices

As explained above, the main difference compared to a standard axis is the presence of transformation matrices between:

- The physical encoders and the logical axes
(E1, E2, E3, [E4]) → (Z, Rx, Ry)
- The logical forces/torques and the physical actuators current set points
(Fz, Tx, Ty) → (M1, M2, M3, [M4])

The choice of the transfer matrices is made with the common parameter **C490**. The user has to choose the value depending on the ZxT type and the mounting angle with the X and Y axes of the stage. Any change to this parameter requires a save and a reboot.

C	Name	Value	Comment
C490	ZxT transfer matrix selection	10	Z3TM at 0°
		11	Z3TM at 30°
		12	Z3TM at 90°
		13	Z3TM at 120°
		14	Z3TM at 180°
		15	Z3TM at 210°
		16	Z3TM at 270°
		17	Z3TM at 300°
		20	Z3TH at 0°
		21	Z3TH at 90°
		22	Z3TH at 180°
		23	Z3TH at 270°

14.2 Tip-tilt

The tip-tilt axes are not fully compatible with a standard axis. It uses a dedicated position controller set by the parameter K78 (K78.1=42). The main differences are:

- Standard monitorings such as M0, M1, M2, M6 or M7 have two depths: depth 0 to address Rx and depth 1 to address Ry.
- KF60 limits the logical axes.
- No trajectory generator but ramp filters.
- In current reference mode (K63.1=1), the set point is configured using the parameter KF492.
- The torque offset correction is set by the parameter KF493.
- No standard feedforward but a stage feedforward

14.2.1 Trajectory generator

A simple non-linear ramp filter is implemented on each axis. The RMVETILT and MVETILT commands (refer to [§14.4](#)) are used to move.

14.2.2 Stage feedforward

The stage feedforward allows the user to compensate the cross talk from the motions in X-Y axes to the tip-tilt axes. It is necessary to have a feedforward from the accelerations in X-Y axes to the torques in the tip-tilt axes. It can be basically seen as the situation of an inverted pendulum onto a cart. For this reason, the required feedforward terms may be represented in the following way:

$$\begin{bmatrix} \text{MF497:0.1} \\ \text{MF497:0.1} \end{bmatrix} = \begin{bmatrix} +\text{CF490:0.1} \cdot (\text{KL250:0.0} + \text{M0.0}) \cdot \text{MF455} \\ -\text{CF490:0.1} \cdot (\text{KL250:0.0} + \text{M0.0}) \cdot \text{MF457} \end{bmatrix}$$

As the monitoring MF497 is given in milliamperes, the value of the common parameter CF490 describes the moving mass m_m :

$$\text{CF490} = \frac{m_m}{K_T}$$

Remark: For stacked X-Y axes, the lever arm KL250 will differ for both Rx and Ry.
Note that the dependency in Z is due to the lever arms between the center of gravity of the components tilting and where the forces in X and Y are applied is varied when the Z position is varied.

14.2.3 Stage Mapping

To enable the Stage Mapping on Rx and Ry, please refer to the 'Stage Mapping' tool available on ComET and use it as follow:

- Select Rx with K495=0
- Configure your setting in both 'Configuration' and 'Corrections' tabs.
- Fill the mapping correction table for Rx
- Press 'Download to controllers'
- Save the mapping data with SAV=0 if required
- Press 'Activate mapping'
- Change K495=1 to select Ry
- Fill the mapping correction table for Ry
- Press 'Download to controllers'
- Save the mapping data with SAV=0 if required
- Press 'Activate mapping'

14.2.4 Software limits

- **Definition**

The behavior of MINSL (KL34) and MAXSL (KL35) differ from the one expected in a standard AccurET firmware. MAXSL defines the maximum norm of the tip-tilt vector in a (Rx, Ry) plane. It can also be seen as the limit circle centered at the home point. For compatibility reasons, MAXSL is expressed in [upi], but has to be seen as a [dpi] value while adding an offset to it makes no sense. Thus, ML4 is not taken in account here.

As mentioned above, the set point given by MVETILT or RMVETILT can be seen as a vector. When bit#0 of K36 is set, the target vector is truncated to fit inside the limit circle previously defined with MAXSL. MINSL is the minimal acceptable norm for the (Rx, Ry) vector. Currently this value is not used and has to be set to 0.

The figure below shows the two strategies used to truncate a target overlapping the MAXSL radius. If only one component is changed (first example), the second component is left unchanged and the modified component is reduced to fit inside the limit circle. In contrast, when both components Rx and Ry are changed (second example), the direction of the vector is preserved and its norm is reduced to fit inside the limit circle.

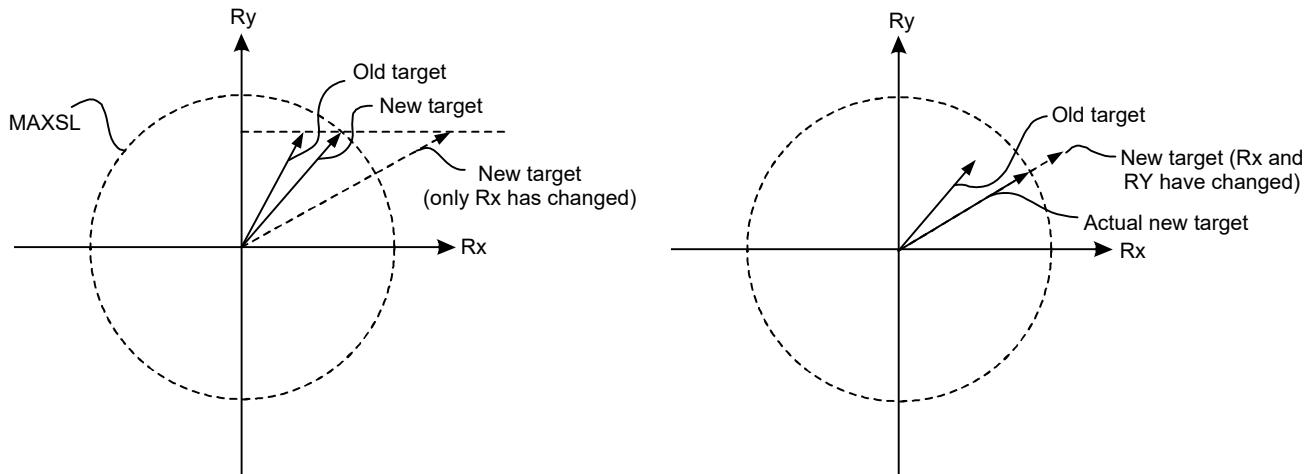
The command SLS sets the MINHL and MAXHL values as defined in the figure below. Notice that these values cannot be simply assigned to MINSL and MAXSL as this would have been the case in a standard firmware. The mechanical end stops are detected using two criteria which reach a threshold on the theoretical current

set point and a threshold on the position tracking error. The maximum theoretical current set point threshold is set with the parameter KF44 and its value represents logical amperes.

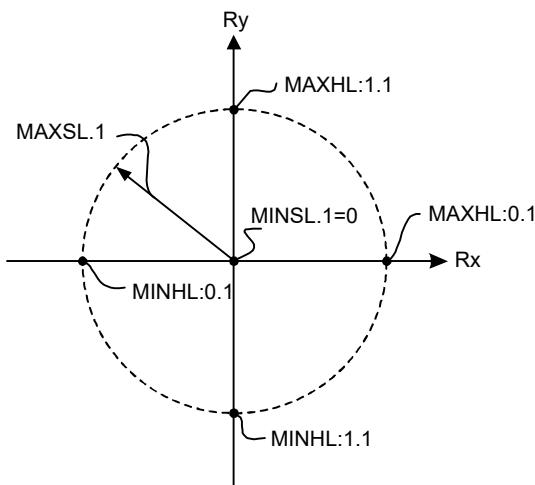
- **Search and configure the software limits**

As an example, the following procedure to set the software limits could be used.

- Software limit MAXSL in the case of tip-tilt



- Definition of MINHL / MAXHL and MINSL / MAXSL



```
# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: rad, rad/s, rad/s2, V, A, s
#define _Z_ 4
#define _TILT_ 5
#define _RX_ 0
#define _RY_ 1

void func0(void)
{
    long z_dead_pos = iconv(0.15e-3, _dpi, _Z_);
    long tilt_dead_pos = iconv(100e-6, _dpi, _TILT_);
    // Assuming homing was made on both axes.
    PWR._Z_ = 1;
    MVE._Z_ = 0;
    WTM._Z_;
    MVETILT._TILT_ = 0, 0;
    WTM._TILT_;
    SLS._TILT_;
```

```

WTM._TILT_;
SLS._Z_;
WTM._Z_;
MINSL.4 = MINHL.4 + z_dead_pos;
MAXSL.4 = MAXHL.4 - z_dead_pos;
MINSL.5 = 0; // Not relevant
if (-MINHL:_RX_._TILT_ < -MINHL:_RY_._TILT_)
    X0 = -MINHL:_RX_._TILT_;
else
    X0 = -MINHL:_RY_._TILT_;
if (MAXHL:_RX_._TILT_ < MAXHL:_RY_._TILT_)
    X1 = MAXHL:_RX_._TILT_;
else
    X1 = MAXHL:_RY_._TILT_;
if (X0 < X1)
    MAXSL.5 = X0 - tilt_dead_pos;
else
    MAXSL.5 = X1 - tilt_dead_pos;
SAV._Z_, _TILT_) = 2;
RSD._Z_ = 255;
}

```

14.3 Homing

The authorized homing modes on a ZxT modules are given by the table below.

K	Name	Value	Axis	Comment
K40	Homing mode	22	Z	Immediate homing. Measured position becomes 0
		40	Z	Search for indexes. Starts in positive direction
		41	Z	Search for indexes. Starts in negative direction
		42	Z	Search for mechanical stops. Starts in positive direction
		43	Z	Search for mechanical stops. Starts in negative direction
		22	Tip-tilt	Immediate homing. Measured position becomes 0 afterward
		48	Tip-tilt	Immediate homing based on index positions

Remark: The mode 48 requires to first perform a homing in Z with modes 40 or 41.

14.3.1 Calibration procedure

The parameter K480 can be set to define the zeros of the Z axis and the tip-tilt axes to any location relative to the index marks. In the following example, K480 sets the 0 to the location where both axes are in the middle of their respective strokes relative to the mechanical end stops.

```

# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: rad, rad/s, rad/s2, V, A, s
#define _Z_ 4
#define _TILT_ 5

/**
 * Reset factory offset.
 */
void func0(void)
{
    K480:0._Z_ = 0;
    K480:1._Z_ = 0;
    K480:2._Z_ = 0;
    K480:3._Z_ = 0;
    SAV._Z_ = 2;
    RSD._Z_ = 255;
}

```

```

}

/***
* Perform initial homing and get factory offset based on mechanical stops.
*/
void func1(void)
{
    RST.(_Z_,_TILT_);
    PWR.(_Z_,_TILT_) = 1;
    K40._Z_ = 41;
    K40._TILT_ = 48;
    SLS._TILT_;
    WTM._TILT_;
    IND._Z_;
    WTM._Z_;
    SLS._Z_;
    WTM._Z_;
    XL0._Z_ = (ML37._Z_ + ML36._Z_)/2;
    MVE._Z_ = XL0._Z_; // Move to the center of the stroke
    WTM._Z_;
    SLS._TILT_;
    WTM._TILT_;
    K480:0._Z_ = M481:0._Z_ - M480:0._Z_;
    K480:1._Z_ = M481:1._Z_ - M480:1._Z_;
    K480:2._Z_ = M481:2._Z_ - M480:2._Z_;
    K480:3._Z_ = M481:3._Z_ - M480:3._Z_;
    SAV._Z_ = 2;
    RSD._Z_ = 255;
}

/***
* Perform homing with index marks.
* NOTE: the reference marks are not in the center of the z-axis stroke.
*/
void func2(void)
{
    RST.(_Z_,_TILT_);
    PWR.(_Z_,_TILT_) = 1;
    IND._Z_;
    WTM._Z_;
    IND._TILT_;
    // Move axes to center of stroke
    MVE._Z_ = 0;
    WTM._Z_;
    MVETILT._TILT_ = 0L, 0L;
    // From here, if you do a SLS._TILT_ the curves should be symmetric.
}

```

14.3.2 Homing procedure

Using the mechanical end stops, the homing procedure could be the following:

```

# COMET special parameters
# ISO linear units: m, m/s, m/s2, V, A, s (used in case of interpolation)
# ISO rotary units: rad, rad/s, rad/s2, V, A, s
#define _Z_ 4
#define _TILT_ 5
void func0(void)
{
    K40._Z_ = 42;
    K40._TILT_ = 22;

```

```

SAV._Z_ = 2;
RSD._Z_ = 255;
}

/***
* Perform homing based on mechanical stops.
*/
void func1(void)
{
    RST._Z_,_TILT_;
    PWR._Z_,_TILT_ = 1;
    IND._Z_;
    SLS._Z_;
    XL0._Z_ = (ML37._Z_ + ML36._Z_)/2;
    MVE._Z_ = XL0._Z_; // Move to the center of the stroke
    WTM._Z_;
    SLS._TILT_;
    IND._TILT_;
}

```

The monitoring **M26** gives the logical offset applied to logical axes commuted after a homing 40 or 41.

M	Name	Comment	Unit
M26	Homing MIMO Offset	Gives the logical offset applied to logical axes commuted after a homing 40 or 41	[dpi]

Because the ZxT is a coupled system, one cannot assume the measured currents on the physical axes are in the same direction as the reference current on logical axes. This implies a small variation from the standard method on the homing procedure when looking for mechanical end stops.

14.4 Commands

The following commands have been disabled on the ZxT modules: ITP, AUT,INI, NEWFC, RESETFC, SETFC, TRE, TRM, TRM2D, TRR, TRS, UST and WTF. If an attempt is made to execute a disabled command, the **Unknown command** error will occur.

The following commands have been disabled on the tip-tilt axes: MVE, RMVE, SET, SMP, STA, STE_ABS, STE_ADD, STE_SUB, STI, WAB, WTP and WTS. Calling any of these command will raise the exception: '**Command not allowed on this axis**'.

A set of specific commands was created for the tip-tilt axes as described hereafter.

14.4.1 MVETILT and RMVETILT

The **MVETILT** and **RMVETILT** commands are used to initiate respectively an absolute and a relative movement on the tip-tilt axes.

Command	<P>	Comment
MVETILT.<axis>=<PL1>,<PL2>[,<PL3>]	<PL1> <PL2> [<PL3>]	Rx angle Ry angle Speed. It can also be set using the SPD register
RMVETILT.<axis>=<PL1>,<PL2>[,<PL3>]		

14.4.2 SETTILT

The **SETTILT** command is used to adjust the offset between dpi and upi values. The offset between dpi and upi given by the monitoring M4 can be set using this command.

Command	<P>	Comment
SETTILT.<axis>=<PL1>,<PL2>	<PL1> <PL2>	Rx offset Ry offset

14.4.3 STETILT_ADD, STETILT_SUB and STETILT_ABS

The **STETILT_ADD**, **STETILT_SUB** and **STETILT_ABS** commands are used to respectively add a step to the reference position, subtract a step to the reference position and modify the reference position to the specified (absolute) position.

Command	<P>	Comment
STETILT_ADD.<axis>=<PL1>,<PL2> STETILT_SUB.<axis>=<PL1>,<PL2> STETILT_ABS.<axis>=<PL1>,<PL2>	<PL1> <PL2>	Rx angle Ry angle

Remark: The **STETILT_ADD** command can be used with negative values. The effect would be the same as with the **STETILT_SUB** command.

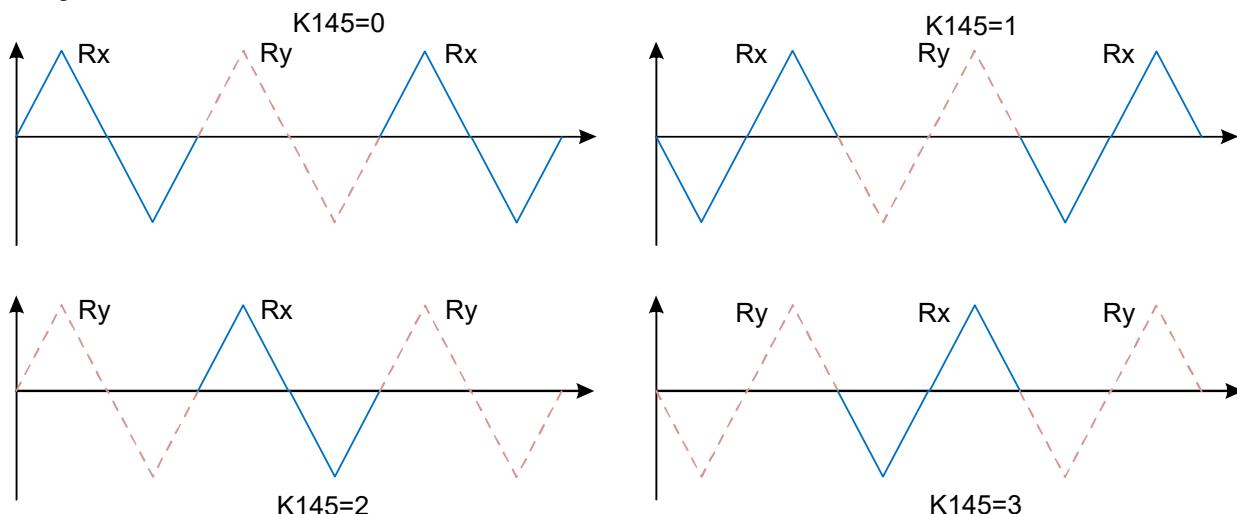
14.4.4 IND

The **IND** command to start a homing procedure. The concerned axis must be powered on and the homing mode is chosen using the parameter K40 (refer to §14.3).

14.4.5 SLS

The **SLS** command to search for limit strokes but it differs from the normal behavior on the tip-tilt axes. The operational mode depends on the parameter K145. The standard mechanical end stop detection use the sign of the moving direction to compare with the sign of the currents. This cannot be applied in MIMO so the SLS command uses the theoretical (logical) current set point compared to the parameter KF44 as criteria for the mechanical end stop detection.

The search of the limit stroke (SLS command) works as usual for the Z axis. For the tip-tilt axes, the following drawings show the different modes. All of them will search for limit stroke on Rx and Ry and then move back at the middle on the stroke. This command is commonly used to place the tip-tilt axes in the middle point of the moving area.



14.4.6 WTW

The **WTW** command works on the Tip-tilt axes. The tracking error used for the function (M2._TILT_) is the largest value between Rx and Ry. Only one depth for K38. _TILT_ and K39. _TILT_ are available.

14.4.7 WTWZXT

The **WTWZXT** command allows the user to wait for the edge of the chuck to be within a position window around the target (+/- K482) for a given time (K481). It checks a combination of tracking errors (M2) from Z, Rx and Ry, representing the worst tracking error on the chuck. For that purpose, it is required to define the chuck radius thanks to the parameter K483.

Command	<P1>	Comment
WTWZXT.<axis>=<P1>	0	Waits for M491 to be in windows defined by K481 and K482. If the theoretical trajectory is finished (bit#4 of SD1 at 0) when this command is executed, the controller acknowledges the command without testing if the motor is in the window or not. WTWZXT.<axis>=0 command must not be preceded by a WTM command because the WTWZXT command must be executed before reaching the theoretical position.
	1	Waits for M491 to be in windows defined by K481 and K482 without taking into account if the theoretical trajectory is finished or not
	2	Restarts a new wait for M491 to be in windows defined by K481 and K482 without taking into account if the theoretical trajectory is finished or not.

Remark: The **WTWZXT** command is only available on Z axis (first axis of the controller). If the command is sent on the tip-tilt axes, the **CMD NOT ALLOWED** error is raised (M64=482). In mode 0, the firmware checks Z, Rx and Ry. If one of the three axes is moving, the WTWZXT command is running. WTWZXT checks a combination of tracking errors (M2) from Z, Rx and Ry. It is not the distance between the current position and the target, as usual with the WTW command. Sending the WTWZXT command without parameter is equivalent to WTWZXT.<axis> = 0.

14.4.8 SAFRX

The **SAFRX** command allows the user to set in once a complete advanced filter for Rx.

Command	<P>	Comment
SAFRX.<axis>=<P1>,<PF2>,<PF3>,<PF4>,<PF5>,<PF6>	<P1> <PF2> <PF3> <PF4> <PF5> <PF6>	Depth of the advanced filter Value of KF104 Value of KF105 Value of KF106 Value of KF107 Value of KF108

Remark: The **BAD CMD PARAM** error (M64=70) is raised if the command does not have the right number of parameters and if the types of parameters are wrong.
This command can be used only for the tip-tilt axes.

14.4.9 SAFRY

The **SAFRY** command allows the user to set in once a complete advanced filter for Ry.

Command	<P>	Comment
SAFRY.<axis>=<P1>,<PF2>,<PF3>,<PF4>,<PF5>,<PF6>	<P1> <PF2> <PF3> <PF4> <PF5> <PF6>	Depth of the advanced filter Value of KF114 Value of KF115 Value of KF116 Value of KF117 Value of KF118

Remark: The **BAD CMD PARAM** error (M64=70) is raised if the command does not have the right number of parameters and if the types of parameters are wrong.
This command can be used only for the tip-tilt axes.

14.5 Parameters

The parameters K201, K202 and KL206 are disabled on the tip-tilt axes. Changing their values will have no effect.

The following parameters and common parameters are used to set the different ZxT functions:

K / C	Name	Comment
CF60	ZxT current saturation	Current saturation value (equivalent to KF60 but for the physical axes)
KL250	ZxT module feedforward (offset)	Set the distance between the position where the X and Y forces are applied and the mass center of the ZxT moving part.
K480	ZxT factory encoder offset	A small deviation may appear on the encoder's scales. This offset can be corrected using this register. A reboot is required after modification.
K481	ZxT global In-window time	Duration of the global ZxT window in MLTI (used with WTWZXT command)
K482	ZxT global In-window position	Position range of the global ZxT window (used with WTWZXT command)
K483	ZxT wafer radius	Wafer radius (used with WTWZXT command)
CF490	ZxT module feedforward, moving mass factor	This register should contain the mass of the moving part of the ZxT times the current factor Kt of the tip-tilt axes.
KF490	ZxT transfer matrix selection	The transformation matrices are unitary matrices. The radius in meters of the encoders from the center is set using this register. The minimum value is 10 millimeters.
KF492	ZxT current reference value	When K63 is set to one, the tip-tilt controller operates in current reference mode where the current set point for each Rx and Ry are taken from the two depths of this register.
KF493	ZxT current offset value	Set an offset to the current set point before the KF60 limitation. The two depths affects each of them Rx and Ry.
C495	ZxT feedforward propagation delay	The received acceleration (from RTVs) can be delayed to synchronize the effective stage movement with the correct correction to be applied. Each increment is a multiple of the PLTI base time (i.e. 50us).
K495	ZxT Sub-axis selection	Logical switch for tip-tilt axes. K495.1=0 is used to select Rx and K495.1=1 is used to select Ry.

Remark: Parameter K481 is similar to K38 for the WTW command and parameter K482 is similar as K39 for the WTW command

Here is the list of the parameters for the PID of the Rx and Ry axes of the tip-tilt.

- **Rx PID**

K	Alias	Name	Comment
KF1	KPP	Proportional gain	Proportional gain of the position loop
KF2	KDP	Speed feedback gain	Speed feedback gain of the position loop
KF4	KIP	Integrator gain	Integrator gain of the position loop

- **Rx advanced filters**

K	Name	Comment
KF104	Advanced filter coefficient	Advanced filter coefficient for term yk-1
KF105	Advanced filter coefficient	Advanced filter coefficient for term yk-2
KF106	Advanced filter coefficient	Advanced filter coefficient for term xk
KF107	Advanced filter coefficient	Advanced filter coefficient for term xk-1
KF108	Advanced filter coefficient	Advanced filter coefficient for term xk-2

Remark: The default values of those parameters make the advanced filters inactive. To make it active, use the 'Filter design' tool of the ComET software.

- Ry PID

K	Name	Comment
KF61	ZxT Ry Proportional gain	Proportional gain of the position loop for ZxT Ry
KF62	ZxT Ry Speed feedback gain	Speed feedback gain of the position loop for ZxT Ry
KF64	ZxT Ry Integrator gain	Integrator gain of the position loop for ZxT Ry

- Ry advanced filters

K	Name	Comment
KF114	ZxT Ry Advanced filter coefficient	ZxT Ry Advanced filter coefficient for term yk-1
KF115	ZxT Ry Advanced filter coefficient	ZxT Ry Advanced filter coefficient for term yk-2
KF116	ZxT Ry Advanced filter coefficient	ZxT Ry Advanced filter coefficient for term xk
KF117	ZxT Ry Advanced filter coefficient	ZxT Ry Advanced filter coefficient for term xk-1
KF118	ZxT Ry Advanced filter coefficient	ZxT Ry Advanced filter coefficient for term xk-2

Remark: The default values of those parameters make the advanced filters inactive. To make it active, use the 'Filter design' tool of the ComET software.

14.6 Monitorings

The following monitorings are used to monitor the different ZxT functions:

M	Name	Comment
M26	Homing MIMO offset	After executing the proper homing procedure, M26 shows the logical offset based on the encoder's reference marks.
M82	Controller max current	Depth 0 of M82 gives the logical maximum current which can be 3 or 4 depending on the ZxT configuration Z3TM or Z3TH (C490). Depth 1 of M82 gives the physical maximum current.
MF249	Theoretical current Iq for current loop	This monitoring shows an equivalent of MF30 on SISO systems but for logical axes.
M480	ZxT raw local encoders value	This monitoring gives the position of each physical linear encoders since the controller was powered. This register can only be read from the Z axis. The unit conversion is given accordingly to K77 and K241. This monitoring may be used during a factory calibration involving K480.
M481	ZxT latched reference marks	When a homing procedure using indexes is initiated, the position of each enabled encoders is latched at the reference mark position. These values are used to compute M26.
M491	ZxT global Z tracking error	Shows the ZxT global Z tracking error.
MF496	ZxT speed error (filtered)	The tip-tilt position controller uses a traditional PID controller where the speed error is found by taking the derivative of the position error after being filtered using a bi-quadratic filter.
MF497	ZxT stage feedforward	Shows the correction value added to the current set point (right before MF233). This value is affected by CF490, the moving mass and KL250, the lever arm distance. The feedforward compensation can be disabled by setting CF490 to a null value.
MF500	Logical axes MIMO correction factor	This conversion factor is given by the relation: $2\pi \frac{KF490}{K241^2 \cdot 10^{-9}}$
MF510	ZxT feedforward mass iso conversion	The unit conversion of CF490 includes values from both Z and tip-tilt axes. This factor is hard-coded and MF510 is used by EDI as the conversion factor: $MF510 = \frac{M82:0}{100 \cdot 2^{15}} \cdot \frac{K243 \cdot 10^9}{K242K241} \cdot 2^{10 + K77} \cdot MF500$

14.7 Errors

The following specific errors may occur:

- **ZXT BAD CONFIG** error (M64=480) meaning that the ZxT firmware is not properly configured. To avoid this error, check the following:

- C1=0
- K78.1=42
- Same K77 for all enabled encoders (4 encoders for Z3T^H and 3 for the Z3T^M).
- Same K241 for all enabled encoders.

- **CMD NOT ALLOWED** error (M64=482) meaning that the command is not allowed on this axis. An unauthorized command was executed on the tip-tilt axes. This error will occur if the MVE command is executed on the tip-tilt axes.

Remark: If **HOME NOT POSSIBLE** error (M64=69) occurs, it means the indexes were not found yet. A homing mode (K40=48) has been done on the tip-tilt axes before searching for the indexes on the Z axis. A IND.0 command should be done first. For example the following will generate an error because the IND:5 is not waiting for the end of IND:4. A WTM.4 will not help since one is addressing two different command managers.

```
K40.4=41;
IND.4;
K40.5=48;
IND.5;
```

14.8 Unit conversion

Some additional unit conversions were implemented specifically for the ZxT firmware.

14.8.1 Tip-tilt positions (dpi) for Rx and Ry

$$dpi_{iso} = \frac{dpi_{inc}}{\frac{K243}{K242} \cdot M241 \cdot M239 \cdot MF500}$$

where MF500 monitoring is the correction factor which includes the encoder's radius given by the parameter KF490.

We could also calculate the position in degree as follows:

$$dpi_{deg} = dpi_{inc} \cdot \frac{360 \cdot K241 \cdot 10^{-9}}{2\pi \cdot M241 \cdot KF490}$$

14.8.2 Current (cur/curPhysical)

The current conversion unit uses M82 monitoring which is the maximum measurable current on the controller. In a ZxT system, the logical current can be bigger than M82. A force in the Z direction involves several actuators and thus the force can be theoretically bigger than M82, e.g. 4 times larger for a Z3T^H and 3 times larger for a Z3T^M. So the unit conversion for the logical current is given by:

$$ci_{iso} = ci_{inc} \cdot \frac{M82:0}{100 \cdot 2^{15}}$$

And for the physical current, this relation is used:

$$ci_{iso} = ci_{inc} \cdot \frac{M82:1}{100 \cdot 2^{15}}$$

Please refer to M82 monitoring for more information.

Chapter D: Programming

15. Programming of the controller

15.1 Introduction

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.

A sequence is written in a specific ETEL language and can be stored in a text file on your PC. In order to be executed, the sequence has to be compiled, downloaded and stored in the controller. This is done thanks to ComET. The size of the memory dedicated to the compiled code is 390 kbytes per axis and the one for the source code is 130 kbytes per axis.

The execution of the sequence starts when a JMP command is sent to the controller. A sequence can also be executed at boot time, if the controller has been configured to automatically start the sequence.

A sequence can contain several routines. Each routine can be executed separately or can be combined with others to build a more complex program. For example, the controller can host a sequence with 5 routines, each one handling one of the 5 specific movements or actions needed by the application. At any time, the host PC can perform the needed movement or action by starting one of these routines on the controller. The sequence is also convenient to execute supervision tasks or to customize existing tasks (the homing process, for example). In the sequence, it can occur that some commands or registers writing are sent to the command manager (refer to [§2.5](#)). In this case, the thread execution is interrupted (same behavior than the yield function, refer to [§15.3.11.3](#)) and execution will continue when the command manager has executed it (next MLTI).

Before downloading a new firmware version, please refer to [§3.2.2](#) for the sequence management.

The parameter **K297** allows the users to select the mask for active thread information on M63 status (bit#31) and M60 (bit#15) (refer to [§8.10](#)).

K	Name	Comment
K297	Mask for active thread	Mask for active thread information on M63 status (bit#1 for thread1, bit#2 for thread2). It is a bit field, then the maximum value is 6.

- K297=0 by default. It clears bit#31 of M63 and bit#15 of M60.
- Bit#1 of K297 allows the user to show thread 1 running on bit#31 of M63 and bit#15 of M60.
- Bit#2 of K297 allows the user to show thread 2 running on bit#31 of M63 and bit#15 of M60.

15.2 Basic concepts

15.2.1 Source file

A sequence is a text file describing the instruction sequence thanks to a specific ETEL language. The sequence is stored in a .cseq file. This file extension is coming from 'compiled sequence' and it is also used to differentiate the AccurET and UltimET sequence from the .seq file used by the previous products. The file can be edited with a standard text editor (Notepad, Notepad++, ...) or by ComET. A sequence must always be stored in a single file. There is no possibility to handle multi-file source code.

15.2.2 Comments

In the sequence, each line starting with '//' is a comment and is not taken into account by the compiler. The same is for any text placed between a "/*" and a "*/".

Lines starting with '#' are special comments generated and managed by various ETEL tools. Please, do not use these syntax for your own comments.

15.2.3 Header

Each sequence containing ISO values must have a header defining the units available in the sequence (if the sequence does not contain ISO value, the header is not necessary). The header is composed of few lines starting with '#'. These lines can be generated by ComET by choosing 'Insert ISO header' and/or 'Insert version header' in the 'Edit' menu. The sequence developer can adapt it with the wished units by modifying them in ComET. To do so, click on 'File' and 'Preferences'.

15.2.4 Register and command

Each of the functions available in the controller, is managed by a set of registers (KF80, K177, ...) and commands (MVE, STA, PWR, IND, ...).

All registers and commands can be reached and executed by the ComET Terminal or by the application running on the host PC. In the same way, these registers and commands can be reached and executed by the sequence.

The ComET Terminal and the sequence have a very similar syntax. Any command available on the ComET Terminal can be used with the same syntax on the sequence. There are few exceptions that will be developed later in this manual.

15.2.5 Instruction delimiter

A semicolon (";") must be placed after each instruction (each expression statement). So, on a single text line it is possible to place several instructions separated by semicolons.

An instruction can be written on more than one text line. The 'Carriage return' (CR) or 'Line feed' (LF) characters ending each text line have no special meaning from the language point of view.

15.2.6 Immediate values

Each time a number is written in the code, the type of this number must be specified. There are 5 different types giving 5 different syntaxes.

Type	Syntax example	Description
Integer (32-bit#values)	123 or -200 or 0x12AB	Simple number, eventually preceded by the '0x' prefix for hexadecimal values.
Long (Integer 64-bit#value)	123L or -200L or 0x12ABL	As an integer 32-bit#value followed by the 'L' suffix.
Float (32-bit#values)	123F or 1.2F or -3.5F	Standard engineering notation followed by the 'F' suffix.
Double (float 40-bit#value)	123D or 1.2D or -3.5D	Standard engineering notation followed by the 'D' suffix.
ISO value	123.0 or 1.2 or -3.5	Standard engineering notation without any suffix.

The ISO values are values representing physical quantities. The units applied to these values are the basic ones defined by the ISO standard (m, s, A, kg, ...) except when a specific unit is defined in the sequence header (refer to [§15.2.3](#)). Refer to [§2.2.1.1](#) for more information about the registers.

ISO value can be used only when the physical unit is implicitly known. For example, when the KL212 (or ACC) register is set, the compiler knows implicitly that KL212 stores an acceleration and that the ISO unit is the meter/ S^2 . Thus, it is possible to write:

```
KL212:1 = 0.02;
```

Please note that the conversion is done during the compilation of the code. No conversion is allowed at run time. ISO value can be used as an argument of the iconv() lconv(), fconv() and dconv() functions. Refer to [§15.3.10](#) for more information.

15.2.7 C preprocessor define

It is possible to declare C preprocessor define in sequences.

Example:

```
#define AXIS_X 1
#define AXIS_Y 2

void func10(void)
{
    RST.AXIS_X;
    PWR.AXIS_X = 1;
}
```

However, there are some restrictions, comparing to C preprocessor define.

- It is not possible to declare a #define from another #define.
- It is not allowed to use identifier similar to an ETEL specific command or register

Example

```
#define AXIS_X 1
#define AXIS_Z AXIS_X + 1      // This is not allowed!
#define M0 0                   // This is not allowed!
#define SAV 4                  // This is not allowed!
```

15.2.8 Thread

It is possible to run several routines at the same time to organize your code and program separated tasks. This is similar to having several processors, each one executing a routine or a portion of the whole sequence. In practice, there is only one processor that operates as several virtual processors. Each virtual processor is called a 'thread'. The AccurET controller has 2 threads per axis. The monitoring **M97** allows the user to know which thread is currently active:

M	Name	Comment
M97	Sequence thread	Mask of the sequence threads currently active (bit#0 is not used, bit#1=1 if the thread 1 is active and bit#2=1 if the thread 2 is active)

Basically, only one thread is needed to execute the sequence, but with several threads, complex programming (multitasking) is possible.

The code of a thread can be shared. Actually, It is also possible to have two threads executing the same routines. For example, a routine computing the average on 2 values can be written. Two threads can perform very different tasks in which they need to compute an average. So, both threads will call this routine (refer to [§15.4](#) for more information).

15.2.9 User variables X

The user variables X are reserved to the user program. Each thread owns 512 variables.

Syntax:

X<number>[:<thread>]

Parameter	Value	Comment
number	0 to 511	User variable number
thread	1 or 2	0: X user variable of the current thread 1 or 2: User variable X of the designated thread

Remark: The [:<thread>] field is optional.

When the thread number is not specified or is equal to 0, it refers to the current thread. It enables the writing of code which can be simultaneously executed by several threads, each thread using its own X variables.

Example:

x1.0 = 44;

If it is executed by the thread 1, this instruction will set the value 44 in X1:1.0

If it is executed by the thread 2, this instruction will set the value 44 in X1:2.0

Etc...

This technique is useful to write subroutine for general use.

The user variables X are also useful to pass information or synchronize two threads. The threads can also read and write in the variables X of another thread. To do so, a thread number must be specified. For example, the instruction X1:2.0 = 22 will set the value 22 in the variable X1.0 of the thread number 2 whatever the thread executing the instruction.

The user variables X of each thread can also be read and written by the PC via the PCI port. There is only one difference in the syntax: when the thread is not specified (X1.0) or is equal to 0 (X1:0.0), it refers to the variables X of the thread 1 because the PCI port does not have its own variables set.

15.3 Structured code

When the applications become complex, it is important to have a language helping the developer to organize his code. A good structure often results in a code more reliable and easier to maintain and reuse. The C language is very popular in the industry and has inspired lots of other languages being used in the industry, as Java, C++, C#, etc.

To improve the sequence language and introduce a structured syntax, a solution based on the industry standards has been built. ETEL's sequences integrate the basic functionalities of the C language in what is called the 'structured code'. The developer being familiar with the C language will quickly feel comfortable with the ETEL sequences. However, basic knowledges in the C language are recommended to enjoy the advantages of the structured code.

The ETEL sequence language has to be seen as a subset of the C language. It includes all the features of the C having a benefit for embedded motion control applications. Pointers, for example, are not available in sequences, because they are not suitable for embedded sequences and can easily decrease reliability. On the other hand, some specific syntax has been added to give a direct access to all ETEL commands and registers (PWR, MVE, WTM, ...) and, in this way, simplify the development of motion control applications.

The combination of C language basis with the ETEL commands and registers results in a very powerful language, easy to learn and perfectly suitable for the development of motion control application.

15.3.1 Identifiers

An identifier is a name used to identify functions, function arguments and variables. The name is a case sensitive character string, containing any of the following character:

- Alphabetic characters (from 'a' to 'z' and from 'A' to 'Z')
- Digits (from 0 to 9)
- Underline sign ('_')

An identifier always starts with an alphabetic character.

Remark: It is not possible to declare a variable with a name similar to any register name.

15.3.2 Types

The available types are integer, long, float and double. These types are used to define variables, function arguments and function returns.

Type	Description
int	Integer value, 32-bit, signed
long	Integer value, 64-bit, signed
float	Floating-point value, 32-bit
double	Floating-point value, 64-bit

Remark: At the controller level, the 'Double' type value are truncated to 40 bits.

Example:

```

float fct(int v)
{
    return 1;                                //Not allowed
    return 1.3F;
}
void func10(void)
{
    X0.0 = 1.5F;                            //is accepted, X0.0 will be equal to 1
    X0.0 = 1.6D;                            //is accepted, X0.0 will be equal to 1
    X0.0 = 0xFFFFFFFFFFFFFFFL;             //is accepted, X0.0 will be equal to -1
    X0.0 = fct(1);                          //is accepted, X0.0 will be equal to 1
    X0.0 = fct(1.0F);                      //Not allowed
}

```

15.3.3 Global variables

The global variable are variable accessible from any function present in the sequence. They must be defined before any function definition.

The syntax used to define a global variable is the following (same as in C):

<type> <variable name>;

It is possible to specify a value which the variable is initialized with:

<type> <variable name> = <immediate value>;

Example:

```

int loop_counter;
float speed = 1.2F;

```

The <type> is one of the 4 types described in [§15.3.2](#).

The <variable name> is an identifier described in [§15.3.1](#).

The <immediate value> must comply with [§15.2.6](#).

Global variables can be used in expressions (refer to [§15.3.7](#)).

Caution: When a global variables is initialized with a value, the initialization takes place only when the sequence is downloaded or when the sequence is restored from the flash (at boot time or with the RES command). When a sequence starts, the values stored in the global variable are not modified (are not initialized). They keep the value set by any sequence executed before. The user's variables X have exactly the same behavior.

15.3.4 Registers

In the sequence code, it is possible to have a direct access to all existing registers (X, M, K, XL, KL, ...). These registers work as global variables but they do not need to be declared.

For example, is it possible to write expression statements involving registers:

```
x1 = M6 + X4 / 2;
```

This possibility is not present in the standard C language.

15.3.5 Functions

15.3.5.1 Function definition

A function is a piece of code (a routine) having a name and, possibly, some arguments and a return value. All code (all routines) must be placed in a function. Functions are located after the definition of global variables.

Here is the syntax of the definition of a function:

```
<return type><function name>(<arguments>)
{
    <function body>
}
```

The <return type> can be one of the 4 types described in [§15.3.2](#) or 'void' when there is no return value. The <function name> is an identifier (refer to [§15.3.1](#)).

The <arguments> are a list of coma separated elements, each one having a type and a name. The type is conformed to [§15.3.2](#) and the name is an identifier conformed to [§15.3.1](#). If no argument is needed, the arguments must be replaced by the 'void' keyword.

Example:

```
void my_routine_to_perform_mouvements(void)
{
    // routine code
}

int get_step_count(void)
{
    // routine code
}

int make_average(int value1, int value2)
{
    // routine code
}
```

When a <return type> is defined, the function returns a value. Inside the function body, the return statement must be used to specify which value has to be returned (refer to [§15.3.9.10](#)).

The <function body> contains the definition of the local variables, followed by a list of statements. Local variables are defined in the same way as global variables (refer to [§15.3.3](#)).

Example:

```
void move_to_park_position(void)
{
    int actual_pos;           // variable storing the actual position
    int target_pos = 10000L;  // variable storing the target position
    ...
    // place the list of statements here
}
```

Local variables are available inside the function body only. Refer to [§15.3.9](#) for informations about the statements.

15.3.5.2 Function call

Functions can be called from within an expression.

Syntax:

<function name>(<argument 1>, <argument 2>, ...)

<argument 1>, <argument 2>, etc. can be immediate values, variables, registers or any other expressions.

When a function is called, the function body is executed. If a value is returned, this value takes the place of the function inside the expression.

Example:

```
foo = 2 * average(x1, 2 * bar, abs(x2)) + 1
```

In this expression 'average' is a function. It is called with 3 arguments. The first is 'X1', the second is '2 * bar' and the third is 'abs(X2)'. The result of this function (the returned value) will then be multiplied by 2 and incremented by 1. Refer also to [§15.4.1](#).

15.3.5.3 Interface functions

Some functions can be called from within the sequence itself, and some from the external application like ComET or another EDI-based host PC application.

Functions called from within the sequence itself do not have any naming or declaration constraints other than the specifications above.

Functions that are meant to be called from outside the sequence, must comply with the following rules:

- They must be named **func<n>** where n is any integer from 0 to 1023.
- They cannot return any values nor accept any parameters.

As a result their prototype must look like: **void func<n>(void)**

15.3.5.4 Function declaration

When a call to a function is made, this function must be known. In the source file, this means that the definition of a function should precede its call.

Example:

```
int average(int a, int b) {
    return (a + b) / 2;
}

void func40(void) {
    x1 = average(m7.0, m7.1);      // Call a function defined above => OK
```

```
x2 = max(M7.0, M7.1);           // Call a function defined below => NOT ALLOWED!
// This function is not yet known because its definition is done later in the file
}

int max(int a, int b) {
    if (a > b)
        return a;
    return b;
}
```

In this example, it is quite simple to define the 'max' function before 'func40', avoiding any problem. But depending on the complexity of the code, defining the functions in the right order can result in a headache.

This problem, well known in the C language, can be solved by declaring the functions at the beginning of the source file.

Example:

```
int max(int a, int b);           // This is a function declaration. It just tells that the function
                                // exists, but the whole definition of the function will be
                                // done later.

int average(int a, int b) {      // Definition of the 'average' function.
    return (a + b) / 2;
}

void func40(void) {              // Definition of the 'func40' function.
    X1 = average(M7.0, M7.1);   // Call a function defined above => OK
    X2 = max(M7.0, M7.1);      // Call a function defined below but declared above => OK
                                // The 'max' function is known here because of the
                                // declaration above.

}
int max(int a, int b) {          // This is the definition of the function already declared
                                // at the beginning of the file.
    if (a > b)
        return a;
    return b;
}
```

Please, do not mix the terms 'declaration' and 'definition'. Function declarations must be located before any function definition and after any definition of global variables.

15.3.6 Operators

The operators define how to combine together the various elements of an expression. A typical example is the sum ('+' operator). It allows the user to add two values. For the operator point of view, the elements on which the operator acts are called operand. The operands can be immediate values, variables or registers.

Example:

x1 + 12	with: 'x1' is the operand (immediate value) '+' is the operator (addition) '12' is the operand (register)
foo / 13.4F	with: 'foo' is the operand (variable) '/' is the operator (division) '13.4F' is the operand (immediate value)

The operand can also be an expression that, once executed, results in a value.

Example:

x1 + (12 * max(x2, 20))	with: 'x1' is the operand (register)
-------------------------	--------------------------------------

'+' is the operator (division)
 '(12 * max(x2, 20))' is the operand (expression)

Each operator performs an operation on its operands. This normally results in a value which takes the place of the operator and operands in the expression. Each operator has a priority (refer to [§15.3.7](#)).

15.3.6.1 Unary operators

Unary operators are operators acting on a single operand. Here is the syntax:

<operator><operand>

The <operand> can be an immediate value, a variable, a register or any expression resulting in a value.

Example:

-x1 with: '-' is the operator
 'x1' is the operand

Here is the list of unary operators:

Operator	Priority	Description
-	12	Sign <right operand>
~	12	Bitwise NOT <right operand>

15.3.6.2 Binary operators

Binary operators are operators acting on two operands. The typical example is the sum ('+' operator). In an expression, the operator is located between two operands. Here is the syntax:

<operand> <operator> <operand>

The <operand> can be an immediate value, a variable, a register or any expression resulting in a value.

Example:

x1 + 12

Here is the list of binary operators:

Operator	Priority	Description
+	10	Increment <left operand> by <right operand>
-	10	Decrement <left operand> by <right operand>
*	11	Multiply <left operand> by <right operand>
/	11	Divide <left operand> by <right operand>
%	11	<left operand> modulo <right operand>
&	6	<left operand> bitwise AND <right operand>
	4	<left operand> bitwise OR <right operand>
<<	9	<left operand> left shift <right operand>
>>	9	<left operand> right shift <right operand>
^	5	<left operand> XOR <right operand>

Remark: Refer also to [§8.13](#) for more information about the AND and OR operators.

15.3.6.3 Boolean operators

Boolean operators are binary operators which result is true or false. They are often used to compare two operands.

Example:

```
x1 . 0 > 12
```

Here, the '**>**' operator compares '**x1 . 0**' and '**12**'. The result of this comparison is not a value. It can only be true or false.

Here is a list of all binary boolean operators:

Operator	Priority	Description
==	7	<left operand> test equal to <right operand>
!=	7	<left operand> different <right operand>
<	8	<left operand> less than <right operand>
>	8	<left operand> greater than <right operand>
>=	8	<left operand> greater or equal <right operand>
<=	8	<left operand> less or equal <right operand>
&&	3	<left operand> logical and <right operand>
 	2	<left operand> logical or <right operand>

Boolean operators are allowed only in boolean expressions (refer to [§15.3.7.1](#)).

15.3.6.4 Assignment operators

All operators described above are performing operations on their operand without affecting the operand itself. Once the operation is performed, the resulting value takes the place of the operator and operands in the expression.

Assignment operators are affecting their left operand. The typical example in the assignment operator '='. When we write, X1 = 23, we expect to change the value stored in the register X1.

The syntax is similar to the one with a binary operator:

<left operand> <assignment operator> <right operand>

The <left operand> can be a variable or a register only!

The <right operand> can be an immediate value, a variable, a register or any expression resulting in a value.

Here is the list of assignment operators:

Operator	Priority	Description
=	1	Assign the value of <right operand> to the <left operand>
+=	1	Increment <left operand> by <right operand>
-=	1	Decrement <left operand> by <right operand>
*=	1	Multiply <left operand> by <right operand>
/=	1	Divide <left operand> by <right operand>
%=	1	<left operand> = <left operand> modulo <right operand>
<<=	1	Left shift <left operand> by <right operand>
>>=	1	Right shift <left operand> by <right operand>

Operator	Priority	Description
<code>&=</code>	1	AND <left operand> with <right operand>
<code> =</code>	1	OR <left operand> with <right operand>
<code>^=</code>	1	XOR <left operand> with <right operand>

15.3.6.5 Increment and decrement operators

To increment or decrement a register or a variable, the ‘`++`’ and ‘`--`’ operators can be used. These operators are unary assignment operators. Their operand can be a variable or a register only.

Example:

```

x1.0++;
// increment X1.0 by one (same as X1.0 = X1.0 + 1;)

x2.0--;
// decrement X2.0 by one (same as X2.0 = X2.0 - 1;)

foo++;
// increment the 'foo' variable by one (same as foo = foo + 1;)

bar--;
// decrement the 'bar' variable by one (same as bar = bar - 1;)

```

These operators can also be combined with other operators inside an expression. In this case, their priority is the higher one. Once evaluated, these operators result to the value of their operand before or after being incremented or decremented. This depends on which side of the operand the operator is located.

Example:

```

x1.0 = x2.0++;
// increment X2.0 by one. The result of this operation is X2.0 before being
// incremented. So, X1.0 is assigned with the value of X2.0 before being
// incremented.

x1.0 = ++x2.0;
// increment X2.0 by one. The result of this operation is X2.0 after being
// incremented. So, X1.0 is assigned with the value of X2.0 after being
// incremented.

foo = bar--;
// decrement 'bar' by one. The result of this operation is 'bar' before being
// decremented. So, 'foo' is assigned with the value of 'bar' before being
// decremented.

foo = 23 * (--bar) + 1; // complex expression

```

15.3.7 Expressions

An expression is a simple or complex operation involving variables, registers, operators and function calls.

Example:

```
mvt_step = x1.0 + (M6.0 * scale_factor + 1) / get_step_count(120.0F);
```

In this example, the expression combines two variables ('`mvt_step`' and '`scale_factor`'), two registers ('`X1.0`' and '`M6.0`'), two immediate values ('1' and '120.0F'), a function call ('`get_step_count`') and several operators (=, +, * and /). The usage of brackets helps to group operations and define their priority, as we usually do in standard mathematics expressions.

An expression can be just an assignment of registers or variables:

```
x1.0 = 12;
foo = 23D;
```

Or a simple function call:

```
do_step(); // do_step is a function having no argument
```

In the example above, all expressions end by a semi-colon (;) because they are expression statements (refer to [§15.3.9.1](#)). Expressions do not always end with a semi-colon. When they are used as an argument in a function call or in statements, for example, they do not have any semi-colon (;).

Example:

```
for (i=0; i<10; i++) {           // i++ is an expression
    DOUT.0 = i & 0x0001;
    yield();
}
```

When several operators appear in an expression, their priority will determine the order on which they are applied.

Example:

```
x1 = x2 + x3 * x4;
```

There are two operators, '+' and '**'. The '**' operator has a higher priority than the '+' operator, so it will be executed first. This expression is so identical to:

```
x1 = x2 + (x3 * x4);
```

Once again, the execution order can be modified thanks to the brackets.

Example:

```
x1 = (x2 + x3) * x4;      // the sum is executed first, its result is multiplied by X4
```

15.3.7.1 Boolean expressions

Boolean expressions are expressions whose result can be true or false.

Example:

```
x1.0 < 12
```

This expression, once evaluated, does not result in a value. The result is true if X1.0 is less than 12, or false otherwise.

A boolean expression can combine several tests like this:

```
(x1.0 < 12) && (i == 100) || ((x2.0 + 200) > x1)
```

Boolean operators are allowed only in boolean expressions. Boolean expressions are used in 'if', 'while', 'do' and 'for' statements (refer to [§15.3.9](#) for more information).

15.3.7.2 Commands

All the controller commands described in the previous chapters can be executed in a sequence.

Examples:

```
PWR.0 = 1
IND.0
MVE.0 = 12000L + 2 * x1.0
WTM.0
WBS.0 = DIN, 1 << x2.0
```

From the language point of view, these commands are expressions.

15.3.8 Commands behavior

All the commands listed in [§16](#), that are put in a sequence go through the command manager (performing a yield) exception for:

- written registers X, L, EL et LD (no yield)
- all the other commands listed in [§15.3](#) (except for [§15.3.7.2](#)) do not go through the command manager (no yield).

15.3.9 Statements

Statements are used to describe the operations to perform. They are also giving a structure to the code and they are used to control the execution of the sequence. Statements are located in the function body, just after the definition of the local variables (refer to [§15.3.5](#)). Most of the statements available in the sequence language are very similar to the ones defined by the C language.

15.3.9.1 Expression statements

Basically an expression statement is just an expression followed by a semi-colon (;). Most of the time, the expression contains an assignment operator or a function call.

Example:

```
x1 = 23;
actual_position = M6.0 * scale_factor(x2);
start_movement(x3);
MVE.0 = 12000L;
PWR.0=1;
```

Refer to [§15.3.7](#) for more information about the expressions.

15.3.9.2 The block statement

A block statement allows the user to group any number of statements into one statement. All statements enclosed within a single set of braces ('{', '}') are treated as a single statement. You can use a block wherever a single statement is allowed.

Syntax:

```
{
    <statement 1>
    <statement 2>
    <statement 3>
    ...
}
```

In the C language, it is possible to define local variables inside a block statement. This is not allowed in a ETEL sequence.

15.3.9.3 The 'if' statement

The 'if' statement allows the user to execute or not a statement depending on the value of a boolean expression.

Syntax:

```
if (<boolean expression>
    <statement 1>
```

or

```
if (<boolean expression>)
```

```
<statement 1>
else
    <statement 2>
```

The <statement 1> is executed only if the <boolean expression> is true, otherwise <statement 2> is executed. When more than one statement must be executed, <statement 1> and <statement 2> must be block statements.

Example:

```
if (X1.0 == 12) {
    // all the following statements are executed if X1.0 is 12
    <statement 1a>
    <statement 1b>
    <statement 1c>
    ...
}
```

The statement located after 'else' can be an 'if' statement. This let you manage multiple choices.

Example:

```
if (X1.0 < 10)
    MVE.0 = 12000L;
else if (X1.0 > 10)
    MVE.0 = -12000L;
else // implicitly, X1.0 value is 10
    BRK.0;
```

15.3.9.4 The 'switch' statement

The 'switch' statement allows the user to select which statement must be executed depending on the value of an expression. It works in collaboration with the 'case' and 'break' statements.

Syntax:

```
switch (<expression>) {
    case <value 1>:
        <statement 1a>
        <statement 1b>
        ...
        break;
    case <value 2>:
        <statement 2a>
        <statement 2b>
        ...
        break;
    case ...
    ...
    default:
        <statement 3a>
        <statement 3b>
        ...
        break;
}
```

<statement 1a>, <statement 1b> and the following ones (...) are executed only if the <expression> is equal to <value 1>.

<statement 2a>, <statement 2b> and the following ones (...) are executed only if the <expression> is equal to <value 2>.

It is possible to add as many 'case' statements as needed. Each 'case' statement must have a unique value.

If the <expression> does not correspond to any value defined by the 'case' statements, <statement 3a>, <statement 3b> and the following ones (...) present under 'default', are executed.

The use of 'break' statements is not mandatory. If a 'break' statement is not present, all the following statements will be executed by ignoring the case.

Example:

```
switch (X1.0) {
    case 1:
        <statement 1a>
        <statement 1b>
        // no break here
    case 2:
        <statement 2a>
}
```

In this example, if X1.0 is equal to 1, <statement 1a>, <statement 1b> and <statement 2a> will be executed. If X1.0 is equal to 2, only <statement 2a> will be executed.

15.3.9.5 The 'while' statement

A 'while' statement repeatedly executes a given statement until the boolean expression is false.

Syntax:

```
while (<boolean expression>)
    <statement>
```

<statement> is executed several times while <boolean expression> is true. <boolean expression> is evaluated before each execution of <statement> to determine whether or not the execution of <statement> must be done. If <boolean expression> is false since the first evaluation, <statement> is never executed.

When more than one statement must be repeatedly executed, <statement> must be a block statement.

Example:

```
while (i < 10) {
    // execute the following instructions until i >= 10
    MVE.0 = 10000L * i;
    WTM.0;
    i++; // increment i
}
```

A 'break' or 'return' statement causes a 'while' statement to end, even when the <boolean expression> is true.

15.3.9.6 The 'do' statement

A 'do' statement repeatedly executes a statement until the conditional expression is false. Because of the order of processing, the statement is executed at least once.

Syntax:

```
do
    <statement>
    while (<boolean expression>);
```

<statement> is executed before <boolean expression> is evaluated. Further processing of the 'do' statement depends on the value of <boolean expression>. If <boolean expression> is true, the statement is executed again. When <boolean expression> is false, the statement ends.

When more than one statement must be repeatedly executed, <statement> must be a block statement.

A 'break' or 'return' statement causes a 'do' statement to end, even when the <boolean expression> is true.

15.3.9.7 The 'for' statement

A 'for' statement repeatedly executes a statement, as the 'while' and 'do' statements do. The control of the execution depends on three expressions:

- An expression is evaluated before the first iteration of the statement
- A boolean expression determines whether or not the statement should be processed. The statement is repeatedly executed if this boolean expression is true
- An expression is evaluated after each iteration of the statement (often used to increment an iteration counter)

Syntax:

```
for (<expression 1>; <boolean expression>; <expression 2>)
    <statement>
```

<expression 1> is evaluated only once, before <statement> is executed for the first time. You can use this expression to initialize a variable (typically the iteration counter). If you do not want to evaluate an expression prior to the first iteration of the statement, this expression can be omitted.

<boolean expression> is evaluated before each iteration of the statement. If it is false, the statement is not processed and control moves to the next statement following the 'for' statement. If <boolean expression> is true, the statement is processed. If <boolean expression> is omitted, it is like if an always true expression was present, and the 'for' statement is not terminated by failure of this condition.

<expression 2> is evaluated after each iteration of the statement. This expression is often used for incrementing an iteration counter. This expression is optional.

When more than one statement must be repeatedly executed, <statement> must be a block statement.

A 'break' or 'return' statement causes a 'for' statement to end, even when the <boolean expression> is true. If the <boolean expression> is omitted, a 'break' or 'return' statement must be used to end the 'for' statement.

Example:

```
int do_movement_steps(int number_of_steps) {
    int i;
    for (i=0; i<number_of_steps; i++) {
        // Do the following movements number_of_steps times.
        // On the first iteration, i = 0, On the second, i = 1, etc.
        // On the last iteration, i = number_of_steps - 1.
        RMVE.0 = +4600L;
        WTM.0;
        // If the DIN is set, stop to make steps (exit from the loop).
        if ((DIN & 0x02) != 0) {
            i++;
            break;
        }
    }
    return i; // Return the real number of step done.
}
```

15.3.9.8 The 'break' statement

A 'break' statement ends a 'switch', 'do', 'for' or 'while' statement and exits from it at any point other than the logical end. A 'break' may only appear on one of these statements.

Syntax:

```
break;
```

In a 'switch' statement, the break passes control out of the switch body to the next statement outside the 'switch' statement (refer to [§15.3.9.4](#)).

In a 'while', 'do' or 'for' statement, the 'break' statement ends the loop and moves control to the next statement outside the loop.

Example:

```
for (;;) { // no end condition, iterate until a break is reached
    if ((DIN & 0x01) != 0) // test the DIN value
        break; // exit the loop
    yield(); // wait next controller cycle
}
```

In this example, we stay in the loop until the DIN has been set. This is similar to waiting the DIN to be set.

15.3.9.9 The 'continue' statement

A 'continue' statement ends the current iteration of a 'while', 'do' or 'for' statement. Program control is passed from the 'continue' statement to the end of the body loop. A 'continue' statement can only appear within the body of these statements.

Syntax:

```
continue;
```

In a 'for' statement, for example, when the 'continue' statement is reached, the control moves to the third expression in the condition part of the statement (the part enclosed in the brackets), then to the second expression (boolean expression) in the condition part of the statement.

15.3.9.10 The 'return' statement

A 'return' statement ends the processing of the current function and returns control to the caller of the function.

Syntax:

```
return;
```

or

```
return <expression>;
```

A function with a non-void return type should always include a 'return' statement, containing an expression.

For a function of return type 'void', a 'return' statement is not strictly necessary. If the end of such a function is reached without encountering a 'return' statement, control is passed to the caller as if a 'return' statement without an expression were encountered. In other words, an implicit return takes place on completion of the final statement, and controls automatically returns to the calling function. If a 'return' statement is used, it must not contain an expression.

15.3.10 Units conversion

To convert an ISO value to an increment there are special conversion functions. For each data type there is a specific conversion function.

Increment value type returned by the function	Conversion function
int	iconv
long	lconv
float	fconv
double	dconv

The conversion functions accept either two or three arguments:

int iconv(isoValue, _unit) long lconv(isoValue, _unit) float fconv(isoValue, _unit) double dconv(isoValue, _unit)	int iconv(isoValue, _unit, axisNumber) long lconv(isoValue, _unit, axisNumber) float fconv(isoValue, _unit, axisNumber) double dconv(isoValue, _unit, axisNumber)
--	--

If no axis number is given, the compiler does the conversion for the axis on which the sequence is running.

Example:

```
x1.0 = iconv(0.02, _upi);      // convert 0.02 meters into user position increments and put
                                // the value in X1.0
```

These conversion functions work only with immediate ISO values (refer to [§15.2.6](#)) on the first argument. It is not possible to make conversions of value stored in variables or registers, or resulting from expressions:

```
x1.0 = iconv(x2.0, _upi);      // this usage of iconv() is NOT ALLOWED!
x1.0 = iconv(2 * bar, _upi);    // this usage of iconv() is NOT ALLOWED!
x1.0 = iconv(0.2F, _upi, 3);    // this usage of iconv() is NOT ALLOWED!
                                // the first argument must be an iso value
```

Here is the meaning of the main units:

Units	Meaning
cur	Current
cur2	Current * Current
milti	Number of MLTI in the given time (e.g. used for WTT)
msec	Conversion for register in millisecond
plti	Number of PLTI in the given time
temp	Conversion for temperature
uai	User Acceleration Increment (e.g. used with KL212)
ufai	User Friendly Acceleration Increment (Interpolation)

Units	Meaning
upi	User Friendly Position Increment (Interpolation)
ufsi	User Friendly Speed Increment (Interpolation)
ufti	User Friendly Time Increment (Interpolation)
upi	User Position Increment (e.g. used for MVE or KL210)
upi2	User Position Increment for secondary encoder
usi	User Speed Increment (e.g. used with KL211)
volt	Voltage

15.3.11 Predefined functions

Functions (refer to [§15.3.5](#)) can be defined by the developer and can be called from the sequence itself. ETEL's sequences have also a set of predefined function. This is similar to the standard library available when programming in C, where a set of functions is available to the C developer. In C, the developer has to insert a #include directive to have access to the standard library. On the ETEL sequences, predefined functions are available without doing anything.

Here is the list of predefined functions with their prototype (arguments and return types) and their behavior. The names of these functions have been chosen as close as possible to the ones of the standard C library.

15.3.11.1 start

This function starts a thread. On AccurET, there are 2 threads available: the number 1 and 2. The thread_number argument defines which one of these 2 threads is started. When started, the thread executes (call) the function given as first argument (refer to [§15.4.4](#) for more information).

Prototype:

```
void start(function f, int thread_number);
```

15.3.11.2 exit

This function stops the thread that calls it. Calling this function has the same behavior of executing the END command.

Prototype:

```
void exit(void);
```

15.3.11.3 *yield*

When this function is called, the execution of the calling thread is suspended until the next controller cycle.

Prototype:

```
void yield(void);
```

Example:

```
// In the following code, X1.0 is incremented by 1 on each controller cycle.  
// In other words, X1.0 is counting the number of cycles.  
for ( ; ; ) {  
    X1.0++;  
    yield();  
}
```

Calling this function has the same behavior of executing the YLD command.

15.3.11.4 *abs, labs, fabs and fabsf*

These functions compute and return the absolute value of the value given as argument. Each function has a specific value type.

Prototype:

```
int abs(int value);  
long labs(long val);  
double fabs(double val);  
float fabsf(float val);
```

15.3.11.5 *sqrt and sqrtf*

The sqrt function returns the square root value of the value given as argument.

Prototype:

```
double sqrt(double val);  
float sqrtf(float val);
```

15.3.11.6 *Trigonometric functions*

These functions compute the basic trigonometric operations. The angles must be expressed in radians.

Prototype:

```
float sinf(float val);  
float cosf(float val);  
float tanf(float val);  
float asinf(float val);  
float acosf(float val);  
float atanf(float val);  
float atan2f(float y, float x);
```

15.3.11.7 *rand*

The rand function returns a pseudo-random integer value in the range $[0, 2^{31} - 1]$.

Prototype:

```
int rand(void);
```

15.4 How to start a sequence

There are several ways to start a sequence thread. The basic way is to send a JMP command to the AccurET or UltimET controller. This command can be sent from ComET, from an application running on the PC or from a sequence running on the UltimET. It is also possible to start a sequence thread automatically when the AccurET or UltimET controller is switched on. Finally, a sequence thread can be started from another thread. All these method are described in the next paragraphs.

It is not allowed to start a sequence thread that is already running. To know how to stop a running thread, refer to [§15.5](#).

15.4.1 Start a sequence thread from ComET

You can start a sequence on a controller by sending a JMP command. This can be done from the ComET's Terminal. The syntax of the JMP command is as following:

Command	<P1>	<P2>	Comment
JMP.<axis> = <P1>[,<P2>]	n° of label	thread	Unconditional jump of a thread defined by <P2> to a label defined by <P1>

The [<thread>] parameter is optional.

Example:

To start a sequence thread on the axis 5, the following command must be sent;

JMP .5 = 25

When the axis 5 receives this command, it starts the execution of the function named 'func25' in the first sequence thread (thread number 1).

The second parameter of the command indicates which thread has to be started. There are two threads available on the AccurET controllers, so, the value of this parameter can be 1 or 2. If this parameter is omitted or set to zero, the thread 1 is started.

Example:

JMP .8 = 20 , 2

This command starts the execution of the function named 'func20' in the thread 2 of the axis 8.

15.4.2 Start a sequence thread from an application running on the PC

Applications running on the PC are normally communicating with AccurET or UltimET controllers through the EDI libraries. Thanks to these library, it is possible to start a sequence thread on any controller. To do so, the application has to call `dsa_execute_sequence_s()` or `dsa_execute_sequence_a()`. Please refer to the '**EDI User Manual**' for more information.

15.4.3 Start a sequence on the AccurET from a sequence present in the UltimET

The JMP command can be used on sequences as well as on the ComET's Terminal. A sequence in the UltimET can then start a sequence thread on an AccurET by executing the command `JMP.<axis> = <label>, <thread>`.

15.4.4 Start a sequence thread from another thread

To take advantage of the multi-thread architecture, a sequence thread must be able to start another sequence thread. This can be done with the function 'start()'.

The 'start()' function has two arguments: the function executed by the new thread and the new thread number.

Example:

```

void supervisor(void)
{
    for (;;) {
        if (M6.0 > X1.0)
            DOUT.0 = 0x10;
    }
}
void func20(void)
{
    X1.0 = 12000;
    start(supervisor, 2);      // start the thread number 2 to execute supervisor()
    MVE.0 = 0L;
    WTM.0;
    MVE.0 = 25000L;
    WTM.0;
    END.0 = 2;                // stop the thread number 2
}

```

In this example, the function named 'func20' starts a new thread (the thread number 2) that executes the function named 'supervisor'. Both functions run then in parallel (at the same time).

15.4.5 Start a sequence thread automatically

When the controller is switched on, it checks for the presence of a function named 'autostart'. If it is found, the thread number 1 is automatically started to execute the autostart function.

The 'autostart' function is a function called from outside the sequence itself, so it must have the following prototype: **void autostart(void)**

15.5 How to stop a sequence thread

The **END** command allows the user to definitively stop the execution of a sequence's thread. The END command can be sent from ComET, from the host PC application or from another sequence.

Command	Comment
END.<axis>=[<P1>]	Stops the sequence's thread defined by <P1> or stops all threads if command without <P1>

Remark: This feature represents a compatibility break if the command END.<axis> was used without parameter with version \leq 2.07B.

HLB, HLO and HLT commands stop the execution of the sequence (refer to [§8.16](#)). By default, in this case, all threads are stopped by a single command. The execution can also be stopped in case of error (refer to [§15.6](#)). However, with the parameter **K144**, it is possible for a thread to be not stopped by the HLB, HLO and HLT commands (normal or urgent).

K	Bit#	Value	Comment
K144	1	2	Mask for thread 1: if set to 1, the thread 1 is not stopped by HLT, HLB or HLO command.
	2	4	Mask for thread 2: if set to 1, the thread 2 is not stopped by HLT, HLB or HLO command.

The default value of the parameter K144 is 0 and then it has the same behavior as for the firmware versions \leq 2.07B (all thread are stopped by HLT, HLB or HLO command).

15.6 Error management

When an error occurs on the AccurET controller, each active thread can react in a way defined by the user.

The possible reactions are:

- Stop the thread.
- Jump to an error routine.
- Ignore the error and just continue the execution.

The error routine is a piece of a specific function in the sequence. Each thread can have its own error routine and its own reaction to the error. This is configured thanks to the parameter **K142**.

Syntax:

K142:thread.<axis>=<label_error>.

K	<label_error>	Comment
K142	<routine_number> -1 -2	The thread is stopped and the error routine is executed. No label error is defined, the thread is stopped in case of error. The routine called errhandler is executed. The thread is not stopped in case of error. The execution just continues and no error routine is executed.

<routine number> is a number from 0 to 1023. It identifies the error routine to be executed. The error routine is a function named 'func<routine number>'. Each depth of K142 represents a thread:

K142:0	Not used
K142:1	Label of the error routine of the thread 1. Default value for depth1=80 (K142:1=80)
K142:2	Label of the error routine of the thread 2

Example:

```
void func120(void)      // error routine
{
    DIN.0 = 0;
    X0.0 = M7.0;
}
void func20(void)
{
    K142:1.0 = 120; // The error routine if the function 'func120'
                      // So, if an error occurs, func120() is called.
    PWR.0 = 1;
    IND.0;
    MVE.0 = 12000L;
    ...
}
```

In any case, an inactive thread stays inactive even if K142 is configured. A thread jumps to the error routine only when the first error appears. The following errors do not cause an interruption of the program flow. Errors can be reset by the RST command (refer to [§7.5.1](#)).

15.6.1 Simplified error routine

For applications having a **single thread**, the user can define an error routine named "errhandler". Caution: the errorhandler routine must not be used with several threads.

Example:

```
void errhandler(void) // error routine
{
    // the following code is executed when an error occurs
    ...
}
```

This routine works with the default value of the parameter K142. This lets you define an error routine in the easier way.

15.6.2 ERR command

From firmware 2.07A, it is possible to add an optional parameter to the command **ERR.<AXIS>[=<P1>]**. It allows the user to set application errors through the sequence for example. In that case, **<P1> must be a negative value**.

- If **<P1>** is a negative value, M64.<axis>=<P1>, the error message in M95.<axis> is 'EXTERNAL ERROR' and the message on ComET application is 'ERROR <P1>: unknown'.
- If the command is without parameter **<P1>**, the error message in M95 is 'EXTERNAL ERROR' and the message on ComET application is: 'ERROR 116': This error is generated by the ERR command'.

Command	Comment
ERR.<axis> [= <P1>]	Puts the controller in error mode (M64=116). If <PAR1> is a negative value, the error code in M64.<axis> is not 116 but the value of <PAR1> defined by the user.

15.7 Performance aspects

There are two factors, apart from the actual sequence code itself, that have an influence on how fast the sequence is executed:

- The time given to the compiled sequence scheduler by the firmware
- The kind of processor memory used for the execution stack

15.7.1 Allocation of time to the compiled sequence

It is possible for the user to optimize the given amount of time for the execution of the compiled sequence by setting the parameter **K200**.

K	Name	Comment
K200	Time allowed for compiled sequence	The value is a multiple of 3.75 ns. The default value is set to 4000, which corresponds to 15 µs. The minimum and maximum values are 1000 (4000 on version <3.16A>) and 8000, i.e. 3.75 µs and 30 µs.

The compiled sequence scheduler splits this time into the number of threads running in the sequence. It is possible to monitor the allotted time for the threads and the effective used time thanks to the monitorings **M201** and **M202**.

M	Name	Comment
M201	Sequence thread allotted time	Time quota for the threads. The <depth> corresponds to the thread number (1 or 2 – 0 not used).
M202	Time used by the thread	Effective time used by the threads. The <depth> corresponds to the thread number (1 or 2 – 0 not used).

It is the user's responsibility to set the compiled sequence time slot so that the MLTI do not overrun the cycle time. The ComET tool called 'Processor Load' should be used to check the risks.

15.7.2 Compiled sequence stack management

15.7.2.1 Setting

To increase the performance of the compiled sequences, it is possible to place the compiled sequence thread stacks in internal memory. This yields approximately a 2.5 fold increase in execution speed, depending on the content of the sequence. When in internal RAM, the stacks size is however significantly reduced. It contains 128 entries (32-bit integers), whereas when in external RAM it is 2048 slots. This means that sequences running in the fast configuration cannot have as many call depths as when running with the stacks in external RAM. The stack location is selected by setting the parameter **K415**.

K	Name	Value	Comment
K415	Enable fast stack for compiled sequences	0 1	Sequence stack in external memory (default case). Sequence stack in internal memory. Warning: the stack size is significantly reduced.

A modification of K415 becomes active after a save operation (SAV.<axis>=2) and a reboot of the controller. The monitorings **M417** and **M418** allow the user to monitor the compiled sequence threads stack sizes and usage.

M	Name	Comment
M417	Total stack size	The <depth> corresponds to the thread number 1 or 2 (0 is not used). This register is equal to 128 when stacks in internal RAM and 2048 when stacks in external RAM.
M418	Available free stack space	The <depth> corresponds to the thread number 1 or 2 (0 is not used).

15.7.2.2 Errors and restrictions

If a stack overflows, the **STACK OVERFLOW** error (M64=410) is raised. This error cannot be reset by a RST command and the board must be rebooted. It is due to too many nested calls or/and local variables. In that case, it is necessary to rework the sequence.

THIS PAGE IS INTENTIONALLY LEFT BLANK

Chapter E: Appendixes

16. Commands reference list

(*) Gantry action in level 2 (refer to [§12](#)):

- Action 1: The master can execute the command, the slave must generate an error
- Action 2: The master duplicates and executes the command, the slave must generate an error if it is for itself
- Action 3: The master and the slave can execute the command, no error is generated
- Action 4: The master and slave generate an error
- Action 5: The master ignores the command and slave must generate an error
- Action 6: The master duplicates and executes the command, the slave must ignore it
- Action 7: The master can execute the command, the slave must ignore it

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
ASG.<AXIS> = <P1>, <P2>, <P3>	All		Assign slots for getting the position and status sources for gantry algorithm			1
		1	Defines the offset in the TransnET DPRAM where the controller reads the LSL part of ML0 for gantry algorithm			
		2	Defines the offset in the TransnET DPRAM where the controller reads the MSL part of ML0 for gantry algorithm			
		3	Defines the offset in the TransnET DPRAM where the controller reads the axis status for gantry algorithm			
ASP.<AXIS> = <P1>, <P2>, <P3>, <P4>, <P5>, <P6>	All		Assign slots for getting the source positions of stage error mapping			1
		1	Defines the offset in the TransnET DPRAM where the controller reads the LSL of first source of stage mapping			
		2	Defines the offset in the TransnET DPRAM where the controller reads the MSL of first source of stage mapping			
		3	Defines the offset in the TransnET DPRAM where the controller reads the LSL of second source of stage mapping			
		4	Defines the offset in the TransnET DPRAM where the controller reads the MSL of second source of stage mapping			
		5	Defines the offset in the TransnET DPRAM where the controller reads the LSL of third source of stage mapping			
		6	Defines the offset in the TransnET DPRAM where the controller reads the MSL of third source of stage mapping			
ASR.<AXIS> = <P1>, <P2>[, <P3>]	All		Gives slot (offset in the TransnET cyclic data) where the controller can read cyclic data RTV. <P1> is the Mx450 to Mx465, where we want to put the read data, <P2> is the slot offset from where it reads the low part and <P3> the high part for 64bits data			3
		1	Defines the monitoring register that is read as a RTV register in the TransnET cyclic data: M450 to M465, or MF450 to MF465, or ML450 to ML457 or MD450 to MD457			
		2	Defines the offset slot address in the TransnET cyclic data for the read RTV register, lower part			
		3	Defines the offset slot address in the TransnET cyclic data for the read RTV register, higher part. Optional, only available for ML or MD			
ASW.<AXIS> = <P1>, <P2>[, <P3>]	All		Sets slot (offset in the TransnET cyclic data) where the controller writes cyclic data RTV			3

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
		1	Defines the register to be written by the controller (RTV) in the TransnET cyclic data. If the value=0, the register is not written anymore in the TransnET cyclic data			
		2	Defines the offset in the TransnET DPRAM where the controller writes the register (RTV) lower part			
		3	Defines the slot in the TransnET cyclic data where the controller writes the register, MSL part (RTV). Only available for 64bits registers			
AUT.<AXIS> = <P1>	All but ZxT		Calculates current loop parameters (KF80, KF81, K98), performs fine phase adjustment (K53 if K52=1) and finds motor connection (K56)			4
		1	Operation mask	Bit 0	Tunes proportional and integrator gain of the current loop KF80, KF81 and DC power voltage rate K98	
				Bit 1	Searches motor phase and sets K56	
				Bit 3	Sets K53 (fine phase adjustment) if K52=1	
AXI.<AXIS> = <P1>, <P2>	All		Changes the current axis number of a controller based on its serial number; <P1> is the serial number (SER or M73), <P2> is the new axis number of the first one. Example: AXI.1=1245,2 (change axis 1 in axis 2, its serial number is 1245)			4
		1	Serial number			
		2	New axis number			
BRK.<AXIS>	All		Stops the motor using programmed deceleration KL206 but does not stop the sequence			2
CEC.<AXIS> = <P1>	All		Clears Ethernet communication flags according to <P1>			1
		1	Mask on groups of flags	Bit 0	Clears Ethernet CRC counter (bit#8 to bit#11)	
				Bit 1	Clears Ethernet communication flags (bit#13 and bit#14)	
				Bit 2	Clears Ethernet warning (bit#20 to bit#29)	
CFGPGFIX.<AXIS> = <P1>, <P2>, <P3>, <P4>, <P5>, <P6>, <P7>, <P8>	All but ZxT		Fast Trigger: Configure Pulse Generator fixed time			3
		1	Trigger: Define Pulse Generator (1 or 2)			
		2	Trigger: Pulse Generator mask DOUT			
		3	Trigger: Pulse Generator mask FDOUT			
		4	Trigger: Pulse Generator pulse periodicity			
		5	Trigger: Pulse Generator delay			
		6	Trigger: Pulse Generator pulse width			
		7	Trigger: Pulse Generator interval pulse			
		8	Trigger: Pulse Generator pulse count			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
CFGPGMOD.<AXIS> = <P1>, <P2>, <P3>, <PF4>, <P5>, <PF6>, <P7>, <PF8>, <P9>, <P10>	All but ZxT		Fast Trigger: Configure Pulse Generator modulated time			3
		1	Trigger: Define Pulse Generator (1 or 2)			
		2	Trigger: Pulse Generator mask DOUT			
		3	Trigger: Pulse Generator mask FDOUT			
		4	Trigger: Pulse Generator pulse period modulation			
		5	Trigger: Pulse Generator pulse periodicity			
		6	Trigger: Pulse Generator delay modulation coefficient			
		7	Trigger: Pulse Generator delay			
		8	Trigger: Pulse Generator pulse width modulation coefficient			
		9	Trigger: Pulse Generator pulse width			
		10	Trigger: Pulse Generator interval pulse			
CFGTRIGEXTPOS.<AXIS> = <P1>, <P2>, <P3>, <P4>, <P5>[, <P6>[, <PF7>]]	All but ZxT		Fast Trigger: Configure Fast Trigger from external register selection			3
		1	Trigger: Define Pulse Generator (1 or 2)			
		2	Triggers external register reference type			
		3	Triggers external register reference index			
		4	Triggers external register reference depth			
		5	Triggers external register reference axis			
		6	Mean filter for trigger position			
		7	Trigger time compensation			
CH_BIT_REG32.<AXIS> = <P1>, <P2>, <P3>	All		Allows to write any bit (set bits and/or clear bits) <P2> with defined value <P3> of any 32 integer register <P1> without modifying the other bits the register			1
		1	Destination register			
		2	Mask of the bit that can be modified			
		3	Mask value of the bit			
CLRWAIT.<AXIS> = <P1>	All		Clear busy state due to a wait command			1
		1	Mask of the entry where the wait could be cleared (Not allowed for the sequence)	Bit 0	TransnET gate 0	
				Bit 1	TransnET gate 1	
				Bit 2	TransnET gate 2	
				Bit 3	TransnET gate 3	
				Bit 4	USB	

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
				Bit 8	TCP/IP	
CLX.<AXIS> = <P1>	All		Clears all user registers (X, XF, XL, XD register depending <P1>)			1
		1	Clears all user registers	0	Clears X, XF, XL, XD registers at all depths	
				Bit 0	Clears X registers at all depths	
				Bit 1	Clears XF registers at all depths	
				Bit 2	Clears XL registers at all depths	
				Bit 3	Clears XD registers at all depths	
END.<AXIS> [= <P1>]	All		Stops the sequence thread defined by <P1> or stops all threads if command without <P1>			1
		1	Thread to stop			
ERR.<AXIS> [= <P1>]	All		Puts the controller in error mode (M64=116). If <PAR1> is a negative value, the error code in M64.<axis> is not 116 but the value of <PAR1> defined by the user.			6
		1	Error number, which will appear in M64 (optional parameter)			
HLB.<AXIS>	All		Stops the sequence depending on K144 value and brakes the motor using programmed deceleration given by KL206			2
HLO.<AXIS>	All		Stops the sequence depending on K144 value and performs a power off			2
HLT.<AXIS>	All		Stops the sequence depending on K144 value and brakes the motor movement using an infinite deceleration			2
IND.<AXIS>	All		Starts a homing process			1
INI.<AXIS>	All but ZxT		Starts the phasing procedure (K90)			2
JMP.<AXIS> = <P1>, <P2>	All		Unconditional jump of a thread defined by <P2> to a label defined by <P1>			1
		1	Target label			
		2	Thread			
MVE.<AXIS> = <PL1>[, <PL2>[, <PL3>[, <PL4>]]]	All		Starts movement			2
		1	Target position			
		2	Speed for the motion (optional parameter)			
		3	Acceleration for the motion (optional parameter)			
		4	Jerk time for the motion (optional parameter)			
MVETILT.<AXIS> = <PL1>, <PL2>[, <PL3>]	ZxT		ZxT Starts movement on Rx, Ry			2
		1	Angle Rx			
		2	Angle Ry			
		3	Speed (optional parameter)			
NEW.<AXIS> = <P1>	All		Restores the default registers values (K, C), clears registers (X, L, E, P) and sequence in ram memory according to <P1>			1

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
		1	Register selection	0	Clears the P stage mapping variables and sequence, sets default values for K parameters and C common parameters in ram memory	
				1	Clears sequence in ram memory	
				2	Sets default values for K parameters and C common parameters in ram memory	
				3	Sets default values for K parameters in ram memory	
				4	Sets default values for C common parameters in ram memory	
				5	Clears X user variables in ram memory	
				6	Clears LD look-up table in ram memory	
				7	Clears sequence in ram memory	
				8	Clears EL triggers in ram memory	
				9	Clears P registers (mapping correction table) and stage error mapping variables in ram memory	
NEWFC.<AXIS> = <PF1>, <P2>[, <PF3>[,<P4>]]	All but ZxT	1	New Force Control			4
			Force reference value			
			Force reference time transition			
			Force range of the window for the force control			
			Duration of the window for Force Control			
OFFSETAUX.<AXIS> = <PL1>	All	1	Apply an offset on the auxiliary position with a specified value (first parameter of SETAUX command). Affect ML15			3
OFFSETSEC.<AXIS> = <PL1>	All	1	Apply an offset on the secondary position with a specified value (first parameter of SETSEC command). Affect ML13			3
POSCAPT.<AXIS> = <P1>, <P2>	All but ZxT	1	Position capture			1
			Enable disable			
			Mask			
PWR.<AXIS> = <P1>	All	1	Power on if <P1> = 1 or power off if <P1> = 0. The first power on does a phasing according to K90. A power off followed by a power on resets the errors			2
			Power On status			
READJUSTPOS.<AXIS> = <P1>	All but ZxT	1	Allow to readjust the position after an error. <P1> represents the depth to be readjusted (main, secondary, auxiliary).			3
			The depth to be readjusted for the analogue encoder position after an error			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
RES.<AXIS> = <P1>	All		Restores the saved registers in flash memory (K, C, X, L, E, P), sequence and axis number in ram memory according to <P1>			4
		1	Register selection	0	Restores sequence, LD look-up tables, K parameters and C common parameters, EL triggers, X user variables, P stage error mapping variables and axis number from flash to ram memory	
				1	Restores sequence and LD look-up tables from flash to ram memory	
				2	Restores K parameters and C common parameters, EL triggers, X user variables and axis number from flash to ram memory	
				3	Restores K parameters from flash to ram memory	
				4	Restores C common parameters and axis number from flash to ram memory	
				5	Restores X user variables from flash to ram memory	
				6	Restores LD look-up tables from flash to ram memory	
				7	Restores sequence from flash to ram memory	
				8	Restores EL triggers from flash to ram memory	
				9	Restores P registers (mapping correction table) and stage error mapping variables from flash to ram memory	
RESETFC.<AXIS> = <P1>[, <PL2>[, <PL3>[, <PL4>]]]	All but ZxT		Reset single axis Force Control			4
		1	Option			
		2	Absolute Position			
		3	Absolute Speed			
		4	Absolute Acceleration			
RMVE.<AXIS> = <PL1>[, <PL2>[, <PL3>[, <P4>]]]	All		Starts relative movement			2
		1	Relative target position			
		2	Speed for the movement (optional parameter)			
		3	Acceleration for the motion (optional parameter)			
		4	Jerk time for the motion (optional parameter)			
RMVETILT.<AXIS> = <PL1>, <PL2>[, <PL3>]	ZxT		ZxT Starts relative movement on Rx, Ry			2
		1	Angle Rx			
		2	Angle Ry			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
		3	Speed (optional parameter)			
RSD.<AXIS> = <P1>	All		Puts the controller in hardware reset if <P1> = 255			7
		1	Magic number			
RST.<AXIS> [= <P1>]	All		Resets the error flags of the controller (bit#10 of SD1). It also exits the 'torque off state' in stage protection mode with parameter <P1>=123			6
		1	Magic number that allow exiting stage protection 'torque off state' (must be 123)			
RTVDOUT.<AXIS> = <P1>	All		Assign slots for setting DOUT FDOUT XDOUT			1
		1	Defines the offset slot address in the TransnET cyclic data for the read RTV set output			
SAF.<AXIS> = <P1>, <PF2>, <PF3>, <PF4>, <PF5>, <PF6>	All		Set an advance filter			4
		1	Depth Advance Filter			
		2	Advanced filter coefficient for yk-1			
		3	Advanced filter coefficient for yk-2			
		4	Advanced filter coefficient for xk			
		5	Advanced filter coefficient for xk-1			
		6	Advanced filter coefficient for xk-2			
SAFRX.<AXIS> = <P1>, <PF2>, <PF3>, <PF4>, <PF5>, <PF6>	ZxT		ZxT Rx set an advance filter			4
		1	ZxT Rx Depth Advance Filter			
		2	ZxT Rx Advanced filter coefficient for yk-1			
		3	ZxT Rx Advanced filter coefficient for yk-2			
		4	ZxT Rx Advanced filter coefficient for xk			
		5	ZxT Rx Advanced filter coefficient for xk-1			
		6	ZxT Rx Advanced filter coefficient for xk-2			
SAFRY.<AXIS> = <P1>, <PF2>, <PF3>, <PF4>, <PF5>, <PF6>	ZxT		ZxT Ry set an advance filter			4
		1	ZxT Ry Depth Advance Filter			
		2	ZxT Ry Advanced filter coefficient for yk-1			
		3	ZxT Ry Advanced filter coefficient for yk-2			
		4	ZxT Ry Advanced filter coefficient for xk			
		5	ZxT Ry Advanced filter coefficient for xk-1			
		6	ZxT Ry Advanced filter coefficient for xk-2			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
SAV.<AXIS> = <P1>	All		Saves the current registers (K, C, X, L, E, P), sequence and axis number in flash memory in function of <P1>			7
		1	Register selection	0	Saves sequence, LD look-up tables, K parameters and C common parameters, EL triggers, X user variables, P stage error mapping variables and axis number in flash memory	
				1	Saves sequence and LD look-up tables in flash memory	
				2	Saves K parameters and C common parameters, EL triggers, X user variables and axis number in flash memory	
				3	Saves K parameters, EL triggers in flash memory	
				4	Saves C common parameters and axis number in flash memory	
				5	Saves X user variables in flash memory	
				6	Saves LD look-up tables in flash memory	
				7	Saves sequence in flash memory	
				8	Saves K registers (parameter) and EL registers (trigger) in flash memory	
				9	Saves P registers (mapping correction table) and stage error mapping variables in flash memory	
SCI.<AXIS> = <P1>	All		Command for calculation of I2 sum. If <P1>=1, the acquisition starts. If <P1>=0, the acquisition stops. The result is given in ML181 for the I2 sum and M180 for number of PLTI during the measure			1
		1	SCI acquisition start			
SET.<AXIS> = <PL1>	All		Sets the current user position to the specified value (first parameter of SET command). Affects user units only (ML6 + ML7)			2
		1	Position			
SETFC.<AXIS> = <PF1>, <P2>, <PL3>[, <PF4>[, <P5>[, <PL6>[, <PL7>]]]]]	All but ZxT		Set single axis Force Control			4
		1	Force reference value			
		2	Fctrl mode			
		3	Absolute Position			
		4	Force range of the window for the force control			
		5	Duration of the window for Force Control			
		6	Absolute Speed			
		7	Absolute Acceleration			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
SETFCI.<AXIS> = <PF1>, <P2>, <PL3>, <P4>[, <PF5>[, <P6>]]	All but ZxT		Set single axis Force Control without movement			4
		1	Force reference value			
		2	Fctrl mode			
		3	Direction			
		4	Smoothing			
		5	Force range of the window for the force control			
		6	Duration of the window for Force Control			
SETTILT.<AXIS> = <PL1>, <PL2>	ZxT		ZxT Adds an offset on upi units for Rx, Ry			2
		1	Angle Rx			
		2	Angle Ry			
SLS.<AXIS>	All		Searches limit stroke according to K145 (search limit stroke mode) and returns the limit position in ML36 and ML37. KL47 is taken into account by SLS command but not in ML36 and ML37 setting. Not available in ITP mode			4
SMP.<AXIS> = <P1>, <PL2>[, <PL3>[, <PL4>[, <P5>]]]	All		Set movement parameter for STA and STI commands. <P1> is the depth, <PL2> position, <PL3> speed, <PL4> acceleration, <P5> jerk time			2
		1	Depth			
		2	Target position			
		3	Speed for the movement (optional parameter)			
		4	Acceleration for the motion (optional parameter)			
		5	Jerk time for the motion (optional parameter)			
SPG.<AXIS> = <P1>, <P2>	All but ZxT		Fast Trigger: Starts or Stop pulse generator. <P1> pulse generator number (1 or 2). <P2> Start or Stop (1 or 0)			3
		1	Pulse Generator number			
		2	Start/stop			
SSR.<AXIS> = <P1>, <P2>[, <P3>, <P4>[, <P5>, <P6>[, <P7>, <P8>[, <P9>, <P10>[, <P11>, <P12>]]]]]	All		Set Several Registres 1 up to 6			3
		1	1st register and depth definition			
		2	1st register value			
		3	2nd register and depth definition (optional parameter)			
		4	2nd register value (optional parameter)			
		5	3th register and depth definition (optional parameter)			
		6	3th register value (optional parameter)			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
STA.<AXIS> = <P1>, <P2>	All	7	4th register and depth definition (optional parameter)			2
		8	4th register (optional parameter)			
		9	5th register and depth definition (optional parameter)			
		10	5th register (optional parameter)			
		11	6th register and depth definition (optional parameter)			
		12	6th register (optional parameter)			
STA.<AXIS> = <P1>, <P2>	All		Starts a movement by using the movement parameters of sub-index (depth 1 to 7) specified by <P1>, and <P2> is a mask of movement parameter to be loaded (0 means all movements parameters are taken into account)			2
		1	Buffer subindex			
		2	Buffer mask	Bit 0	Gets target position (TARGET=KL210) from the specified index. Also possible during a trapezoidal movement if MMC=1	
				Bit 1	Gets profile velocity (SPD=KL211) from the specified index. Also possible during a trapezoidal movement if MMC=1	
				Bit 2	Gets profile acceleration (ACC=KL212) from the specified index. Also possible during a trapezoidal movement if MMC=1	
				Bit 3	Gets jerk time (JRT=K213) from the specified index	
				Bit 6	Gets profile type (MMD=K202) from the specified index	
				Bit 7	Gets user look-up table number (LTN=K203) and the look-up table number selection mode (K207) from the specified index	
				Bit 8	Gets look-up table time (LTI=K204) from the specified index	
				Bit 9	Gets user amplitude for look-up table (KL208) in mode K207=1	
				Bit 10	Gets rotary movement type selection (K209) only when the motor is not moving (bit#4 of SD1 at 0)	
				Bit 11	Gets the value of K230 from the specified index	
				Bit 12	Gets the value of K229 from the specified index	
STE_ABS.<AXIS> = <P1>	All		Performs a position step (infinite acceleration) to the specified (absolute) position			4
		1	Position			
STE_ADD.<AXIS> = <P1>	All		Performs a position step (infinite acceleration) of the specified size in the positive direction			4
		1	Step size			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
STE_SUB.<AXIS> = <P1>	All		Performs a position step (infinite acceleration) of the specified size in the negative direction			4
		1	Step size			
STETILT_ABS.<AXIS> = <PL1>, <PL2>	ZxT		ZxT Modifies the reference position to the specified (absolute) position			2
		1	Angle Rx			
		2	Angle Ry			
STETILT_ADD.<AXIS> = <PL1>, <PL2>	ZxT		ZxT Adds a step to the reference position			2
		1	Angle Rx			
		2	Angle Ry			
STETILT_SUB.<AXIS> = <PL1>, <PL2>	ZxT		ZxT subtracts a step to the reference position			2
		1	Angle Rx			
		2	Angle Ry			
STI.<AXIS> = <P1>, <P2>	All		Synchronous start of a movement on an input by using the movement parameters of the sub-indexes specified by <P1>, and <P2> is a mask to enable movement variable to load (0 means all movements parameters are taken into account)			2
		1	Buffer subindex			
		2	Buffer mask	Bit 0	Gets target position (TARGET=KL210) from the specified index. Also possible during a trapezoidal movement if MMC=1	
				Bit 1	Gets profile velocity (SPD=KL211) from the specified index. Also possible during a trapezoidal movement if MMC=1	
				Bit 2	Gets profile acceleration (ACC=KL212) from the specified index. Also possible during a trapezoidal movement if MMC=1	
				Bit 3	Gets jerk time (JRT=K213) from the specified index	
				Bit 6	Gets profile type (MMD=K202) from the specified index	
				Bit 7	Gets user look-up table number (LTN=K203) and the look-up table number selection mode (K207) from the specified index	
				Bit 8	Gets look-up table time (LTI=K204) from the specified index	
				Bit 9	Gets user amplitude for look-up table (KL208) in mode K207=1	
				Bit 10	Gets rotary movement type selection (K209) only when the motor is not moving (bit#4 of SD1 at 0)	

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
				Bit 11	Gets the value of K230 from the specified index	
				Bit 12	Gets the value of K229 from the specified index	
STP.<AXIS>	All		Stops the current motor movement with an infinite deceleration but does not stop the sequence			2
TRE.<AXIS> = <P1>, <P2>[, <P3>]	All but ZxT		Fast Trigger: Command to enable/disable trigger 1D function when axis is not moving: <P1>=table starting line, <P2>=table line number, ([<P3>]= Enable missed event detection). If <P1>=-1, trigger is disabled			3
		1	Trigger starting line			
		2	Trigger number of line			
		3	Trigger enable missed_event			
TRM.<AXIS> = <P1>, <P2>[, <P3>]	All but ZxT		Fast Trigger: Command to enable/disable trigger 1D function when axis is moving: <P1>=table starting line, <P2>=table line number, ([<P3>]= Enable missed event detection). If <P1>=-1, trigger is disabled			3
		1	Trigger starting line			
		2	Trigger number of line			
		3	Trigger enable missed_event			
TRM2D.<AXIS> = <P1>, <P2>[, <P3>]	All but ZxT		Fast Trigger: Command to enable/disable trigger 2D function (axis is moving or not): <P1>=table starting line, <P2>=table line number, ([<P3>]= Enable missed event detection). If <P1>=-1, trigger is disabled			1
		1	Trigger starting line			
		2	Trigger number of line			
		3	Trigger enable missed_event			
TRR.<AXIS> = <P1>[, <P2>]	All but ZxT		Fast Trigger: Command to initialize the DOUT/FDOUT used by the trigger. <P1> is the state for the DOUT and <P2> is the state for FDOUT			3
		1	Initial state for the DOUT			
		2	Initial state for the FDOUT (optional parameter)			
TRS.<AXIS> = <P1>, <P2>[, <P3>]	All but ZxT		Fast Trigger: Command to enable/disable trigger status function when axis is moving: <P1>=table starting line, <P2>=table line number, [<P3>]=max number missed element. If <P1>=-1, trigger status is disabled			3
		1	Trigger starting line			
		2	Trigger number of line			
		3	Max number missed element (optional parameter)			
UST.<AXIS> = <P1>, <PL2>, <PF3>	All but ZxT		Encoder Stretch command			2
		1	Enable or Disable			
		2	Null correction position (Offset)			
		3	Gradient factor			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
WBC.<AXIS> = <P1>, <P2>	All		Waits for the specified bits (which have their mask value included in <P2>) of the specified register (<P1>) to be all cleared. Sets bit in M101 as long as the condition is not true and then clears it			1
		1	Value or register to test			
		2	Bit mask			
WBS.<AXIS> = <P1>, <P2>	All		Waits for the specified bits (which have their mask value included in <P2>) of the specified register (<P1>) to be all set. Sets bit in M101 as long as the condition is not true and then clears it			1
		1	Value or register to test			
		2	Bit mask			
WPG.<AXIS> = <P1>, <P2>	All		Waits until <P1> becomes greater than <P2>. Sets bit in M101 as long as the condition is not true and then clears it			1
		1	Value or register to test			
		2	Reference value or register			
WPL.<AXIS> = <P1>, <P2>	All		Waits until <P1> becomes lower than <P2>. Sets bit in M101 as long as the condition is not true and then clears it			1
		1	Value or register to test			
		2	Reference value or register			
WSG.<AXIS> = <P1>, <P2>	All		Waits until <P1> becomes greater than <P2> (signed). Sets bit in M101 as long as the condition is not true and then clears it. The test is signed			1
		1	Value or register to test			
		2	Reference value or register			
WSL.<AXIS> = <P1>, <P2>	All		Waits until <P1> becomes lower than <P2> (signed). Sets bit in M101 as long as the condition is not true and then clears it. The test is signed			1
		1	Value or register to test			
		2	Reference value or register			
WTD.<AXIS> = <P1>, <P2>	All		WaiT all Data are available for the Continuous Traces			3
		1	First index for the range of data			
		2	Last index for the range of data			
WTF.<AXIS>	All but ZxT		Wait in Force Control that the current error is around the current reference MF401			4
WTM.<AXIS>	All		Waits for the end of the movement. Sets bit in M101 during all the movements and clears it at the end of the movement			1
WTP.<AXIS> = <PL1>	All		Waits for the specified position. Sets bit in M101 as long as it does not reach this position and then clears it			1
		1	Position			
WTS.<AXIS>	All		Wait during a motion, when the real speed (usi) is in a window of speed (+/- KL181) during a time (K180 in MLTI) around the speed target KL211			1
WTT.<AXIS> = <P1>	All		Waits for the specified amount of time (MLTI). Sets bit in M101 as long as it waits and then clears it			1

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
		1	Time			
WTW.<AXIS> [= <P1>]	All		Waits for the end of the movement, when the real position (upi) is in a window of position (+/- K39) during a time (K38 in MLT1) around the target			1
		1	Defines the type of wait windows (optional parameter)	0	Waits for the bit in window (bit#5 of SD1) to be at 1. If the motor is not moving (bit#4 of SD1 at 0) when this command is executed, the controller does not wait	
				1	Waits for the bit in window (bit#5 of SD1) to be at 1. It does not take into account if the motor is moving or not	
				2	Restarts a new test by clearing the bit in window (bit#5 of SD1). It does not take into account if the theoretical trajectory is finished or not	
WTWZXT.<AXIS> [= <P1>]	ZxT		Wait for the edge of the chuck to be within a position window around the target (+/- K482) for a given time (K481).			1
		1	Defines the type of wait windows (optional parameter)	0	Waits for the bit in window (bit#5 of SD1) to be at 1. If the motor is not moving (bit#4 of SD1 at 0) when this command is executed, the controller does not wait	
				1	Waits for the bit in window (bit#5 of SD1) to be at 1. It does not take into account if the motor is moving or not	
				2	Restarts a new test by clearing the bit in window (bit#5 of SD1). It does not take into account if the theoretical trajectory is finished or not	
<REG>.<AXIS> = <P2>	All		Affects the value of <P2> to <REG>. This is available with all kinds of registers excepted monitoring register (M)			3
		1	Destination register			
		2	Source value or register			
<REG>.<AXIS> += <P2>	All		Adds <P2> to <REG> and puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 += M7.1, but not X1.1 += ML7.1			3
		1	Destination register			
		2	Value or register to add			
<REG>.<AXIS> &= <P2>	All		Logical AND between each bit of <REG> and <P2>, result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 &= K30.1			3
		1	Destination register			
		2	Value or register to AND with			
<REG>.<AXIS> &~= <P2>	All		Logical NOT AND between each bit of <REG> and <P2>, result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 &~= K30.1			3
		1	Destination register			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
		2	Value or register to NOT AND with			
<P1>.<AXIS> = <P2>	All		Executes the operation: <P1> = conversion <P2>. This is available with all kinds of registers (<P1> cannot be a monitoring register). The syntax is for example XF2.1 = X3.1, or X3.1 = XF2.1			3
		1	Destination register			
		2	Source register			
<REG>.<AXIS> /= <P2>	All		Divides <REG> by <P2> and puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 /= K30.1			3
		1	Destination register			
		2	Value or register to divide by			
<REG>.<AXIS> %= <P2>	All		Modulo between <REG> and <P2> and puts the result in <REG>			3
		1	Destination register			
		2	Value or register to modulo with			
<REG>.<AXIS> *= <P2>	All		Multiplies <REG> by <P2> and puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 *= K30.1			3
		1	Destination register			
		2	Value or register to multiply with			
<REG>.<AXIS> ~= <P2>	All		Inverts <P2> and puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 ~= M7.1			3
		1	Destination register			
		2	Source value or register			
<REG>.<AXIS> != <P2>	All		Logical OR between each bit of <REG> and <P2>, result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 != K30.1			3
		1	Destination register			
		2	Value or register to OR with			
<REG>.<AXIS> ~= <P2>	All		Logical NOT OR between each bit of <REG> and <P2>, result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 ~= K30.1			3
		1	Destination register			
		2	Value or register to NOT OR with			
<REG>.<AXIS> <<= <P2>	All		Shifts <REG> of <P2> bit to the left and puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 <<= M7.1			3
		1	Destination register			
		2	Value or register to shift by			

Syntax	Product	Param. number	Command or parameter description	Values or bit#	Value description	Gantry action (*)
<REG>.<AXIS> >>= <P2>	All		Shifts <REG> of <P2> bit to the right and puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 >>= M7.1			3
		1	Destination register			
		2	Value or register to shift by			
<REG>.<AXIS> -= <P2>	All		Subtracts <P2> from <REG> and puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 -= M7.1, but not X1.1 -= ML7.1			3
		1	Destination register			
		2	Value or register to subtract			
<REG>.<AXIS> ^= <P2>	All		Logical XOR between each bit of <REG> and <P2>, puts the result in <REG>. The left operand can be any of K, X or C registers, the right operand can be any immediate or indirect value of the same type. Example: X1.1 = K30.1			3
		1	Destination register			
		2	Value or register to XOR with			
<REG>.<AXIS> ^~= <P2>	All		Logical NOT XOR between each bit of <REG> and <P2> and puts the result in <REG>			3
		1	Destination register			
		2	Value or register to NOT XOR with			
ZTE.<AXIS> = <P1>	All		Trace: With K120=0, starts acquisition if <P1> = 1 (or 2). With K120=1, if <P1>=1 the axis waits the start acquisition signal, if <P1>=2 the axis becomes the master and sends start acquisition signal. Stops acquisition if <P1>= 0			3
		1	Enable state			

Remark: The following commands can be sent as an urgent command: ASR, ASW, BRK, CLRWAIT, END, ERR, HLB, HLO, HLT, PWR, STP, <REG>.<AXIS>=<P2> and ZTE. Refer to [§2.5.1](#) for more information about the urgent commands.

17. Parameters K

(*) Gantry action in level 2 (refer to [§12.](#)):

- Action 1: The master can execute the command, the slave must generate an error
- Action 2: The master duplicates and executes the command, the slave must generate an error if it is for itself
- Action 3: The master and the slave can execute the command, no error is generated
- Action 4: The master and slave generate an error
- Action 5: The master ignores the command and slave must generate an error
- Action 6: The master duplicates and executes the command, the slave must ignore it
- Action 7: The master can execute the command, the slave must ignore it

(**) Same value in gantry level 1 or 2 (refer to [§12.](#)):

- Yes: the value **must be identical** in gantry configuration 1 (C245=1) and 2 (C245=2)
- No: the value **can be different** in gantry configuration 1 (C245=1) and 2 (C245=2)

FLOAT_MAX = 3.4028234663852886E+38 and FLOAT_MIN = -3.4028234663852886E+38

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
KF1	KPP	All		Position loop proportional gain	100	0	INT32_MAX	4	No
KF2	KDP	All		Position loop speed feedback gain	20	0	INT32_MAX	4	No
KF4	KIP	All		Position loop integrator gain	0	0	INT32_MAX	4	No
KF5	KAWP	All		Position loop anti-windup gain	10	0	INT32_MAX	4	No
KF6		All		Position loop integrator limitation	31000	0	31000	4	No
K10		All		Gain scheduling activation	0	0	31	4	No
			Bit 0	Gain scheduling KF2 depending current MF31; 0: not active; 1: active					
			Bit 1	Gain scheduling KF4 depending speed M11; 0: not active; 1: active					
			Bit 2	Advanced gain scheduling on KF1 KF2 and KF4; 0: not active; 1: active					
K11		All		Speed smooth filter for TTL encoder	0	0	256	3	No
KF12		All		Depth 0: filter on current level (MF31) for KF2 gain scheduling, depth 1: filter on speed level (M11) for KF4 gain scheduling	0	0	1	4	No
KF13		All		Current level (MF31) from which KF2 gain scheduling is modified (depth 0 and 1)	31000	0	31000	4	No
KF14		All		Speed loop proportional gain for Stage Protection	100	0	INT32_MAX	4	No
KF15		All		Speed loop integrator gain for Stage Protection	0	0	INT32_MAX	4	No
K19		All		Speed level for I2t motor, gain scheduling KF4, Friction feedforward and Force Control. (depth 0 and 1: gain scheduling KF4), (depth 2: I2t motor for stall mode), (depth 3: Friction feedforward), (depth4: threshold switching in force control)	INT32_MAX	1	INT32_MAX	4	No
KF20	KVFFP	All		Position loop speed feedforward gain	0	0	INT32_MAX	4	No
KF21	KAFFP	All		Position loop acceleration feedforward gain	0	0	INT32_MAX	4	No
KF23		48, 300, 400		Back EMF compensation	0	0	INT32_MAX	4	No

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
K24		All		Filter of the theoretical acceleration for the acceleration feedforward (0: disabled, 1: enabled)	0	0	1	2	Yes
KL27		All		Maximum position range limit for rotary movement	0	0	INT48_MAX	4	Yes
K30		ZxT		Absolute tracking error limit. When tracking error greater than K30:0 (M2), the controller raises error M64=23.	10000000	0	1073741823	2	Yes
K30		All but ZxT		Absolute tracking error limit. When tracking error greater than K30:0 (M2), the controller raises error M64=23. In Gantry mode (C245 != 0), the controller raises error M64=119. In Dual Feedback (K76:0 or K76:1 != 0), the controller raises error M64=79.	10000000	0	1073741823	2	Yes
K31		All		Absolute speed limit. When the speed is greater than K31, the controller generates an overspeed error (M64=24)	100000000	0	INT32_MAX	2	Yes
K32		All		Limit switch and home switch inversion	0	0	62	4	Yes
			Bit 1	Inverts home switch from DIN2					
			Bit 2	Inverts limit switches or home switch from the encoder					
			Bit 3	Inverts limit switches from digital inputs					
			Bit 4	Swaps limit switches or home switch from the encoder					
			Bit 5	Swaps limit switches from digital inputs					
K33		All		Enables input mode	0	0	31	4	Yes
			Bit 1	Enables the limit switches					
			Bit 2	Enables the dynamic braking controlled by the transistors in case of Power OFF					
			Bit 3	Enables the dynamic braking controlled by the transistors in case of error					
KL34	MINSL	All		Minimum software position limit	0	INT48_MIN	INT48_MAX	2	Yes
KL35	MAXSL	All		Maximum software position limit	0	INT48_MIN	INT48_MAX	2	Yes
K36	MODESL	All		Enables the software position limits (KL34, KL35)	0	0	7	2	Yes
			Bit 0	Use KL34 and KL35 as target limits the motor can reach (after MVE, RMVE, STE, STE+, STE-, STA and STI). If the motor reaches these limits, no error will be generated. Used with K61=1. Not available in interpolation mode (ITP)					
			Bit 1	Use KL34 and KL35 as limits on the current position. If the motor reaches these limits, it generates an error (M64=65). These limits are tested every MLTI but only if a homing has been previously done. Available for all values of K61 and ITP					
			Bit 2	Use KL34 and KL35 as target limits to generate an error (M64=66) when the movement starts (after MVE, RMVE, STE, STE+, STE-, STA and STI). Used with K61=1. Not available in interpolation mode (ITP)					
K37		All		Delta position threshold of the status bit IsMoving	0	0	1073741823	2	Yes
K38	TIMEW	All		Duration of the window (used with WTW command)	0	0	393210	2	Yes
K39	POSW	All		Position range of the window (used with WTW command)	0	0	1073741823	2	Yes
K40	HMODE	ZxT		Homing mode	22	0	48	2	Yes
			22	Immediate homing at the current position (It is used to take K53 at a known position into account)					
			40	Homing on index for ZxT					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			42	Homing on mechanical end stop for ZxT					
			48	Tip-tilt immediate homing based on index positions					
			Bit 0	Homing with a negative movement					
K40	HMODE	All but ZxT		Homing mode	8	0	47	2	Yes
			22	Immediate homing at the current position (It is used to take K53 at a known position into account)					
			40	Homing on index for ZxT					
			42	Homing on mechanical end stop for ZxT					
			48	Tip-tilt immediate homing based on index positions					
			Bit 0	Homing with a negative movement					
KL41	HSPD	All		Homing speed	2000000	1	1.0995e+12	2	Yes
KL42	HACC	All		Homing acceleration	1000000	256	1.0995e+12	2	Yes
KL43		All		Homing tracking limit for mechanical end stop detection	10000000	0	INT32_MAX	2	Yes
KF44		All		Homing current limit for mechanical end stop detection	4096	0	INT16_MAX	2	Yes
KL45	HOFFS	ZxT		Homing offset on absolute position	0	-268435456	268435455	3	No
KL45	HOFFS	All but ZxT		Homing offset on absolute position	0	INT48_MIN	INT48_MAX	3	No
KL46		All		Homing stroke for mode K40 = 20 to 21, 24 to 27, 36 to 39, 44 to 45	0	0	INT48_MAX	2	Yes
KL47		All		Movement to go out of a limit switch or mechanical end stop at the end of the homing	0	0	INT48_MAX	2	Yes
KL48		All		Movement to go out of an index or home switch if the motor is on the top of it when starting the homing	100	0	INT48_MAX	2	Yes
KL49		All		Homing delta offset absolute position in Gantry mode	0	INT48_MIN	INT48_MAX	4	No
K52		All		Enables fine phase adjustment (takes K53 into account) after homing	0	0	1	4	Yes
K53		All		Motor commutation phase adjustment after homing (is taken into account only if K52=1)	0	0	2048	4	No
K54	PPOLE	All		Number of pole pairs of the motor (=1 for a linear motor)	1	1	INT32_MAX	4	Yes
K55		All		Encoder position increment factor: number of rdpi per revolution for rotary motor or number of dpi per magnetic period for linear motor	0	0	INT32_MAX	4	Yes
KL55		All		Encoder position increment factor: number of rdpi per revolution for rotary motor or number of dpi per magnetic period for linear motor	0	0	INT48_MAX	4	Yes
K56		All		Motor commutation phase inversion enabled	0	0	1	4	No
K58		All		Mode of hardware limit switch	0	0	3	4	Yes
			0	No limit switch					
			1	Limit switch on DIN9 and DIN10					
			2	Limit switch L1/L2					
			3	Limit switch L/H					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
KF59		All		Theoretical software current limit for stepper, when no movement is required	20000	0	31000	4	No
KF60	IPEAK	All		Theoretical software current limit (regulator output)	20000	0	31000	2	Yes
K61		All		Position reference mode	1	1	36	2	Yes
			1	Standard position profile mode					
			3	Pseudo-speed reference defined by K220, K221, K222, K223 and KF224 depth 0.					
			4	Position reference defined by K220, K221, K223 and KF224 depth 0.					
			36	Position reference defined by K220, K221, K223 and KF224 depth 0. After a power on, it takes into account the actual motor position as reference					
KF61		ZxT		ZxT Ry position loop proportional gain	100	0	INT32_MAX	4	No
K62		All		Configuration of the status bit IsMoving	0	0	2	2	Yes
			0	The isMoving bit is managed by the trajectory generator (standard mode)					
			1	The isMoving bit is function of the delta of measured positions between 2 MLTI and the threshold level.					
			2	The isMoving bit is function of the delta of theoretical positions between 2 MLTI and the threshold level.					
KF62		ZxT		ZxT Ry position loop speed feedback gain	20	0	INT32_MAX	4	No
K63		All		Current reference mode	0	0	2	4	Yes
			0	Current reference comes from position regulator					
			1	Current reference is defined by registers K220, K221, K222, K223 and KF224 depth 0.					
KF64		ZxT		ZxT Ry position loop integrator gain	0	0	INT32_MAX	4	No
K66		All		ComET's display mode	1	1	32	4	No
			1	Displays normal informations					
			2	Displays temperature of the controller					
			4	Displays analog encoder amplitude					
			32	Displays DC power voltage (Vpower) [V]					
K68		All		Encoder reading way inversion (depth 0: main, depth 1: secondary, depth 2: auxiliary)	0	0	1	4	No
K69	ERRORAUX	All but ZxT		Allow error on auxiliary analogue encoder	1	0	1	3	No
K70		All		Analog encoder sine offset (depth 0: main, depth 1: secondary, depth 2: auxiliary)	0	-409	409	4	No
K71		All		Analog encoder cosine offset (depth 0: main, depth 1: secondary, depth 2: auxiliary)	0	-409	409	4	No
KF72		All		Analog encoder sine factor (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1	0.75	1	4	No
KF73		All		Analog encoder cosine factor (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1	0.75	1	4	No
K75		All		Distance between two indexes for multi-reference marks encoder	0	0	INT32_MAX	4	Yes
K76		All but ZxT		Dual Encoder Feedback mode. Depth0: mode 0 Permanent Dual Encoder Feedback, Depth1: mode 1 Intermittent Dual Encoder Feedback, Depth2: mode2 Combined Dual Encoder Feedback	0	0	23	4	Yes

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			0	No dual encoder feedback configuration					
			1	1Vpp as payload encoder; TTL as motor encoder (permanent only)					
			2	1Vpp as payload encoder; Absolute EnDat2.2 as motor encoder (permanent only)					
			3	Absolute EnDat2.2 as payload encoder; 1Vpp as motor encoder (permanent only)					
			4	1Vpp as payload encoder; HSEI as motor encoder (permanent only)					
			5	Absolute EnDat2.2 as payload encoder; HSEI as motor encoder (permanent only)					
			6	HSEI as payload encoder; 1Vpp as motor encoder (permanent only)					
			7	HSEI as payload encoder; TTL as motor encoder (permanent only)					
			8	HSEI as payload encoder; EnDat2.1 as motor encoder (permanent only)					
			9	HSEI as payload encoder; Absolute EnDat2.2 as motor encoder (permanent only)					
K77		All		Encoder interpolation shift value (depth 0: main, depth 1: secondary, depth 2: auxiliary)	0	-4	16	4	Yes
K78		All		Position Controller Mode	0	0	99	4	Yes
			0	State regulator					
K79		All		Encoder type selection (depth 0: main, depth 1: secondary, depth 2: auxiliary)	98	0	99	4	Yes
			0	Analog sine/cosine encoder 1Vpp					
			2	TTL encoder high frequency (10MHz).					
			3	EnDat 2.2 encoder without sine and cosine					
			4	EnDat 2.1 encoder or EnDat 2.2 encoder with sine and cosine					
			5	HSEI Analog sine/cosine encoder 1Vpp (6MHz)					
			20	Absolute from any register					
			98	No encoder & not possible to do PWR ON					
			99	No encoder					
KF80	KPC	48, 300, 400		Current loop proportional gain	500	0	INT32_MAX	2	Yes
KF80	KPC	All VHP		Current loop proportional gain	500	0	UINT16_MAX	2	Yes
KF81	KIC	48, 300, 400		Current loop integrator gain	0	0	INT32_MAX	2	Yes
KF81	KIC	All VHP		Current loop integrator gain	0	0	UINT16_MAX	2	Yes
KF83		All		Current loop software overcurrent limit	16000	0	32000	2	Yes
KF84		All		I2 motor rms current limit. Depth 0: motor is moving, depth 1: motor in stall. The speed limit for the changing mode is defined by K19:2	510	0	8192	2	Yes
KF85		All		I2t motor integration limit	20966400	0	FLOAT_MAX	2	Yes
K89		400		Motor phase number and PWM frequency (default value for AccurET Modular 600 is 31)	10	10	38	4	Yes
K89		48, 300		Motor phase number and PWM frequency	10	10	38	4	Yes

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
K89		VHP48 & 100		Motor phase number	10	10	38	4	Yes
K90	All but ZxT			Phasing mode. Depth 1: phasing mode for command AUT.<axis>=8	2	0	8	4	Yes
			0	No phasing (with 1-ph. motor or absolute EnDat encoder). The value stored in K53 is used if K52 = 1 except for absolute EnDat encoder.					
			1	Phasing with current pulses (with 3-phase ironcore motors only). The value stored in K53 is used if K52=1					
			2	Phasing by sending constant current to the motor (ironcore and ironless motors). The value stored in K53 is used if K52=1					
			3	Phasing with digital Hall sensors until the index is found then commutation by position encoder. The value stored in K53 is used if K52=1					
			4	Phasing with digital Hall sensors until the first edge of signal then commutation by position encoder. When index is found, the value stored in K53 is used if K52=1					
			5	Phasing and commutation with digital Hall sensors only. The value stored in K53 is used if K52=1					
			6	Small movements phasing. The value stored in K53 is used if K52=1					
			7	Small movements phasing. Repeated automatically 3 times when phasing process fails. The value stored in K53 is used if K52=1					
KF91		All		Pulse amplitude	12000	0	28000	4	Yes
KF92		All		Maximum current level	12000	0	28000	4	Yes
K93		All		Pointers final position in the look-up table of the current loop	1024	0	2047	4	Yes
K94		All		Maximum time allowed before the motor is in a stable balance point	5000	0	25000	4	Yes
K97		All		Pointers initial position adjustment in the look-up table of the current loop	512	0	2047	4	Yes
K98		All		Phasing power voltage rate in percent with K90=1. This is useful for low inductance motor	100	25	100	4	Yes
K101		All		Phasing time process (with K90=6 and 7)	1000	0	25000	4	Yes
KF104		All		Advanced filter coefficient for y[k-1]	0	-1000	1000	4	No
KF105		All		Advanced filter coefficient for y[k-2]	0	-1000	1000	4	No
KF106		All		Advanced filter coefficient for x[k]	1	-1000	1000	4	No
KF107		All		Advanced filter coefficient for x[k-1]	0	-1000	1000	4	No
KF108		All		Advanced filter coefficient for x[k-2]	0	-1000	1000	4	No
KF114		ZxT		ZxT Ry advanced filter coefficient for y[k-1]	0	-1000	1000	4	No
KF115		ZxT		ZxT Ry advanced filter coefficient for y[k-2]	0	-1000	1000	4	No
KF116		ZxT		ZxT Ry advanced filter coefficient for x[k]	1	-1000	1000	4	No
KF117		ZxT		ZxT Ry advanced filter coefficient for x[k-1]	0	-1000	1000	4	No
KF118		ZxT		ZxT Ry advanced filter coefficient for x[k-2]	0	-1000	1000	4	No
K119	All			Enables or disables current compensation with look-up table (cogging compensation)	0	0	2	4	Yes
			0	Cogging compensation disabled					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			1	Enable cogging compensation for linear axis					
			2	Enable cogging compensation for rotary axis					
K120		All		Trace: Enables trace synchronization mode	0	0	1	3	No
K121		All		Trace: Register selection type (depth 0 for trace 0; depth 1 for trace 1,...)	3	0	INT32_MAX	3	No
			1	User variable register (X)					
			2	Parameter register (K)					
			3	Monitoring register (M)					
			7	Address register (A)					
			8	LKT register (L)					
			13	Common parameter register (C)					
			33	User variable register (XF)					
			34	Parameter register (KF)					
			35	Monitoring register (MF)					
			45	Common parameter register (CF)					
			65	User variable register (XL)					
			66	Parameter register (KL)					
			67	Monitoring register (ML)					
			77	Common parameter register (CL)					
			97	User variable register (XD)					
			98	Parameter register (KD)					
			99	Monitoring register (MD)					
			104	LKT register (LD)					
			109	Common parameter register (CD)					
K122		All		Trace: Register number (bit#0-15) and depth definition (bit#16-23). Depth 0 for trace 0; depth 1 for trace 1 ,...	0	INT32_MIN	INT32_MAX	3	No
K123		All		Trace: Sampling time selection (period of 25us). As the acquisition is made in the PLTI, K123 must be a multiple of 2	2	2	2147483646	3	No
K124		All		Trace: Trigger edge selection	0	-1	1	3	No
K125		All		Trace: trigger mode selection	0	0	9	3	No
			0	Immediate acquisition					
			1	Start of movement					
			2	End of movement					
			3	Trigger at a position defined by KL127					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			4	Trigger at a given value (defined in K127 or KF127, KL127, KD127) of the trace defined in K126					
			5	Trigger never starts. An external freeze command in synchronized mode can stop the acquisition					
			6	Trigger at a given value (defined in K127 or KF127, KL127, KD127) of a register defined in K126					
			7	Trigger on a bit field state (K127 or KL127:0: bit to be low, K127 or KL127:1: bit to be high) of a register defined in K126					
			8	Trigger on a bit field rising or falling edge (K127 or KL127:0: bit on rising edge, K127 or KL127:1: bit on falling edge) of a register defined in K126					
			9	Continuous acquisition mode - No trigger					
K126		All		Trace: Definition of the trigger parameter according to the trigger mode (K125)	0	INT32_MIN	INT32_MAX	3	No
K127		All		Trace: Definition of the trigger level according to the trigger mode (K125)	0	INT32_MIN	INT32_MAX	3	No
KD127		All		Trace: Definition of the trigger level according to the trigger mode (K125)	0	FLOAT_MIN	FLOAT_MAX	3	No
KF127		All		Trace: Definition of the trigger level according to the trigger mode (K125)	0	FLOAT_MIN	FLOAT_MAX	3	No
KL127		All		Trace: Definition of the trigger level according to the trigger mode (K125)	0	INT64_MIN	INT64_MAX	3	No
K128		All		Trace: Number of points to acquire for the traces	1024	0	16384	3	No
K129		All		Trace: Pre or post trigger value	0	-16384	INT32_MAX	3	No
K133		All		AUT command mode for fine phase adjustment 0: old principle 1: new principle	0	0	1	4	Yes
K140		All		Error propagation checked group mask	1	0	15	3	Yes
			Bit 0	Error propagation group 1					
			Bit 1	Error propagation group 2					
			Bit 2	Error propagation group 3					
			Bit 3	Error propagation group 4					
K141		All		Error propagation published group mask	1	0	15	3	Yes
			Bit 0	Error propagation group 1					
			Bit 1	Error propagation group 2					
			Bit 2	Error propagation group 3					
			Bit 3	Error propagation group 4					
K142		All		Error interrupt label. K142:thread.<axis>=<label_error>. If the <label_error>=-1, in case of error the thread is stopped, the routine called errhandler is executed. If <label_error>=-2, in case of error the thread is not stopped	-1	-2	1023	1	No
K144		All		Thread not stopped by HLT HLB HLO. Bit#1 for thread 1, bit#2 for thread 2	0	0	6	1	No
K145		All		Searches limit stroke (SLS) mode	0	0	3	2	No
			0	Positive movement to search mechanical end stop					
			1	Negative movement to search mechanical end stop					
			2	Positive movement to search mechanical limit switch					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			3	Negative movement to search mechanical limit switch					
K146	UVWL	400		Undervoltage warning limit (M66=10) in [V/100]. If K146=0, the limit detection is disabled.	0	0	65000	2	Yes
K146	UVWL	48		Undervoltage warning limit (M66=10) in [V/100]. If K146=0, the limit detection is disabled.	0	0	6000	2	Yes
K146	UVWL	300		Undervoltage warning limit (M66=10) in [V/100]. If K146=0, the limit detection is disabled.	0	0	50000	2	Yes
K146	UVWL	VHP100		Undervoltage warning limit (M66=10) in [V/100].	1500	1500	12000	2	Yes
K146	UVWL	VHP48, ZxT		Undervoltage warning limit (M66=10) in [V/100].	900	900	6000	2	Yes
K147	UVEL	400		Undervoltage error limit (M64=9) in [V/100]. If K147=0, the limit detection is disabled.	0	0	65000	2	Yes
K147	UVEL	48		Undervoltage error limit (M64=9) in [V/100]. If K147=0, the limit detection is disabled.	0	0	6000	2	Yes
K147	UVEL	300		Undervoltage error limit (M64=9) in [V/100]. If K147=0, the limit detection is disabled.	0	0	50000	2	Yes
K147	UVEL	VHP100		Undervoltage error limit (M64=9) in [V/100].	1500	1500	12000	2	Yes
K147	UVEL	VHP48, ZxT		Undervoltage error limit (M64=9) in [V/100].	900	900	6000	2	Yes
K148	OVWL	400		Oversvoltage warning limit (M66=4) in [V/100]. If K148=0, the limit detection is disabled.	65000	0	65000	2	Yes
K148	OVWL	300		Oversvoltage warning limit (M66=4) in [V/100]. If K148=0, the limit detection is disabled.	39500	0	39500	2	Yes
K148	OVWL	VHP100		Oversvoltage warning limit (M66=4) in [V/100]. If K148=0, the limit detection is disabled.	12000	0	12000	2	Yes
K148	OVWL	48,VHP4 8,ZxT		Oversvoltage warning limit (M66=4) in [V/100]. If K148=0, the limit detection is disabled.	6000	0	6000	2	Yes
K149	OVEL	400		Oversvoltage error limit (M64=6) in [V/100]. If K149=0, the limit detection is disabled.	65000	0	65000	2	Yes
K149	OVEL	300		Oversvoltage error limit (M64=6) in [V/100]. If K149=0, the limit detection is disabled.	39500	0	39500	2	Yes
K149	OVEL	VHP100		Oversvoltage error limit (M64=6) in [V/100]. If K149=0, the limit detection is disabled.	12000	0	12000	2	Yes
K149	OVEL	48,VHP4 8,ZxT		Oversvoltage error limit (M64=6) in [V/100]. If K149=0, the limit detection is disabled.	6000	0	6000	2	Yes
K164		All		Start synchronization timeout (used with STI command)	0	0	600000	2	Yes
K165		All		Enable Scale Mapping correction	0	0	4	4	Yes
			0	Mapping deactivated					
			1	Linear mapping activated					
			4	Rotary mapping activated					
K166	WARNMSK	All		Warning mask	0	0	INT32_MAX	3	No
			Bit 1	AccurET Warning bit#1: I2t motor					
			Bit 2	AccurET Warning bit#2: Overtemperature					
			Bit 3	AccurET Warning bit#3: Inrush					
			Bit 4	AccurET Warning bit#4: Overvoltage					
			Bit 5	AccurET Warning bit#5: Encoder amplitude					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			Bit 6	AccurET Warning bit#6: Tracking					
			Bit 7	AccurET Warning bit#7: Position synchro					
			Bit 8	AccurET Warning bit#8: Hall effect					
			Bit 9	AccurET Warning bit#9: PWM synchro					
			Bit 10	AccurET Warning bit#10: Undervoltage					
			Bit 11	AccurET Warning bit#11: I2t drive					
			Bit 12	AccurET Warning bit#12: Power Relay					
			Bit 13	AccurET Warning bit#13: Init small mve					
			Bit 15	AccurET Warning bit#15: PWM update					
			Bit 17	AccurET Warning bit#17: EnDat encoder					
			Bit 18	AccurET Warning bit#18: HSEI encoder					
			Bit 19	AccurET Warning bit#19: Sequence debugger; breakpoints present.					
			Bit 28	AccurET Warning bit#28: Present when a command MVE RMVE STA STI is sent when axis is moving (Concatenate movement)					
			Bit 30	AccurET Warning bit#30: Corruption of data with continuous trace					
KL166		All		Position (dpi) where the Scale Mapping correction starts	0	INT48_MIN	INT48_MAX	4	Yes
KL167		All		Length (dpi) where the Scale Mapping correction is active	0	0	INT48_MAX	4	Yes
K171	DOUT	All		Activates / deactivates the digital outputs of the controller	0	0	3	3	No
K177		All		User word of M63 (bit#0 to 15) and user byte of SD2 (bit#8 to 15)	0	0	UINT16_MAX	2	No
K180	TIMSW	All		Duration of the speed window (used with WTS command)	0	0	1073741823	2	Yes
KL181	SPDW	All		Speed range of the window (used with WTS command)	0	0	3.4360e+10	2	Yes
K182		All but ZxT		Enables position capture on input according to K301. Writing a 1 in K182 enables the capture. Writing a 0 stops the process	0	0	1	1	No
K190		All but ZxT		Duration of the window in Force Control mode	0	0	20000	4	No
K191		All but ZxT		Delay between the touchdown detection and the change of the force control gain	0	0	20000000	4	No
KF191		All but ZxT		Force range of the window in Force Control mode	0	0	INT16_MAX	4	No
K192		All but ZxT		Smoothing estimated force in Force Control mode	0	0	20000000	4	No
K193		All but ZxT		Time during which the contact detection is ignored	0	0	20000000	4	No
K194		All		Time added at the end of the movement where KF60 is kept as current reference. At the end of this time, KF59 is kept as current reference	0	0	600000	4	No
K198		All		The register number is given by K198 when it is equal to Y or y (ex: Ky.1=...)	0	0	UINT16_MAX	1	No
K200		All		Time allowed for compiled sequence	4000	1000	8000	1	No
K201	MMC	All		Concatenated movements selection	0	0	3	2	Yes

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			0	Concatenated movements disabled					
			1	Concatenated movements enabled for MMD=1,17 and 24					
			2	Continuous back and forth movements enabled for MMD = 10, 26, 3 and 19 movement (can be stopped by MVE, BRK, HLT, HLB and HLO commands)					
			3	One back and forth MMD = 10, 26, 3 and 19 (can be stopped by BRK, HLT, HLB and HLO commands)					
K202	MMD	All		Movement type	1	1	26	2	Yes
			1	S-curve (jerk time) linear movement					
			3	Selects a predefined linear movement according to K230 value and executes the trajectory in a time given by K229					
			10	Look-up table linear movement					
			17	S-curve (jerk time) rotary movement					
			19	Selects a predefined rotary movement according to K230 value and executes the trajectory in a time given by K229					
			24	Infinite rotary movement					
			26	Look-up table rotary movement					
K203	LTN	All		Look-up table number movement	0	0	9	2	Yes
K204	LTI	All		Time to execute a look-up table movement	10000	4	1250000	2	Yes
KF205	CAM	All		Came value (in percent). Decreases cinematic quantities (speed, acceleration) and increases time quantities	100	1	120	2	Yes
KL206	BRKDEC	All		Brake deceleration (depth 0: with BRK and HLB command; depth1: brake ramp with the stage protection)	1000000	256	5.4976e+11	2	Yes
K207		All		LKT mode selection	0	0	1	2	Yes
			0	LKT movement running to target defined by MVE command					
			1	LKT movement with same starting and end point. Parameter KL208 defines the amplitude of the movement					
KL208		All		Maximum stroke for LKT movement with K207=1 and for MMD=10 & 26	0	INT48_MIN	INT48_MAX	2	Yes
K209		All		Rotary movement type selection	0	0	2	2	Yes
			0	Movement always positive					
			1	Movement always negative					
			2	Movement minimum distance					
KL210	TARGET	All		Gives the target position for STA or STI command	0	INT48_MIN	INT48_MAX	2	No
KL211	SPD	All		Absolute maximum speed	2000000	1	3.4360e+10	2	Yes
KL212	ACC	All		Absolute maximum acceleration and deceleration	1000000	256	5.4976e+11	2	Yes
K213	JRT	All		Jerk time	0	0	500	2	Yes
K220		All		External reference type	2	1	77	2	Yes
			1	Reference type is a user's variable X					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
			2	Reference type is a parameter K					
			3	Reference type is a monitoring M					
			13	Reference type is a register C					
			33	Reference type is a user's variable XF					
			34	Reference type is a parameter KF					
			35	Reference type is a monitoring MF					
			45	Reference type is a register CF					
			65	Reference type is a user's variable XL					
			66	Reference type is a parameter KL					
			67	Reference type is a monitoring ML					
			77	Reference type is a register CL					
K221		All		External reference index	0	0	511	2	Yes
K222		All		External reference depth	0	0	7	2	Yes
K223		All		External reference offset	0	INT32_MIN	INT32_MAX	2	Yes
KF224		All		External reference gain	1	-2147483647	INT32_MAX	2	Yes
K225		All		External mean filter on the position offset compensation. Valid only at the depth 1. 1=no filtering. 2-16 = value * MLTI, length of the mean filter	1	1	16	2	Yes
K226		All		Advance gain scheduling points	0	INT32_MIN	INT32_MAX	4	No
KF226		All		Advance gain scheduling points	0	FLOAT_MIN	FLOAT_MAX	4	No
KL226		All		Advance gain scheduling points	0	INT64_MIN	INT64_MAX	4	No
KF227		All		Allows to apply a gain on the value extracted from the look-up table (LDx:9)	1	-100	100	2	Yes
K229		All		Execution time for calculated movement profile selected by K230	10000	4	1250000	2	Yes
K230		All		Calculated movement profile selection for MMD = 3 or 19					
			0	Triangular (speed) movement					
			1	S-curve (full jerk) movement					
			2	Sine modified movement					
			3	Real sine movement					
K240		All		Movement type conversion (depth 0: main, depth 1: secondary, depth 2: auxiliary). 0: linear movement, 1: rotary movement, 2: EnDat linear movement, 3: EnDat rotary movement	0	0	3	4	Yes
K241		All		Encoder period in [nm] for linear encoder or number of period for rotary encoder (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1	0	INT32_MAX	4	Yes
K242		ZxT		Position multiplication factor	1	1	INT32_MAX	4	Yes
K242		All but ZxT		Position multiplication factor used to calculate the position unit with indirect single encoder and indirect Dual Encoder Feedback (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1	1	INT32_MAX	4	Yes

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
K243		ZxT		Position division factor	1	1	INT32_MAX	4	Yes
K243		All but ZxT		Position division factor used to calculate the position unit with indirect single encoder and indirect Dual Encoder Feedback (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1	1	INT32_MAX	4	Yes
KF247	KOFFP	All		Current offset feedforward gain	0	-31000	31000	4	No
KF248	KFCP	All		Current friction feedforward gain	0	0	31000	4	No
K249		All		Disables all feedforward. 0:all feedforward are enabled 1:all feedforward are disabled	0	0	1	3	No
K250		All		Selection ffwd dry friction. 0 select dry friction with hysteresis, 1 select dry friction without hysteresis	0	0	1	3	No
KL250		ZxT		ZxT Stage feedforward (offset)	0	INT48_MIN	1.4074e+14	4	No
K251		All		Filter for dry friction feedforward (0, or 1, 2, 4, 8, 16, 32, 64 PLTI)	0	0	64	3	No
K291	TFTD	All		Time delay of trajectory filter	0	0	4095	2	Yes
K292	TFAR	All		Amplitude ratio of trajectory filter	0	0	4096	2	Yes
K294		All		Number of point for Scale Mapping correction	0	0	8192	4	No
K297		All		Mask for active thread information on M63 status (bit#1 thread1, bit#2 thread2)	0	0	6	1	No
K298		All		Input edge mask for BRK (bit 0-15 rising edge, bit 16-31 falling edge, depth 0 DIN, depth 1 FDIN)	0	INT32_MIN	INT32_MAX	4	No
KF299		All but ZxT		Force threshold for touchdown detection with the force feedback	INT32_MAX	INT32_MIN	INT32_MAX	4	No
K300		All		Input state mask for STI (bit 0-15: state 1, bit 16-31: state 0, depth 0: DIN, depth 1: FDIN)	0	0	INT32_MAX	1	No
KF300		All but ZxT		Kt motor for Force regulator	0	0	INT32_MAX	4	No
K301		All but ZxT		Input edge mask for position capture (bit 0-15: rising edge, bit 16-31: falling edge, depth 0: DIN, depth 1: FDIN)	0	0	INT32_MAX	1	No
KF301		All but ZxT		Acceleration gain for Force estimator	0	0	INT32_MAX	4	No
K302		All but ZxT		Force Control mode activation & configuration	0	0	31	4	No
			Bit 0	The force control mode is selected					
			Bit 2	Approach with constant speed					
KF302		All but ZxT		Current offset for Force estimator	0	-31000	31000	4	No
KL302		All		Input state mask for limitation KF60/K31 (bit 0-31: state 1, bit 32-63: state 0, depth 0: CDIN of both axes)	0	0	4.5036e+15	4	No
K303		All		Input edge mask for RSD (bit 0-15 rising edge, bit 16-31 falling edge, depth 0 DIN, depth 1 FDIN)	0	0	INT32_MAX	4	No
KF303		All but ZxT		Speed gain for Force estimator	0	0	INT32_MAX	4	No
K304		All		Input state mask for motor overtemperature protection (bit 0-15: state 1, bit 16-31: state 0, depth 0: DIN)	0	0	INT32_MAX	4	No

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
KF304		All but ZxT		Dry friction gain for Force estimator	0	0	31000	4	No
KF305	KPF	All but ZxT		Kp gain for force regulator in mode 0	0	0	10000000	4	No
KL305		All		Input state mask for enabling the motor power ON (bit 0-31: state 1, bit 32-63: state 0, depth 0: CDIN of both axes)	0	0	4.5036e+15	4	No
KF306	KIF	All but ZxT		Ki gain for force regulator in mode 0	0	0	10000000	4	No
KL306		All		DIN/FDIN edge mask for starting Master/Slave Compensation (bit 0-31 rising edge, bit 32-63 falling edge, depth 0: CDIN of both axes, depth 1 FDIN)	0	0	4.5036e+15	1	No
K307		All but ZxT		Integrator limit for Force regulator	INT32_MAX	0	INT32_MAX	4	No
K308		All but ZxT		Distance of the collapse detection threshold	0	INT32_MIN	INT32_MAX	4	No
KF308		All but ZxT		Lever arm for force regulator	1	0	INT32_MAX	4	No
KF309		All but ZxT		Force estimator sensor offset	0	INT32_MIN	INT32_MAX	4	No
KF310		All but ZxT		Force estimator sensor gain	0	-32768000	32768000	4	No
K311		All but ZxT		Force estimator sensor definition, depth 0 input type, depth 1 input index, depth 2 input depth, depth 3 axis number	0	0	511	4	No
K312		All		Master Gantry axis number	-1	-1	62	4	Yes
KF312	KPFC	All but ZxT		Kp gain for force regulator in mode 1	0	0	10000000	4	No
KF313	KIFC	All but ZxT		Ki gain for force regulator in mode 1	0	0	10000000	4	No
KL313		All		Gantry axes mask	0	0	INT64_MAX	4	Yes
K314		All		Beam Gantry axis number	-1	-1	62	4	No
KF314	KDFC	All but ZxT		Kd gain for force regulator in mode 1	0	0	10000000	4	No
KL314		All but ZxT		Force Control: upper limit after touchdown	0	0	INT32_MAX	4	No
KF316		All but ZxT		Measured current gain for Force estimator	1	-1	1	4	No
KF317		All but ZxT		Cogging compensation gain for Force estimator	0	-1	1	4	No
KF318	KDF	All but ZxT		Kd gain for force regulator in mode 0	0	0	10000000	4	No
KF319	KFFFWFC	All but ZxT		Force feedforward gain for force regulator in mode 1	0	0	10000000	4	No
K320		All but ZxT		Fast Trigger: Combination 1 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K321		All but ZxT		Fast Trigger: Combination 2 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
K322		All but ZxT		Fast Trigger: Combination 3 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K323		All but ZxT		Fast Trigger: Combination 4 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K324		All but ZxT		Fast Trigger: Combination 5 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K325		All but ZxT		Fast Trigger: Combination 6 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K326		All but ZxT		Fast Trigger: Combination 7 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K327		All but ZxT		Fast Trigger: Combination 8 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K328		All but ZxT		Fast Trigger: Combination 9 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K329		All but ZxT		Fast Trigger: Combination 10 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K330		All but ZxT		Fast Trigger: Combination 11 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K331		All but ZxT		Fast Trigger: Combination 12 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K332		All but ZxT		Fast Trigger: Combination 13 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K333		All but ZxT		Fast Trigger: Combination 14 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K334		All but ZxT		Fast Trigger: Combination 15 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K335		All but ZxT		Fast Trigger: Combination 16 (depth 0: mask for DOUT, depth 1: mask for FDOUT, depth 3: reset of event counter group(M351), depth 7: mask for users status)	0	INT32_MIN	INT32_MAX	3	No
K336		ZxT		Fast Trigger: Selects the position type (3=publishing the real position and speed, 4=publishing the theoretical position and speed)	0	0	4	3	No
K336		All but ZxT		Fast Trigger: Selects the position type (0=on real position and speed, 1=on theoretical position and speed, 2=via registers, 3=publishing the real position and speed, 4=publishing the theoretical position and speed)	0	0	4	3	No
K337		All but ZxT		Fast Trigger: Continuous setting (Depth 0: When indicating there is enough place in memory; Depth1: Which user status could be used)	0	0	INT32_MAX	3	No
K338		All but ZxT		Fast Trigger: Box tolerance. When box is enabled, the trigger event will fire only when the position is within the +/- tolerance	0	0	INT32_MAX	3	No
K339		All but ZxT		Fast Trigger: Pulse Generator pulse periodicity (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	INT32_MAX	3	No
KF339		All but ZxT		Fast Trigger: Pulse Generator pulse period modulation coefficient (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	FLOAT_MAX	3	No
K340		All but ZxT		Fast Trigger: Pulse Generator 1 mask (DOUT (bit 0 to 1 on depth 0), FDOUT (bit 0 to 3 on depth 1))	0	0	1048576	3	No
K341		All but ZxT		Fast Trigger: Pulse Generator 2 mask (DOUT (bit 0 to 1 on depth 0), FDOUT (bit 0 to 3 on depth 1))	0	0	1048576	3	No
K342		All but ZxT		Fast Trigger: Pulse Generator delay (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	INT32_MAX	3	No

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
KF342		All but ZxT		Fast Trigger: Pulse Generator delay modulation coefficient (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	FLOAT_MAX	3	No
K343		All but ZxT		Fast Trigger: Pulse Generator pulse width (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	INT32_MAX	3	No
KF343		All but ZxT		Fast Trigger: Pulse Generator pulse width modulation coefficient (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	FLOAT_MAX	3	No
K344		All but ZxT		Fast Trigger: Pulse Generator interval pulse (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	INT32_MAX	3	No
K345		All but ZxT		Fast Trigger: Pulse Generator pulse count (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	0	0	INT32_MAX	3	No
K346		All		Delays for the motor brake function. Depth0: Power On; Depth1: Power Off	250	-2500	2500	2	Yes
K347		All but ZxT		Fast Trigger: Missed event timeout	2	2	INT32_MAX	3	No
K348		All but ZxT		Fast Trigger: Missed event detection tolerance	0	0	INT32_MAX	3	No
K349		All but ZxT		Fast Trigger: Missed event action (0: generate an error - disable the trigger - fixed the state of output with k357 1: generate a warning - disable the trigger - fixed the state of output with k357)	0	0	1	3	No
K350		All		FDOUT mask for SIN signal (available only at depth 1; bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	4	No
K351		All		FDOUT mask for COS signal (available only at depth 1; bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	4	No
K352		All		FDOUT mask for index signal (available only at depth 1; bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	4	No
K353		All but ZxT		Fast Trigger: External register reference type. Depth 0: modulation Pulse Generator1; Depth 1: modulation Pulse Generator2; Depth 2: position source for trigger coming from any register; Depth 3: reference position for missed event feature	2	1	67	3	No
			1	Reference type is a user's variable X					
			2	Reference type is a parameter K					
			3	Reference type is a monitoring M					
			13	Reference type is a register C					
			33	Reference type is a user's variable XF					
			34	Reference type is a parameter KF					
			35	Reference type is a monitoring MF					
			45	Reference type is a register CF					
			65	Reference type is a user's variable XL					
			66	Reference type is a parameter KL					
			67	Reference type is a monitoring ML					
			77	Reference type is a register CL					
K354		All but ZxT		Fast Trigger: External register reference index. Depth 0: modulation Pulse Generator1; Depth 1: modulation Pulse Generator2; Depth 2: position source for trigger coming from any register; Depth 3: reference position for missed event feature	0	0	511	3	No

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
K355		All but ZxT		Fast Trigger: External register reference depth. Depth 0: modulation Pulse Generator1; Depth 1: modulation Pulse Generator2; Depth 2: position source for trigger coming from any register; Depth 3: reference position for missed event feature	0	0	7	3	No
K356		All but ZxT		Fast Trigger: External register reference axis. Depth 0: modulation Pulse Generator1; Depth 1: modulation Pulse Generator2; Depth 2: position source for trigger coming from any register; Depth 3: reference position for missed event feature	0	0	1	3	No
K357		All		Mask DOUT/FDOUT for security management (bit 0-15: state 1, bit 16-31: state 0, depth 0: DOUT, depth1: FDOUT)	0	0	1048576	3	No
K358		All		Output state mask for STI command (bit 0-15: state 1, bit 16-31: state 0, depth 0: DOUT, depth 1: FDOUT)	0	0	1048576	1	No
K359		All but ZxT		Fast Trigger: DOUT mask for trigger signal (bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	3	No
K360		All but ZxT		Fast Trigger: Mean filter for trigger position. 1=no filtering. 2-8 = value * PLTI, length of the mean filter. Depth 0: Mean filter for trigger position coming via register; Depth 1: Compensation for position used by Missed event feature.	1	1	8	3	No
KF360		All but ZxT		Fast Trigger: Time compensation in sec	0.00025	0	1	3	No
K363		All		DOUT mask for motor brake function (bit 0-15: state 1 when axis is powered (DOUT1 bit#0 DOUT2 bit#1), bit 16-31: state 0 when axis is powered (DOUT1 bit#16 DOUT2 bit#17))	0	0	1048576	3	No
K364		All		External motor braking configuration for phasing (K90 = 6 7 or 8): 0 phasing with brake open, 1 phasing with brake close.	0	0	1	2	Yes
K365		All		Stage error mapping activation (1=activated, 0=deactivated)	0	0	1	1	No
KL370		All		Position (dpi) where the cogging compensation starts	0	INT48_MIN	INT48_MAX	4	No
KL371		All		Length (dpi) where the cogging compensation is active	0	0	INT48_MAX	4	No
K372		All		Number of point for cogging compensation	8192	0	8192	4	No
K373		All but ZxT		Position spring offset for Force estimator	0	INT32_MIN	INT32_MAX	4	No
KF373		All but ZxT		Force spring factor	0	0	INT32_MAX	4	No
K374		All but ZxT		Maximum compression of the spring position (saturation) for the Force estimator (between 0 and K374)	0	INT32_MIN	INT32_MAX	4	No
KF374		All but ZxT		Damping spring factor	0	0	INT32_MAX	4	No
K375		48, 300, 400		EtherCAT ControlWord and StatusWord source selection depending on the mode of operation set (object 6061h)	0	0	1	4	Yes
K380		All		Stage protection threshold low speed. This value is taken into account only at the power on of the controller.	2000000	1	INT32_MAX	4	Yes
K381		All		Stage protection minimum deceleration. This value is taken into account only at the power on of the controller.	1000000	1	INT32_MAX	4	Yes
K415		All		Enable fast stack for Compiled Sequences. Warning: the stack size is significantly reduced	0	0	1	3	No
			0	Sequence stack in slow external memory (2048 32-bit integers)					
			1	Sequence stack in fast internal memory (128 32-bit integers)					

K	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)	Gantry axes identical (**)
K420		All		Type register for position coming via register. (depth 0: main, depth 1: secondary, depth 2: auxiliary)	2	1	77	2	Yes
			1	Reference type is a user's variable X					
			2	Reference type is a parameter K					
			3	Reference type is a monitoring M					
			13	Reference type is a register C					
			33	Reference type is a user's variable XF					
			34	Reference type is a parameter KF					
			35	Reference type is a monitoring MF					
			45	Reference type is a register CF					
			65	Reference type is a user's variable XL					
			66	Reference type is a parameter KL					
			67	Reference type is a monitoring ML					
			77	Reference type is a register CL					
K421		All		Index register for position coming via register. (depth 0: main, depth 1: secondary, depth 2: auxiliary)	0	0	511	2	Yes
K422		All		Depth register for position coming via register. (depth 0: main, depth 1: secondary, depth 2: auxiliary)	0	0	7	2	Yes
K423		All		Use other axis register for position coming via register. (depth 0: main, depth 1: secondary, depth 2: auxiliary)	0	0	1	2	Yes
K424		All		Mean filter for position via register. 1=no filtering. 2-8 = value * PLTI, length of the mean filter. (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1	1	8	2	Yes
KF446	ACF	All but ZxT		Dual Encoder Feedback: Accuracy convergence factor	0.5	0	1	2	Yes
KF453		All but ZxT		Dual Encoder Feedback: Smoothing factor	1000	0	20000	2	No
K460		All		Extended number of RTV slots	0	0	1	3	No
K480		ZxT		ZxT Factory offset used to align the 0 position of the 4th encoders	0	-268435456	268435455	4	No
K481		ZxT		Duration of the global ZxT window in MLTI (used with WTWZXT command)	0	0	393210	2	No
K482		ZxT		Position range of the global ZxT window (used with WTWZXT command)	0	0	1073741823	2	No
K483		ZxT		Wafer radius (used with WTWZXT command)	0	0	1073741823	2	No
KF490		ZxT		ZxT Conversion radius	0	0	5	3	Yes
KF492		ZxT		ZxT Current reference value	0	INT16_MIN	INT16_MAX	4	Yes
KF493		ZxT		ZxT Current offset value	0	INT16_MIN	INT16_MAX	4	Yes
K495		ZxT		ZxT Sub-axis selection	0	0	1	4	No
			0	Rx selected					
			1	Ry selected					

18. Common parameters C

(*) Gantry action in level 2 (refer to [§12](#)):

- Action 1: The master can execute the command, the slave must generate an error
- Action 2: The master duplicates and executes the command, the slave must generate an error if it is for itself
- Action 3: The master and the slave can execute the command, no error is generated
- Action 4: The master and slave generate an error
- Action 5: The master ignores the command and slave must generate an error
- Action 6: The master duplicates and executes the command, the slave must ignore it
- Action 7: The master can execute the command, the slave must ignore it

C	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)
C1		ZxT		Selects the manager mode. This value is taken into account only at the power on of the controller	0	0	0	4
			0	Manager mode TransnET				
C1		All but ZxT		Selects the manager mode. This value is taken into account only at the power on of the controller	0	0	5	4
			0	Manager mode TransnET				
C2	IPADD	All		IP address. If C2 = 0 => DHCP	0	INT32_MIN	INT32_MAX	4
C3	NAME	All		Controller name	0	INT32_MIN	INT32_MAX	4
C5	FDOUT	All		Activates / deactivates the fast outputs of the controller	0	0	15	1
C6	XDOUT	48, 300, 400		Activates / deactivates the digital outputs of the optional board	0	0	255	1
C7	XAOUT	48, 300, 400		Optional board analog outputs (4 for AccurET I/O) (32767 = 20V between XAOUT+ and XAOUT-, -32768 = -20V between XAOUT+ and XAOUT-)	0	INT16_MIN	INT16_MAX	1
C10		All but ZxT		Stage protection activation (1 or 2=activated, 0=deactivated). This value is taken into account only at the power on of the controller	0	0	2	4
			1	Stage protection active with in case of error a brake with a step to 0 on the speed				
			2	Stage protection active with in case of error a brake with a ramp defined by KL206:1 on the speed				
C11		All		Stage protection brake time in number of [10ms] (min 2==>20ms, max 40==>400ms). This value is taken into account only at the power on of the controller.	20	2	40	4
C12		All		DOUT mask stage protection indicates that the AccurET has finished braking (speed is lower than predefined threshold & stage protection braking time has elapsed)	0	0	983055	4
C13		All		DOUT mask stage protection driving an external short-circuit relay after the speed braking	0	0	983055	4
C17	XAIO	48, 300, 400		XAIN analog input offset	0	INT32_MIN	INT32_MAX	1
CF18	XAIA	48, 300, 400		XAIN analog input gain	1	-32768000	32768000	1
C19		ZxT		Enable command synchro between different controllers	1	1	1	1
C19		All but ZxT		Enable command synchro between different controllers	0	0	1	1
C27	XAOO	48, 300, 400		XAOUT analog output offset	0	INT32_MIN	INT32_MAX	1

C	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)
CF28	XAOA	48, 300, 400		XAOOUT analog output gain	1	-32768000	32768000	1
C30	XSRT	48, 300, 400		XAOOUT source register type	0	0	77	1
			0	Immediate value defined by C7				
			1	Reference type is a user's variable X				
			2	Reference type is a parameter K				
			3	Reference type is a monitoring M				
			13	Reference type is a register C				
			33	Reference type is a user's variable XF				
			34	Reference type is a parameter KF				
			35	Reference type is a monitoring MF				
			45	Reference type is a register CF				
			65	Reference type is a user's variable XL				
			66	Reference type is a parameter KL				
			67	Reference type is a monitoring ML				
			77	Reference type is a register CL				
C31	XSRI	48, 300, 400		XAOOUT source register index	0	0	512	1
C32	XSRA	48, 300, 400		XAOOUT source register axis. Choose the axis to link with the monitoring source	0	0	1	1
C40		All		SPI data out, also starts communication on write if SPI set as master	0	INT32_MIN	INT32_MAX	1
C41		All		SPI configuration	0	0	16777215	1
			Bit 0	Master or Slave configuration for SPI. Bit = 0 for Slave, Bit = 1 for Master				
			Bit 1	CPHA configuration. Bit = 0 for data valid on rising edge, Bit = 1 for data valid on failing edge.				
			Bit 2	DWORD configuration. Bit = 0 for MSB first, Bit = 1 for LSB first.				
			Bit 4..6	Clock speed configuration. 000: 5MHz, 001:2.5MHz ... lowest is 111: 39kHz				
			Bit 8..9	Size of transmission configuration. 00: 8bit, 01: 16bit, 10: 32bit, 11:N.U.				
			Bit 16	Enable CRC. Bit = 0 disable, Bit = 1 enable				
			Bit 18..20	CRC size. 000: CRC-1, 010: CRC-3, 011: CRC-4, 100: CRC-5, 101: CRC-6, 110: CRC-7, 111: CRC-8				
			Bit 22..23	CRC error management. 00: increase CRC error counter M262, 01: re-asks the SPI data and in case of a 2nd error increase CRC error counter M262				
CF60		ZxT		Current saturation value (equivalent to KF60, but for the physical axes)	0	0	32768	4
C100		300, 400		Enable the feature and define the maximum sag duration (in MLTI increments). The feature is disabled if C100 = 0.	0	0	1250	1
C101		300, 400		Speed reduction in percent during the voltage sag. This reduction is only applied if the motor is accelerating when the sag is detected.	100	0	100	1

C	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)
C102		400		Voltage threshold triggering the power sag protection when input voltage M91 becomes smaller than C102. Value in [V/100].	0	0	65000	1
C102		300		Voltage threshold triggering the power sag protection when input voltage M91 becomes smaller than C102. Value in [V/100].	0	0	50000	1
CF107	AOUT	All VHP		AOUT analog outputs (32767.0 = 20V between AOUT+ and AOUT-, -32768.0 = -20V between AOUT+ and AOUT-)	0	INT16_MIN	INT16_MAX	1
CF117		All VHP		AIN analog input offset	0	INT32_MIN	INT32_MAX	1
CF118		All VHP		AIN analog input gain	1	-32768000	32768000	1
CF127	AOO	All VHP		AOUT analog outputs offset	0	INT32_MIN	INT32_MAX	1
CF128	AOA	All VHP		AOUT analog outputs gain	1	-32768000	32768000	1
C130	SRT	All VHP		AOUT source register type	0	0	77	1
			0	Immediate value defined by C7				
			1	Reference type is a user's variable X				
			2	Reference type is a parameter K				
			3	Reference type is a monitoring M				
			13	Reference type is a register C				
			33	Reference type is a user's variable XF				
			34	Reference type is a parameter KF				
			35	Reference type is a monitoring MF				
			45	Reference type is a register CF				
			65	Reference type is a user's variable XL				
			66	Reference type is a parameter KL				
			67	Reference type is a monitoring ML				
			77	Reference type is a register CL				
C131	SRI	All VHP		AOUT source register index	0	0	512	1
C132	SRA	All VHP		AOUT source register axis. Choose the axis to be linked with the monitoring source	0	0	1	1
CL211		All		CDIN mask for generating error 211 (available at depth 0)	0	INT64_MIN	INT64_MAX	1
CL212		All		CDIN mask for generating error 212 (available at depth 0)	0	INT64_MIN	INT64_MAX	1
CL213		All		CDIN mask for generating error 213 (available at depth 0)	0	INT64_MIN	INT64_MAX	1
CL240		All		Gantry min beam position	0	INT48_MIN	INT48_MAX	1
CL241		All		Gantry max beam position	0	INT48_MIN	INT48_MAX	1
C245		All		Determines the level for the Gantry functionality. Depth 0: 0 => no gantry, 1=> Gantry protection active, 2=> only 1 axis visible for HMI; depth 1 configuration when Dual Encoder Feedback Gantry	0	0	2	1
			0	Gantry in level 0 (no parameter checked, no additional control, and both axes are visible)				
			1	Gantry in level 1 (parameters are checked, additional control on power on, tracking, error, and both axes are visible)				

C	Alias	Product	Values or bit#	Description	Default value	Minimum value	Maximum value	Gantry action (*)			
			2	Gantry in level 2 (parameters are checked, additional control on power on, tracking, error, only the master axis of the gantry is visible from HMI)							
C246		All		Determines the advanced Gantry feedforward. 0 => no advanced Gantry feedforward, 1=> advanced Gantry feedforward	0	0	1	1			
C305		All		FDIN mask for SPI data input (available at depth 1). Bit#0-5 respectively FDIN1-FDIN6 (non-inverted); Bit#16-21 respectively FDIN1-FDIN6 (inverted)	0	0	4128831	1			
C306		All		FDIN mask for SPI clock input (available at depth 1). Bit#0-5 respectively FDIN1-FDIN6 (non-inverted); Bit#16-21 respectively FDIN1-FDIN6 (inverted)	0	0	4128831	1			
C307		All		FDIN mask for SPI chip select input (available at depth 1). Bit#0-5 respectively FDIN1-FDIN6 (non-inverted); Bit#16-21 respectively FDIN1-FDIN6 (inverted)	0	0	4128831	1			
C353		All		FDOUT mask for bus synchronization (available at depth 1; bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	1			
C354		All		FDOUT mask for MLTI cycle (available at depth 1; bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	1			
C355		All		FDOUT mask for PLTI cycle (available at depth 1; bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	1			
C358		All		Mask for output managed over RTV slots	0	INT32_MIN	INT32_MAX	1			
C359		All but ZxT		Fast Trigger: FDOUT mask for trigger signal (available at depth 1; bit 0-15: non-inverted, bit 16-31: inverted)	0	0	1048576	1			
C360		All		FDOUT mask for SPI data output (available at depth 1). Bit#0-3 respectively FDOUT1-FDOUT4 (non-inverted); Bit#16-19 respectively FDOUT1-FDOUT4 (inverted)	0	0	983040	1			
C361		All		FDOUT mask for SPI clock output (available at depth 1). Bit#0-3 respectively FDOUT1-FDOUT4 (non-inverted); Bit#16-19 respectively FDOUT1-FDOUT4 (inverted)	0	0	983040	1			
C362		All		FDOUT mask for SPI chip select output (available at depth 1). Bit#0-3 respectively FDOUT1-FDOUT4 (non-inverted); Bit#16-19 respectively FDOUT1-FDOUT4 (inverted)	0	0	983040	1			
C375		48, 300, 400		Configured Station Alias set by the EtherCAT master. This is used for emulating EEPROM	0	0	UINT16_MAX	1			
C490	ZxT	ZxT transfer matrix selection				0	0	23			
		10 Z3TM configuration 3 encoders and 3 actuators with alpha 0 degree									
		11 Z3TM configuration 3 encoders and 3 actuators with alpha 30 degree									
		12 Z3TM configuration 3 encoders and 3 actuators with alpha 90 degree									
		13 Z3TM configuration 3 encoders and 3 actuators with alpha 120 degree									
		14 Z3TM configuration 3 encoders and 3 actuators with alpha 180 degree									
		15 Z3TM configuration 3 encoders and 3 actuators with alpha 210 degree									
		16 Z3TM configuration 3 encoders and 3 actuators with alpha 270 degree									
		17 Z3TM configuration 3 encoders and 3 actuators with alpha 300 degree									
		20 Z3TH configuration 4 encoders and 4 actuators with alpha = 0 degree									
		21 Z3TH configuration 4 encoders and 4 actuators with alpha = 90 degree									
		22 Z3TH configuration 4 encoders and 4 actuators with alpha = 180 degree									
		23 Z3TH configuration 4 encoders and 4 actuators with alpha = 270 degree									
CF490		ZxT		ZxT Stage feedforward, moving mass factor	0	0	1000000000	4			
C495		ZxT		ZxT Feedforward propagation delay (0..31 x PLTI)	0	0	31	4			

19. Monitorings M

(*) Gantry feedback in level 2 (refer to [§12.](#)):

- Feedback 1: The master shows the master value and the slave shows the slave.
- Feedback 2: The master shows the gantry value and the slave shows the slave.
- Feedback 3: The master shows the gantry value (depth0), the master value (depth1), the slave value (depth2), and the slave shows the slave value.
- Feedback 4: The master shows a logical combination of the master and the slave (depth0), master (depth1), slave (depth2), and the slave shows the slave value. The logical combination can be either an AND, OR, MIN or MAX depending on the register.

A gantry value means a computed value based on the master and the slave representing at best the real state of the gantry. Today this is the same as the master, but future improvements may change this behavior.

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
M0		All		Theoretical position (dpi). Does not take care of SET command. LSL part of ML0	3
ML0		All		Theoretical position (dpi). Does not take care of SET command	3
M1		All		Real position (dpi). Takes the mapping corrections into account but does not take care of SET command. LSL part of ML1	3
MF1		All		Position loop proportional gain value KF1 modified by gain scheduling	1
ML1		All		Real position (dpi). Takes the mapping corrections into account but does not take care of SET command	3
M2		All		Tracking error. This is the difference between ML0 and ML1	3
MF2		All		Position loop speed feedback gain value KF2 modified by gain scheduling	1
M3		All		Maximum tracking error during movement	2
ML3		All		Braking distance	1
M4		All		Offset between dpi and upi. [upi] = ([dpi] + ML4). LSL part of ML4	1
ML4		All		Offset between dpi and upi. [upi] = ([dpi] + ML4)	1
M5		All		Covered distance from controller boot or reboot to the homing position (dpi). LSL part of ML5	1
ML5		All		Covered distance from controller boot or reboot to the homing position (dpi)	1
M6		All		Theoretical position (upi). Takes SET command into account. LSL part of ML6	3
ML6		All		Theoretical position (upi). Takes SET command into account	3
M7		All		Real position (upi). Takes SET command and mapping correction into account. LSL part of ML7	3
ML7		All		Real position (upi). Takes SET command and the mapping correction into account	3
M9		All		Scale Mapping compensation value (dpi) LSL part of ML9	1
ML9		All		Scale Mapping compensation value (dpi)	1
M10		All		Theoretical speed after adv filter (dsi)	3
M11		All		Real speed after advanced filter depth 0 (dsi)	3
M13		All		Position given by the secondary encoder (dpi2). LSL part of ML13	1
ML13		All		Position given by the secondary encoder (dpi2)	1
M14		All		Theoretical acceleration (dai)	3
M15		All		Position given by the auxiliary encoder (dpi3). LSL part of ML15	1
ML15		All		Position given by the auxiliary encoder (dpi3)	1
M17		All		Real position after Scale Mapping (dpi), before Stage Mapping. LSL part of ML17	1
ML17		All		Real position after Scale Mapping (dpi), before Stage Mapping	1
M18		All		Master/Slave Compensation value	1
M19		All		Real position coming from the encoder, before Scale Mapping (dpi). LSL part of ML19	1
ML19		All		Real position coming from the encoder, before Scale Mapping (dpi)	1
MF20	RCUR1	All		Real current in phase 1	1

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
M21		All		Stage Mapping compensation value (dpi)	1
MF21	RCUR2	All		Real current in phase 2	1
M22		All		Compensation value on real position adding to the measured position [dpi]	1
MF22	RCUR3	All		Real current in phase 3	1
MF24		48, 300, 400		Phase PWM value. Depth0: phase 1, depth1: phase 2 and depth2: phase 3	1
MF24		All VHP		Output phase voltage. Depth0: phase 1, depth2: phase 3	1
M25		All		Phase angle	1
M26		ZxT		Homing MIMO Offset	1
MF26	UD	48, 300, 400		Ud vector control part	1
MF27	TID	All		Theoretical current Id reference	1
MF28	RID	All		Real current Id measured	1
MF29	UQ	48, 300, 400		Uq vector control part	1
MF30	TIQ	All		Theoretical current Iq for current loop after KF60 limitation (Physical axes)	1
MF31	RIQ	All		Real current Iq measured	1
MF32		All		Theoretical current Iq before advanced filters	1
M33		All		Offset secondary position. LSL part of ML33	1
MF33		All		Position loop integrator output value	1
ML33		All		Offset secondary position	1
MF34		All		Position loop integrator gain value KF4 modified by gain scheduling	1
M35		All		Offset auxiliary position. LSL part of ML35	1
ML35		All		Offset auxiliary position	1
M36		All		Inferior position after a SLS command. LSL part of ML36	4
ML36	MINHL	All		Inferior position after a SLS command	1
M37		All		Superior position after a SLS command. LSL part of ML37	4
ML37	MAXHL	All		Superior position after a SLS command	1
M38		All		Time between the WTW command and the setup bit#5 of M60 (motor in the windows defined by K38 and K39)	1
ML39		All		Index position on the main encoder (dpi)	1
M40	SINENC	All		Analog encoder sine signal after correction (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1
ML40		All		Index position on the secondary encoder (dpi2)	1
M41	COSENC	All		Analog encoder cosine signal after correction (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1
M43	AMPENC	All		Analog encoder sine^2 + cosine^2 (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1
M44		All		Encoder limit switch	1
			Bit 0	Encoder limit switch EHO (L1 or H)	
			Bit 1	Encoder limit switch ELS (L2 or L)	
ML45		All		Homing offset on absolute position	1
M48		All		Digital Hall effect sensor signal	1
M50	DIN	All		Digital inputs	1
MF51	AIN	All VHP		Analog input after gain and offset compensation	1
M52	FDIN	All		Common fast inputs	1
M53		All VHP		Analog input before gain and offset compensation	1
M54	CDIN	All		Common digital inputs from both axes	1
M55	XDIN	48, 300, 400		Optional board digital inputs	1
M56	XAIN	48, 300, 400		Optional board analog inputs	1
M58		All but ZxT		Status position capture	2
			Bit 0	Capture position 1 activated	

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
			Bit 1	Capture position 2 activated	
			Bit 2	Capture position 1 done	
			Bit 3	Capture position 2 done	
			Bit 4	Capture position 1 in process	
			Bit 5	Capture position 2 in process	
			Bit 6	Capture position on main encoder	
			Bit 7	Capture position on secondary encoder	
			Bit 8	Capture position on auxiliary encoder	
			Bit 9	Single Capture position	
			Bit 10	Multiple Capture position	
M60	SD1	All		Controller status 1	4
			Bit 0	The controller is in power on	
			Bit 1	This bit is set when the controller has been initialized once (first PWR.# = 1)	
			Bit 2	This bit is set when the controller has finished the homing process with success	
			Bit 3	Bit present, always 1	
			Bit 4	The motor is executing a trajectory (bit moving)	
			Bit 5	This bit is set when the motor is in the position/time window defined by K38 and K39	
			Bit 8	The controller is executing an internal sequence	
			Bit 9	This bit is set when the motor is in the speed/time window defined by K180 and KL181	
			Bit 10	The controller is in error mode	
			Bit 11	Trace busy flag is set during a register trace acquisition	
			Bit 12	Breakpoint for sequence debug	
			Bit 13	This bit is set during the homing process	
			Bit 15	The controller is executing thread depending setting K297	
			Bit 23	The controller is in warning mode	
M61	SD2	All		Controller status 2	4
			Bit 0	Sequence error label pending	
			Bit 1	The scale mapping is enabled	
			Bit 2	Position captured according to the digital input (refer to K182/K301). This bit is set when the conditions on the digital input allow the capture of the position. It is reset when 1 is written in K182	
			Bit 8	User bit 0, could be modified by trigger functions or by parameter K177	
			Bit 9	User bit 1, could be modified by trigger functions or by parameter K177	
			Bit 10	User bit 2, could be modified by trigger functions or by parameter K177	
			Bit 11	User bit 3, could be modified by trigger functions or by parameter K177	
			Bit 12	User bit 4, could be modified by trigger functions or by parameter K177	
			Bit 13	User bit 5, could be modified by trigger functions or by parameter K177	
			Bit 14	User bit 6, could be modified by trigger functions or by parameter K177	
			Bit 15	User bit 7, could be modified by trigger functions or by parameter K177	
MF61		ZxT		ZxT Ry position loop proportional gain value	1
MF62		ZxT		ZxT Ry position loop speed feedback gain value	1
M63	SDC	All		Controller status	4
			Bit 0	User bit 0, could be modified by trigger functions or by parameter K177	
			Bit 1	User bit 1, could be modified by trigger functions or by parameter K177	
			Bit 2	User bit 2, could be modified by trigger functions or by parameter K177	
			Bit 3	User bit 3, could be modified by trigger functions or by parameter K177	

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
			Bit 4	User bit 4, could be modified by trigger functions or by parameter K177	
			Bit 5	User bit 5, could be modified by trigger functions or by parameter K177	
			Bit 6	User bit 6, could be modified by trigger functions or by parameter K177	
			Bit 7	User bit 7, could be modified by trigger functions or by parameter K177	
			Bit 8	User bit 8, could be modified by trigger functions or by parameter K177	
			Bit 9	User bit 9, could be modified by trigger functions or by parameter K177	
			Bit 10	User bit 10, could be modified by trigger functions or by parameter K177	
			Bit 11	User bit 11, could be modified by trigger functions or by parameter K177	
			Bit 12	User bit 12, could be modified by trigger functions or by parameter K177	
			Bit 13	User bit 13, could be modified by trigger functions or by parameter K177	
			Bit 14	User bit 14, could be modified by trigger functions or by parameter K177	
			Bit 15	User bit 15, could be modified by trigger functions or by parameter K177	
			Bit 16	The controller is in power on	
			Bit 17	This bit is set when the motor is in the torque/time window	
			Bit 19	Bit present, always 1	
			Bit 20	The motor is executing a trajectory (bit moving)	
			Bit 21	This bit is set when the motor is in the position/time window defined by K38 and K39	
			Bit 22	This bit is set when the motor is in the speed/time window defined by K180 and KL181	
			Bit 23	The controller is in warning mode	
			Bit 24	The controller is executing an internal sequence	
			Bit 25	End stroke information for synchronized homing	
			Bit 26	The controller is in error mode	
			Bit 27	Trace busy flag is set during a register trace acquisition	
			Bit 28	Breakpoint for sequence debug	
			Bit 29	Index information for synchronized homing	
			Bit 30	Position captured according to the digital input (refer to K182/K301). This bit is set when the conditions on the digital input allow the capture of the position. It is reset when 1 is written in K182	
			Bit 31	The controller is executing thread depending setting K297	
M64	ERRCD	All		Gives the error code	4
MF64		ZxT		ZxT Ry position loop integrator gain value	1
M66	WARCD	All		Gives the warning code	4
MF67		All		I2t motor current value. When MF67 is greater than KF85, the controller generates an I2T error (M64=4)	4
M70	CTYP	All		Indicates the type of controller. Depth0: Product type (32 for AccurET Modular 400/600, 33 for AccurET Modular 48, 34 for AccurET Modular 300, 36 for AccurET VHP 48, 39 for AccurET VHP100, 42 for AccurET VHP 48 ZxT); depth1: Hardware version	2
M71		All		Gives the software boot version of the controller. A special ETEL procedure allows the conversion of this value in the software boot version (format is similar to M72)	1
M72	VER	All		Gives the firmware version of the controller. A special ETEL procedure allows the conversion of this value in the firmware version. Depth 0 firmware version in SDRAM, depth1 firmware version in Flash	2
M73	SER	All		Gives the serial number of the controller	1
M76		48, 300, 400		Optional board type	1
		0		No optional board	
		64		I/O board with 8 digital I/O and 4 analog I/O	
M82		All		Controller maximum current. M82/100 gives the maximum current in ampere	2
MF82	IMAXC	All		Controller maximum current. MF82/100 gives the maximum current in ampere	2
ML83		All		Gives the mask of the axes number present on the controller	2

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
MF84		All		I2 motor rms current limit value for I2t integration according to the speed (M11)	1
M85	ARTICLE	All		Gives the article number string. The 25 strings of the article number are read using 7 depths of M85. Each depth shows 4 strings (in ASCII)	2
M86		All		Active communication mode	1
			0	Manager mode TransnET	
M87		All		Gives the axis number	2
M88		All		Dip switch axis number	1
M89		All		Flash axis number	1
M90	CTEMP	All		Gives the temperature of the controller measured by a thermostat on the heat sink	4
M91	UBUS	All		Gives the DC power voltage (Vpower) [V/100]. M91/100 gives the tension in volt	4
M95		All		Shows the strings on the software display. The 16 strings of the controller display are read using the 4 depths of M95. Each depth shows 4 strings (in ASCII)	1
MF95		All but ZxT		Fitting coefficient of the successive phasing moves	1
M97		All		Mask of the sequence threads currently active	1
M101		All		Bit field indicating that an entry is waiting (=1)	1
			Bit 0	TransnET gate 0	
			Bit 1	TransnET gate 1	
			Bit 2	TransnET gate 2	
			Bit 3	TransnET gate 3	
			Bit 4	USB	
			Bit 8	TCP/IP	
M102		All but ZxT		Dual Encoder Feedback configuration	1
			Bit 0	Dual Encoder Feedback feature enabled	
			Bit 1	Mode 0: Permanent Dual Encoder Feedback enabled	
			Bit 2	Mode 1: Intermittent Dual Encoder Feedback enabled	
			Bit 4	Allow error management on auxiliary analog encoder 1Vpp and HSEI	
			Bit 5	Ignores errors from the secondary encoder	
M103		All but ZxT		Dual Encoder Feedback recovery status	1
M104		All but ZxT		Permanent Dual Encoder Feedback: Real payload position. LSL part of ML104.	1
ML104		All but ZxT		Permanent Dual Encoder Feedback: Real payload position.	1
M105		All but ZxT		Permanent Dual Encoder Feedback: Real motor position. LSL part of ML105	1
ML105		All but ZxT		Permanent Dual Encoder Feedback: Real motor position.	1
M106		All		Advanced filters status bit 7..0 : enable filter bit number, bit 15..8: number of enabled filters	3
ML106		All but ZxT		Real offset between motor and payload in Dual Encoder Feedback. For permanent mode, show the delta position between motor and payload when an Absolut encoder is present. For the intermittent mode, represent the delta position after the VALPOS command.	1
M109		All		Theoretical speed before adv filter depth 0	3
MF110		All but ZxT		Delta position between Motor and Payload	1
M111		All		Real speed before advanced filter (dsi)	3
MF112		All but ZxT		Dual Encoder Feedback: Motor tracking error	1
M114		All		Real acceleration before adv filter depth 5	1
ML114		All but ZxT		Intermittent Dual Encoder Feedback: Real motor position.	1
M115		All		Real acceleration after adv filter depth 5	1
ML115		All but ZxT		Intermittent Dual Encoder Feedback: Real payload position.	1
MF116		All		Current reference value defined by parameters K220 K221 K222 K223 & KF224	1

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
M117		All		Position reference value defined by parameters K220 K221 K222 K223 & KF224. LSL part of ML117	1
ML117		All		Position reference value defined by parameters K220 K221 K222 K223 & KF224	1
M118		All		Speed reference value defined by parameters K220 K221 K222 K223 & KF224. LSL part of ML118	1
ML118		All		Speed reference value defined by parameters K220 K221 K222 K223 & KF224	1
M119		All		Position reference value after trajectory filter in (dpi).	1
M120		All		Trace: Trace synchronization mode	1
M121		All		Trace: Register selection type (depth 0 for trace 0; depth 1 for trace 1,...)	1
			1	User variable register (X)	
			2	Parameter register (K)	
			3	Monitoring register (M)	
			7	Address register (A)	
			8	LKT register (L)	
			13	Common parameter register (C)	
			33	User variable register (XF)	
			34	Parameter register (KF)	
			35	Monitoring register (MF)	
			45	Common parameter register (CF)	
			65	User variable register (XL)	
			66	Parameter register (KL)	
			67	Monitoring register (ML)	
			77	Common parameter register (CL)	
			97	User variable register (XD)	
			98	Parameter register (KD)	
			99	Monitoring register (MD)	
			104	LKT register (LD)	
			109	Common parameter register (CD)	
MF121		All		Position loop acceleration feedforward gain value	1
M122		All		Trace: Register number and depth selection (bit#0-15 for register number, bit#16-23 for depth)	1
M123		All		Trace: Sampling time (period of 25us)	1
M124		All		Trace: Trigger edge	1
M125		All		Trace: trigger mode	1
			0	Immediate acquisition	
			1	Start of movement	
			2	End of movement	
			3	Trigger at a position defined by KL127	
			4	Trigger at a given value (defined in K127 or KF127, KL127, KD127) of the trace defined in K126	
			5	Trigger never starts. An external freeze command in synchronized mode can stop the acquisition	
			6	Trigger at a given value (defined in K127 or KF127, KL127, KD127) of a register defined in K126	
			7	Trigger on a bit field state (K127 or KL127:0: bit to be low, K127 or KL127:1: bit to be high) of a register defined in K126	
			8	Trigger on a bit field rising or falling edge (K127 or KL127:0: bit on rising edge, K127 or KL127:1: bit on falling edge) of a register defined in K126	
			9	Continuous acquisition mode - No trigger	
M126		All		Trace: Definition of the trigger parameter according to the trigger mode	1
M127		All		Trace: Trigger level according to the trigger mode	1

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
MF127		All		Trace: Trigger level according to the trigger mode	1
ML127		All		Trace: Trigger level according to the trigger mode	1
M128		All		Trace: Number of acquired points for the traces	1
M129		All		Trace: Pre or post trigger value	1
M131		All		Trace: Acquisition state	1
			0	Acquisition state: Initialization 1	
			1	Acquisition state: Initialization 2	
			2	Acquisition state: Wait for buffer synchronization	
			3	Acquisition state: Wait for trigger	
			4	Acquisition state: Trigger condition reached	
			5	Acquisition state: End of acquisition	
M134		All		Trace: Freeze time of the measure	1
ML134		All		Trace: Freeze time of the measure	1
M135		All		Trace: Number of the T register with the trigger condition	1
M136		All		Trace: Number of available points	1
M138		All		Maximum tracking error during WTW command and the setup of bit#5 of M60 (motor in the windows defined by K38 and K39)	1
M139		All		Maximum tracking error during in-window time and the setup of bit#5 of M60 (motor in the windows defined by K38 and K39)	1
M140		All		Mask for fuse control	2
			Bit 0	The encoders fuse is broken. The encoders are no more powered	
			Bit 1	The DOUTs fuse is broken. The DOUTs are no more powered	
M141		All		TransnET error groups status	1
			Bit 0	Error propagation group 1	
			Bit 1	Error propagation group 2	
			Bit 2	Error propagation group 3	
			Bit 3	Error propagation group 4	
M160		All but ZxT		Captured position conditions. Bit 0-4 DIN rising condition, Bit 5-10 FDIN rising condition, Bit 11-15 DIN falling condition, Bit 16-21 FDIN falling condition	2
			Bit 0	Rising Edge on DIN1	
			Bit 1	Rising Edge on DIN2	
			Bit 3	Rising Edge on DIN9	
			Bit 4	Rising Edge on DIN10	
			Bit 5	Rising Edge on FDIN1	
			Bit 6	Rising Edge on FDIN2	
			Bit 7	Rising Edge on FDIN3	
			Bit 8	Rising Edge on FDIN4	
			Bit 9	Rising Edge on FDIN5	
			Bit 10	Rising Edge on FDIN6	
			Bit 11	Falling Edge on DIN1	
			Bit 12	Falling Edge on DIN2	
			Bit 14	Falling Edge on DIN9	
			Bit 15	Falling Edge on DIN10	
			Bit 16	Falling Edge on FDIN1	
			Bit 17	Falling Edge on FDIN2	
			Bit 18	Falling Edge on FDIN3	
			Bit 19	Falling Edge on FDIN4	
			Bit 20	Falling Edge on FDIN5	

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
			Bit 21	Falling Edge on FDIN6	
M161	All but ZxT			Captured position 2 conditions. Bit 0-4 DIN rising condition, Bit 5-10 FDIN rising condition, Bit 11-15 DIN falling condition, Bit 16-21 FDIN falling condition	2
			Bit 0	Rising Edge on DIN1	
			Bit 1	Rising Edge on DIN2	
			Bit 3	Rising Edge on DIN9	
			Bit 4	Rising Edge on DIN10	
			Bit 5	Rising Edge on FDIN1	
			Bit 6	Rising Edge on FDIN2	
			Bit 7	Rising Edge on FDIN3	
			Bit 8	Rising Edge on FDIN4	
			Bit 9	Rising Edge on FDIN5	
			Bit 10	Rising Edge on FDIN6	
			Bit 11	Falling Edge on DIN1	
			Bit 12	Falling Edge on DIN2	
			Bit 14	Falling Edge on DIN9	
			Bit 15	Falling Edge on DIN10	
			Bit 16	Falling Edge on FDIN1	
			Bit 17	Falling Edge on FDIN2	
			Bit 18	Falling Edge on FDIN3	
			Bit 19	Falling Edge on FDIN4	
			Bit 20	Falling Edge on FDIN5	
			Bit 21	Falling Edge on FDIN6	
M163		All		State motion for Advanced Gain Scheduling	1
ML165		All but ZxT		Captured position on main encoder. Takes SET command and mapping correction into account	2
M166	WARNFLG	All		AccurET warning flags	4
			Bit 1	AccurET Warning bit#1: I2t motor	
			Bit 2	AccurET Warning bit#2: Overtemperature	
			Bit 3	AccurET Warning bit#3: Inrush	
			Bit 4	AccurET Warning bit#4: Overvoltage	
			Bit 5	AccurET Warning bit#5: Encoder amplitude	
			Bit 6	AccurET Warning bit#6: Tracking	
			Bit 7	AccurET Warning bit#7: Position synchro	
			Bit 8	AccurET Warning bit#8: Hall effect	
			Bit 9	AccurET Warning bit#9: PWM synchro	
			Bit 10	AccurET Warning bit#10: Undervoltage	
			Bit 11	AccurET Warning bit#11: I2t drive	
			Bit 12	AccurET Warning bit#12: Power Relay	
			Bit 13	AccurET Warning bit#13: Init small mve	
			Bit 15	AccurET Warning bit#15: PWM update	
			Bit 17	AccurET Warning bit#17: EnDat encoder	
			Bit 18	AccurET Warning bit#18: HSEI encoder	
			Bit 19	AccurET Warning bit#19: Sequence debugger; breakpoints present.	
			Bit 28	AccurET Warning bit#28: Present when a command MVE RMVE STA STI is sent when axis is moving (Concatenate movement)	
			Bit 30	AccurET Warning bit#30: Corruption of data with continuous trace	
ML166		All but ZxT		Captured position 2 on main encoder. Takes SET command and mapping correction into account	2

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
MF167		All		I2t controller current value. When MF167 is greater than controller's value, the controller generates an I2T error (M64=11)	4
ML167		All but ZxT		Captured position on secondary encoder.	2
ML168		All but ZxT		Captured position on auxiliary encoder	2
M170		All		Gives the state of the digital outputs of the optional board. Takes XDOUT (C6) into account	1
M171		All		Gives the state of the digital outputs of the controller at the beginning of the PLTI	1
M172		All		Gives the state of the fast outputs of the controller at the beginning of the PLTI	1
M173		All		Real state of the optional board's analog outputs	1
M174		All VHP		Real state of the analog outputs	1
M180		All		I2 sum PLTI counter	1
ML181		All		I2 sum	1
M201		All		Time quota for thread 1 (depth 1) and thread 2 (depth 2)	1
M202		All		Time used by thread 1 (depth 1) and thread 2 (depth 2)	1
M203		All but ZxT		Gives the MAC address	1
M204	MIPADD	All but ZxT		Gives the IP address	1
M205	All but ZxT		Ethernet status		1
			Bit 0 Ethernet communication ready		
			Bit 1 Ethernet DHCP running		
			Bit 2 Ethernet communication ok (DHCP is done & ok)		
			Bit 3 Ethernet hardware network connection ok		
			Bit 6 Ethernet network speed (0: 10 Mbps, 1: 100Mps)		
			Bit 7 Ethernet network full duplex mode (0: half, 1: full)		
			Bit 8 Ethernet CRC error bit 0		
			Bit 9 Ethernet CRC error bit 1		
			Bit 10 Ethernet CRC error bit 2		
			Bit 11 Ethernet CRC error bit 3		
			Bit 12 Etnd Status		
			Bit 13 Multicast IP messages received		
			Bit 14 Good IP address messages received		
			Bit 15 Allow transmission records 34H		
			Bit 16 TCP port status bit 0		
			Bit 17 TCP port status bit 1		
			Bit 18 TCP port status bit 2		
			Bit 19 TCP port status bit 3		
			Bit 20 Warning, sent messages has been segmented		
			Bit 21 Warning, a received message was segmented		
			Bit 22 Warning, DHCP failed		
			Bit 23 Warning, message bigger than permitted received		
			Bit 24 Warning, IP checksum is wrong		
			Bit 25 Warning, message with unrecognized protocol received		
			Bit 26 Warning, ICMP message asking for fragmentation received		
			Bit 27 Warning, ICMP message "host unreachable" received		
			Bit 28 Warning, unrecognized ICMP message		
			Bit 29 Warning, DHCP server gives an already attributed address		
M206		All but ZxT		Number of TCP messages: depth 0: received, depth 1: transmitted, depth 2: received more than once, depth 3: transmitted more than once	1
M207		All but ZxT		Number of UDP messages: depth 0: received, depth 1: transmitted, depth 2: lost	1

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
M208		All but ZxT		IP address of the TCP host	1
M209		All but ZxT		TCP port of the host	1
M211		All		Absolute maximum speed after CAM calculation. LSL part of ML211	1
ML211		All		Absolute maximum speed after CAM calculation	1
M212		All		Absolute max. acc./deceleration after CAM calculation. LSL part of ML212	1
ML212		All		Absolute max. acc./deceleration after CAM calculation	1
M213		All		Jerk time applied after CAM calculation	3
M228		All		Reference time counter in unit of 25us. This counter runs freely and is synchronized with UltimET when TransnET is connected	1
ML228		All		Reference time counter in unit of 25us. This counter runs freely and is synchronized with UltimET when TransnET is connected	1
MF230		All		Theoretical current Iq after advanced filters	1
MF231		All		Theoretical current Iq ffwd part	1
MF232		All		Theoretical current Iq with ffwd and cogging part	1
MF233		All		Theoretical current Iq before KF60 limitation	1
MF234		All		Theoretical current Iq before Kt compensation	1
M239		All		Encoder period in [nm] for linear encoder or number of period for rotary encoder (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1
M240		All		Movement type conversion. 0: linear movement, 1: rotary movement (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1
M241		All		Encoder interpolation factor (depth 0: main, depth 1: secondary, depth 2: auxiliary)	1
M243	CLTI	All		Controller current loop time factor (CLTI). M243 = time in second	1
M244	PLTI	All		Controller position loop time factor (PLTI). M244 = time in second	1
M245	MLTI	All		Controller manager loop time factor (MLTI). M245 = time in second	1
M247		All		Master/Slave Compensation pointer	1
M248		All		Master/Slave Compensation pointer offset	1
MF249	ZxT			Theoretical current Iq for current loop after KF60 limitation (Logical axes)	1
MF250		All		Cogging compensation value	1
M251		All		Real position after Stage Mapping (dpi), before Encoder Stretch. LSL part of ML251	1
ML251		All		Real position after Stage Mapping (dpi), before Encoder Stretch	1
M252		All		Encoder Stretch offset value (dpi). LSL part of ML252	1
ML252		All		Encoder Stretch offset value (dpi)	1
MF253		All		Encoder Stretch factor value	1
M254		All		Encoder Stretch state	1
M260		All		Status of SPI communication	1
			Bit 0	SPI Buffer Out full	
			Bit 1	SPI Data received	
			Bit 2..4	SPI Reserved	
			Bit 5..11	SPI Current position	
			Bit 12	SPI CRC error	
M261		All		Data received from SPI communication	1
M262		All		SPI CRC error counter	1
M265		All		Gantry parameter error. Depth 0: parameter number, depth 1: parameter type, depth 2: parameter depth	1
M266		All		Gantry parameter error. Depth 0: second error MLTI counter, depth 1: first error MLTI counter	1
M267		All		Gantry parameter first error. Depth 0: parameter number, depth 1: parameter type, depth 2: parameter depth	1
M274		All		Stage protection current mode. 0: stage protection disabled 1: stage protection enabled brake with a step on the speed reference (step to 0) 2: stage protection enabled brake with a ramp defined by KL206:1 on the speed	1

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
M282		All		Registers or sequence version (depth 0 to 2: not used, depth 3 to 9 as for SAV command)	1
M289		All		Gantry current Level	1
M330		All		Stage error mapping state	1
			Bit 4	Current source positions are in the mapping area	
			Bit 5	Stage error mapping configuration is local to this AccurET controller	
			Bit 6	Stage error mapping correction is enabled	
M340		All		Fast Trigger: Published position. LSL part of ML340	1
ML340		All		Fast Trigger: Published position. Real (if K336= 3) or Theoretical (if K336= 4) position used for the trigger. This position takes in account different compensations	1
M345		All but ZxT		Fast Trigger: Pulse Generator pulse count (depth 0: Trigger Pulse Generator 1; depth 1: Trigger Pulse Generator 2)	2
M346		All but ZxT		Fast Trigger: Number treated event	2
M349		All but ZxT		Fast Trigger: Number of free event in EL memory	1
M350		All but ZxT		Fast Trigger: First event free	1
M351		All but ZxT		Fast Trigger: Number of event per group	1
M363		All		DOUT state for motor brake function (bit 0-15: state 1 (DOUT1 bit#0 DOUT2 bit#1), bit 16-31: state 0 (DOUT1 bit#16 DOUT2 bit#17))	1
M370		300, 400		Power Sag Detection timer	1
M390	All but ZxT			Status for Force Control mode	1
			Bit 0	The force control mode is active	
			Bit 1	Check the position for the switch in force control mode	
			Bit 3	Force control on mode 0 is selected (mode 0 and 1 command SETFC)	
			Bit 4	Force control on mode 1 is selected (mode 1 command SETFC)	
			Bit 6	Force control on mode 1 is active	
			Bit 12	Integrator limit reached	
			Bit 14	Collapse detected	
			Bit 17	In windows Force control	
			Bit 18	Check rebound	
M391		All but ZxT		Position reference added in Force regulation mode	1
MF391		All but ZxT		Current reference added in Force regulation mode	1
M392		All but ZxT		Mode selector for Force regulator	1
MF392		All but ZxT		Force reference for force regulator	1
M395		All but ZxT		Duration of the window in Force Control mode	1
MF395		All but ZxT		Force range of the window in Force Control mode	1
ML396		All but ZxT		Upper limit threshold after the touchdown applied on the measured position ML1	1
MF397		All but ZxT		Estimated force after filtering and smoothing	1
MF398		All but ZxT		Estimated current after filtering and smoothing	1
MF399		All but ZxT		Estimated force after filtering before smoothing	1
MF400		All but ZxT		Estimated current after filtering before smoothing	1
MF401		All but ZxT		Reference current for force regulator	1
MF402		All but ZxT		Error current for force regulator	1
MF406		All but ZxT		Current feedback before smoothing and filtering	1
MF407		All but ZxT		Filtered error current for force regulator	1
MF409		All but ZxT		Smoothing factor for the estimated force	1
M417		All		Sequence stack size in 32-bit words	1
M418		All		Sequence stack free space in 32-bit words	1
ML434		All		Minimum software position limit (upi)	1

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
ML435		All		Maximum software position limit (upi)	1
M443		All		Number of possible RTV write slots (ASW)	1
M447		All		Number of possible RTV read slots (ASR)	1
M448		All		Used RTV slots (by ASW)	1
M449		All		Number of written RTV (by ASW command). By default M63 is already put on RTV and can be read through UltimET (*M520:axis_number)	1
M450		All		RTV monitoring M450	2
MF450		All		RTV monitoring MF450	2
ML450		All		RTV monitoring ML450	2
M451		All		RTV monitoring M451	2
MF451		All		RTV monitoring MF451	2
ML451		All		RTV monitoring ML451	2
M452		All		RTV monitoring M452	2
MF452		All		RTV monitoring MF452	2
ML452		All		RTV monitoring ML452	2
M453		All		RTV monitoring M453	2
MF453		All		RTV monitoring MF453	2
ML453		All		RTV monitoring ML453	2
M454		All		RTV monitoring M454	2
MF454		All		RTV monitoring MF454	2
ML454		All		RTV monitoring ML454	2
M455		All		RTV monitoring M455	2
MF455		All		RTV monitoring MF455	2
ML455		All		RTV monitoring ML455	2
M456		All		RTV monitoring M456	2
MF456		All		RTV monitoring MF456	2
ML456		All		RTV monitoring ML456	2
M457		All		RTV monitoring M457	2
MF457		All		RTV monitoring MF457	2
ML457		All		RTV monitoring ML457	2
M458		All		RTV monitoring M458	2
MF458		All		RTV monitoring MF458	2
M459		All		RTV monitoring M459	2
MF459		All		RTV monitoring MF459	2
M460		All		RTV monitoring M460	2
MF460		All		RTV monitoring MF460	2
M461		All		RTV monitoring M461	2
MF461		All		RTV monitoring MF461	2
M462		All		RTV monitoring M462	2
MF462		All		RTV monitoring MF462	2
M463		All		RTV monitoring M463	2
MF463		All		RTV monitoring MF463	2
M464		All		RTV monitoring M464	2
MF464		All		RTV monitoring MF464	2
M465		All		RTV monitoring M465	2
MF465		All		RTV monitoring MF465	2
M470		All		Maximum downloadable sharc instructions for a compiled sequence	1

M	Alias	Product	Values or bit#	Description	Gantry feedback (*)
M471		All		Maximum downloadable data longs for a compiled sequence	1
M472		All		Maximum downloadable source characters for a compiled sequence	1
M475		All		Effectively downloaded sharc instructions in the last compiled sequence	1
M476		All		Effectively downloaded data longs in the last compiled sequence	1
M477		All		Effectively downloaded source characters in the last compiled sequence	1
M480		ZxT		ZxT Raw local encoders value	1
M481		ZxT		ZxT latched reference marks	1
M482		ZxT		ZxT reference mark homing status	1
M483		ZxT		ZxT ry Advanced filters status bit 7..0: enable filter bit number, bit 15..8: number of enabled filters	3
M483		All but ZxT		Fctrl mode2 Advanced filters status bit 7..0: enable filter bit number, bit 15..8: number of enabled filters	3
MF486	MF31_TILT	ZxT		ZxT Rx, Ry PID controllers output before feedforward components	1
M490		ZxT		ZxT MIMO factor	1
M491		ZxT		ZxT global Z tracking error	3
MF496		ZxT		ZxT Speed error (filtered)	1
MF497		ZxT		ZxT Shows the correction value added to the current setpoint (right before MF233). This value is affected by CF490, the moving mass and KL250, the lever arm distance. The feedforward compensation can be disabled by setting CF490 to a null value.	1
MF500		ZxT		Adjustment factor used on MIMO controllers for logical axes	1
MF510		ZxT		ZxT Feedforward mass iso conversion	1

20. Warnings reference list

M66	Displayed message	Description	Solutions	Product
1	W I2T MOTOR	This warning occurs when MF67 is greater than KF85/2	General case: - See error "I2T ERR MOTOR" (error num 4).	All
2	W OVER TEMPERAT	This warning occurs when the temperature of the controller is greater than 60 C	General case: - See error "OVER TEMPERAT" (error num 5).	VHP 100
2	W OVER TEMPERAT	This warning occurs when the temperature of the controller is greater than 69 C	General case: - See error "OVER TEMPERAT" (error num 5).	400/600
2	W OVER TEMPERAT	This warning occurs when the temperature of the controller is greater than 70 C	General case: - See error "OVER TEMPERAT" (error num 5).	300, VHP 48, ZxT
2	W OVER TEMPERAT	This warning occurs when the temperature of the controller is greater than 75 C	General case: - See error "OVER TEMPERAT" (error num 5).	48
3	W INRUSH P.SUPPLY	Power supply inrush warning	General case: - See error "INRUSH P:SUPPLY" (error num 7).	All
4	W OVER VOLTAGE	This warning occurs when the power voltage (Vpower) in volt is greater than K148/100	General case: - See error "OVER VOLTAGE" (error num 6).	All
5	ENCODER AMPLITUD	This warning occurs when the amplitude of the main analog encoder signals are too small (2x the error)	General case: - See error "ENCODER AMPLITUD" (error num 20).	All
6	TRACKING WARNING	This warning occurs when the tracking error is greater than K30/2	General case: - See error "TRACKING ERROR" (error num 23).	All
8	W DIGIT. HALL	Wrong value coming from the digital Hall sensor	General case: - Wrong value coming from the digital Hall sensor	All
9	WUCONTROL LSYNCHRO	Lost synchronization between TransnET and controller interrupts	General case: - Lost synchronization between TransnET and controller interrupts.	All
10	W UNDER VOLTAGE	This warning occurs when the power voltage (Vpower) in volt is lower than K146/100	General case: - See error "UNDER VOLTAGE" (error num 9).	All
11	W I2T CTRL	This warning occurs when MF167 is greater than MF285/2	General case: - See error "I2T ERR MOTOR" (error num 4).	All
12	W POWER RELAY	This warning occurs when the control signal coming from the power relay is at 0V and the controller is in Power Off	General case: - See error "POWER RELAY ERR" (error num 132).	300, 400/ 600
13	WPHASING FAILURE	This warning occurs during the automatic repetition of the initialization, when K90=7	General case: - This warning occurs during the automatic repetition of the initialization, when K90=7	All
15	W AMPLI UPDATE	Warning indicating that the current references have been calculated too late. As a consequence, the internal dead time in position loop is not constant	General case: - Internal warning: indicating that the PWM have been calculated too late. As a consequence, the internal dead time in current loop is not constant	VHP 100
15	W PWM UPDATE	Warning indicating that the PWM have been calculated too late. As a consequence, the internal dead time in current loop is not constant	General case: - Internal warning: indicating that the PWM have been calculated too late. As a consequence, the internal dead time in current loop is not constant	300, 400/ 600, 48, VHP 48, ZxT
17	W ENDAT ENCODER	Warning relative to EnDat encoder	General case: - Indicating that EnDat encoder is in warning.	All
18	W HSEI ENCODER	Internal warning indicating that HSEI encoder is in warning	General case: - Internal warning related to HSEI encoder	VHP 100, VHP 48, ZxT
19	W SEQUEN BRKPOINT	Warning indicating that breakpoints are set on the sequence	General case: - Sequence debugger: Breakpoints present	All
24	TRIGGER MISSED	Fast Trigger: detect a missed event	General case: - Fast Trigger: detect a missed event	All
25	W TRIG TOO LATE	Internal warning indicating that the position for the fast trigger has been written too late on the FPGA	General case: - Internal warning indicating that the position for the fast trigger has been written too late on the FPGA	All
26	SEC ENCO AMPLITUD	This warning occurs when the amplitude of the secondary analog encoder signals are too small (2x the error)	General case: - See error "SEC ENCO AMPLITUD" (error num 22).	All
27	AUX ENCO AMPLITUD	This warning occurs when the amplitude of the auxiliary analog encoder signals are too small (2x the error)	General case: - See error "AUX ENCO AMPLITUD" (error num 32).	All

M66	Displayed message	Description	Solutions	Product
28	JERKTIME IGNORED	Present when a command MVE RMVE STA STI is sent when axis is moving (Concatenate movement)	General case: - Command MVE with jerk time in concatenated mode when axis is moving - Command RMVE with jerk time in concatenated mode when axis is moving	All
29	TRIGGER CONTINUO	No more event programmed in EL memory	General case: - Too slow communication between PC and controller. - Speed too fast.	All
30	UPLOAD TOO SLOW	Corruption of data with continuous trace	General case: - The Traces upload is not fast enough and data was corrupted. Increase the upload rate or reduce sampling frequency.	All

21. Errors reference list

The points to check in the **errors reference list** are indicated as follows (possible reasons for the error are listed, but the list is non-exhaustive):

HW Res = x

Brk = OFF/ON

Protection error type

It is recommended or compulsory to reset the error by hardware (RSD command or switch off/on)

OFF or ON means that this error deactivates or activates the dynamic braking (when used).

Error type number (refer to §9.)

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
2	OVER CURRENT1	The current measured in phase 1 is greater than KF83 parameter	Bad regulation: - The difference between the controller maximum allowed current (KF60) and the overcurrent limit (KF83) is too small. - The current loop is oscillating (KF80, KF81). Initialization problem: - If the phasing mode is a phasing by pulse, the pulse current level is too high. - If the motor has a low inductance, either the DC bus or the voltage ratio parameter (K98) have to be decreased.		OFF	1	All
3	OVER CURRENT3	The current measured in phase 3 is greater than KF83 parameter	Bad regulation: - The difference between the controller maximum allowed current (KF60) and the overcurrent limit (KF83) is too small. - The current loop is oscillating (KF80, KF81). Initialization problem: - If the phasing mode is a phasing by pulse, the pulse current level is too high. - If the motor has a low inductance, either the DC bus or the voltage ratio parameter (K98) have to be decreased.		OFF	1	All
4	I2T ERR MOTOR	This occurs when MF67 becomes greater than KF85. This is a power protection of the motor	If an error occurs during the phasing: - Phasing in constant current has been made against the mechanical end stop. - Phasing in constant current, the KF92 is set too high. - Bad phasing setting. Check repeatability of your phasing and if needed change the setting of your phasing. - If a digital hall effect sensor is used, the wiring may be done incorrectly. - If an EnDat encoder is used: sin/cos clock and EnDat clock are counting in the opposite direction (bad wiring). If an error occurs during the homing or just after it: - Mechanical end stop detection parameters are not adjusted correctly. - Wrong encoder period or wrong magnetic period. - If an encoder with multiple reference marks is used: The coding of the multiple reference mark is incorrect. - The number of motor phases or its wiring are incorrect. - Bad fine phase adjustment.		OFF	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
			<p>General case:</p> <ul style="list-style-type: none"> - The regulation loops are not properly set. - The I2t parameters are set with a too low value. - An obstacle may be on the way of the motor. - The cycle is too restrictive for the possibilities of the system (a wait time may be needed in order to allow the current integration to decrease at zero). - The mechanical stress has been increased, due to a degradation of the system components. - The position target is out of the allowed stroke. - Too much friction force or too much load. - A mechanical break is still active. 				
5	OVER TEMPERAT	The thermostat mounted on the controller detect an over temperature	<p>General case:</p> <ul style="list-style-type: none"> - The electronic needs more air cooling. - Temperature sensor is damaged. 	ON	1	All	
6	OVER VOLTAGE	The power voltage (Vpower) in volt is greater than K149/100	<p>General case:</p> <ul style="list-style-type: none"> - The protection parameter is set with a too low value. - Unstable network. <p>Error appends during the deceleration:</p> <ul style="list-style-type: none"> - An external regeneration resistor is needed. - Speed and deceleration parameters are defined with too high values. - Too much moving mass. 	OFF	1	All	
7	INRUSH P.SUPPLY	Power supply inrush or power voltage too low when motor power is enabled	<p>General case:</p> <ul style="list-style-type: none"> - The instantaneous current during a move cycle is too high. - Either a tri-phases transformer or a more powerful transformer is needed. - The voltage of the DC bus has to be increased. - No voltage on the connector P01 of the AccurET power supply. - Dynamic braking is working before switching on the power voltage. 	ON	1	All	
9	UNDER VOLTAGE	The power voltage (Vpower) in volt is lower than K147/100	<p>General case:</p> <ul style="list-style-type: none"> - There is a transformer before the power connector, which does not provide enough power. - The instantaneous current during a move cycle is too high. - A tri-phases transformer is needed. - The voltage of the DC bus has to be increased. - The protection parameter is set with a too high value K147. 	ON	1	All	
10	OFFSET CURERROR	Error in offset of current measurement. The offset measured is greater than 8% of the maximum current of the controller (M82)	<p>General case:</p> <ul style="list-style-type: none"> - If this error appears each time the controller is switched on, please contact ETEL. 	x	OFF	2	All
11	I2T ERR CTRL	This occurs when MF167 becomes greater than MF285. This is a power protection of the controller	<p>General case:</p> <ul style="list-style-type: none"> - See error 4 I2T ERROR MOTOR. 	OFF	1	All	
12	POWER CHK 3V	Error on Power Supply 3V	<p>General case:</p> <ul style="list-style-type: none"> - Please contact your provider. 	x	ON	1	All
13	SENSOR TEMP ERR	The temperature given by the sensor is not correct	<p>General case:</p> <ul style="list-style-type: none"> - Please contact your provider. 	x	OFF	1	All
14	DOUT FUSE KO	The fuse for the DOUT is broken	<p>General case:</p> <ul style="list-style-type: none"> - Please contact your provider. 	x	ON	1	300, 400/ 600, 48, VHP 48, ZxT

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
15	ERR MON VOLTAGE	The voltage and temperature cannot be read	General case: - Please contact your provider.	x	OFF	2	300, 400/600, 48, VHP 48, ZxT
16	ENDAT OVERFLOW	The calculated EnDat position is too big. Decrease K77:0 parameter	General case: - The encoder interpolation factor is too high. Please decrease this parameter (K77). - The current position of the EnDat encoder is too high. Please contact your provider to know how to proceed.	x	ON	1	All
17	ENDAT POS LOST	Zero position of the EnDat encoder not found	General case: - EnDat encoder not well connected. - The encoder wiring is not correct. - Check the corresponding controller fuse.	x	ON	2	All
18	ENDAT BAD CRC	Error on CRC from the EnDat	General case: - Problem of communication between the controller and the EnDat encoder. Please check if the shielding or / and the wiring are properly done. - The encoder cable is too long, the cable inductance is too big (see encoder data sheet).	x	ON	2	All
19	ENDAT POS ERR	The absolute position of EnDat is wrong	General case: - EnDat encoder not well connected. - The wiring is not correct.	x	ON	2	All
20	ENCODER AMPLITUD	The amplitudes of the main analog encoder signals are too small	General case: - The encoder's head is not correctly fixed. - A fuse may be broken (F4 in AccurET). - The speed during a movement may be too high (The signal frequency is too high (>500Khz for the analog encoder input)). - Unconnected or broken cable. - The encoder's head or/and the scale are dirty.	x	ON	2	All
21	ENCODER POS LOST	The main encoder has lost the position acquisition, the frequency is too high	General case: - The shielding of the encoder is not good, there is some noise on the cable. - The shielding of the I/O and motor cables have to be checked. - See also error 20 ENCODER AMPLITUD.	x	ON	2	All
22	SEC ENCO AMPLITUD	The amplitudes of the secondary analog encoder signals are too small	General case: - See error 20 ENCODER AMPLITUD.	x	ON	2	All
23	TRACKING ERROR	The tracking position error is greater than K30	General case: - The regulation loops are not properly set or the feed-forward parameters are not well set. - The tracking limit may be set with a too small value. - An obstacle may be on the way of the motor. - The cycle is too restrictive for the possibilities of the system (a waiting time may be needed in order to allow the current integration to decrease at zero). - The mechanical stress has been increased, due to a degradation of the system components. - The position target is out of the allowed stroke. - Too much friction force or too much load. - A mechanical break is still active. If an error occurs during the phasing: - Phasing in constant current has been made against the mechanical end stop. - Bad phasing. - If a digital hall effect sensor is used, the wiring may be done incorrectly. - If an EnDat encoder is used: sin/cos clock and EnDat clock are counting in the opposite direction (bad wiring).	ON	1		All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
			If an error occurs during the homing or just after it: - Mechanical end stop detection parameters are not adjusted correctly. - Wrong encoder period or wrong magnetic period. - If an encoder with multiple reference marks is used: The coding of the multiple reference mark is incorrect. - The number of motor phases or its wiring are incorrectly. - Bad fine phase adjustment.				
24	OVER SPEED	The speed is greater than K31	General case: - The regulation loops are not properly set. - The speed limit may be set with a too small value. If an error occurs during the phasing: - Phasing in constant current has been made against the mechanical end stop. - Bad phasing. - If a digital hall effect sensor is used, the wiring may be done incorrectly. - If an EnDat encoder is used: sin/cos clock and EnDat clock are counting in the opposite direction (bad wiring). If an error occurs during the homing or just after it: - Mechanical end stop detection parameters are not adjusted correctly. - Wrong encoder period or wrong magnetic period. - If an encoder with multiple reference marks is used: The coding of the multiple reference mark is incorrect. - The number of motor phases or its wiring are incorrectly. - Bad fine phase adjustment. If an error occurs during Force Control mode: - The regulation Force Control loops are not properly set. - The speed limit(K31:1) may be set with a too small value.	ON	1	All	
25	SEC ENCO POS LOST	The secondary encoder has lost the position acquisition, the frequency is too high	General case: - The shielding of the encoder is not good, there is some noise on the cable. - The shielding of the I/O and motor cables have to be checked. - See also error 20 ENCODER AMPLITUD.	x	ON	2	All
26	POWER OFF/ON	Error when the bridge is powered ON as it is forbidden by a DIN specified by register KL305	General case: - A command "PWR" is given to the controller when the motor power up protection has not been activated KL305.		ON	1	All
27	ENDAT TIMEOUT	Time-out during the communication with the EnDat encoder	General case: - Problem of communication between the controller and the EnDat encoder. Please check if the shielding or / and the wiring are properly done. - The encoder cable is too long, the cable inductance is too big (see encoder data sheet). - Incorrect assembly of the scale.	x	ON	2	All
28	TORQUE MODE ERR	Command HLT or HLB received in torque/force mode	General case: - HLT and HLB cannot be send when axis is in force mode.		ON	1	All
29	MOTOR OVERTEMP	The temperature of the motor is too high, measured by temperature sensor connected to digital input according to parameter K304	General case: - The defined cycle time does not fit the capability of the motor. - The surface of dissipation connected mechanically to the motor is not sufficient. - The cable of the thermostat is broken or unconnected.	OFF	1	All	
30	SWITCH LIMIT	The controller generates this error when the controller reaches a limit switch during a movement (except during IND and SLS), if bit #1 of K33 is set	General case: - The movement defined by a sequence or through any communication bus is not correct, exceed the limits defined by the limit switches. - A limit switch is not present. - The cables of the limit switches are broken or unconnected.		ON	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
31	ENDAT ENCO ERR	EnDat encoder is in error	General case: - Problem of communication between the controller and the EnDat encoder. Please check if the shielding or / and the wiring are properly done. - The encoder cable is too long, the cable inductance is too big (see encoder data sheet). - Incorrect assembly of the scale.	x	ON	2	All
32	AUX ENCO AMPLITUD	The amplitudes of the auxiliary analog encoder signals are too small	General case: - The encoder's head is not correctly fixed. - A fuse may be broken (F4 in AccurET). - The speed during a movement may be too high (The signal frequency is too high (>500Khz for the analog encoder input)). - Unconnected or broken cable. - The encoder's head or/and the scale are dirty.	x	ON	2	All
33	AUX ENCO POS LOST	The auxiliary encoder has lost the position acquisition, the frequency is too high	General case: - The shielding of the encoder is not good, there is some noise on the cable. - The shielding of the I/O and motor cables have to be checked. - See also error 20 ENCODER AMPLITUD.	x	ON	2	All
34	ENCODER FUSE KO	The fuse for the HSEI encoder is broken	General case: - Check fuse.	x	ON	2	VHP 48, ZxT
35	ENCODER FUSE KO	The fuse for the encoder is broken	General case: - The fuse of the encoder may be broken when the encoder to the controller is disconnected, without switching off completely the controller.	x	ON	2	All
36	LABEL ERROR	Error when the label is not present	General case: - Check if sequence is present.		ON	1	All
37	ERR GAIN SCHEDUL	This error occurs when the setting (Kx226) for the advance gain scheduling is incorrect.	General case: - Check Kx226.		ON	1	All
38	REGISTER NUM ERR	This error occurs when the register number or sub-index is greater than its maximum value	General case: - Check with the manual the maximum number of each register type.		ON	1	All
39	ERR CFG XDOUT	This error occurs when several actions are selected on a XDOUT	General case: - By using on ComET the IO control tool, you can see if a XDOUT is reserved for any special feature. Or you can check following registers: - C358 output managed by RTV		ON	1	300, 400/ 600, 48
40	K79 BAD VALUE	This error occurs when K79 has a wrong value	General case: - Check the encoder selection (K79 parameter).	x	ON	1	All
41	K89 BAD VALUE	This error occurs when K89 has a wrong value	General case: - Check the motor phases configuration (K89 parameter).	x	ON	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
42	ERR CFG FDOUT	This error occurs when several actions are selected on a FDOUT	<p>General case:</p> <ul style="list-style-type: none"> - By using on ComET the IO control tool, you can see if a FDOUT is reserved for any special feature. Or you can check following registers: - C353:1 TransnET mask - C354:1 MLTI mask - C355:1 PLTI mask - C359 Trigger - C360:1 MOSI/MISO SPI - C361:1 SCLK SPI - C362:1 Chip Select SPI - K320:1 to K335:1 mask combination 1to 16 - K350:1 SIN mask - K351:1 COS mask - K352:1 INDEX mask - K357:1 error mask - K358:1 sti movement start - C358 output managed by RTV 		ON	1	All
43	BAD VER FIRMWARE	Setting incompatible with firmware or hardware	<p>General case:</p> <ul style="list-style-type: none"> - C1 = 2 applied to a drive that does not support EtherCAT mode - C1 with a bad value regarding the drive type or the firmware type - C200 with a bad value regarding the drive type or the firmware type 	x	ON	1	All
44	SET REG BAD PAR	This register affection function is not yet implemented	<p>General case:</p> <ul style="list-style-type: none"> - Please contact your provider. 		ON	1	All
45	ERR CFG DOUT	This error occurs when several actions are selected on a DOUT	<p>General case:</p> <ul style="list-style-type: none"> - By using on ComET the IO control tool, you can see if a DOUT is reserved for any special feature. Or you can check following registers: - K357 state in case of error - K358 sti movement start - K359 Trigger - K363 motor brake - C358 output managed by RTV - C12 Stage Protection - C13 Stage Protection 		ON	1	All
46	ERR CFG TRIGGER	This error occurs when a bad setting for the trigger is entered	<p>M264 = 460001: <ul style="list-style-type: none"> - Trigger type set in EL table (depth 1) is not a valid value. </p> <p>M264 = 460002: <ul style="list-style-type: none"> - Trigger Combi set in EL table (depth 2) is not a valid value. </p> <p>M264 = 460003: <ul style="list-style-type: none"> - Trigger Pulse Generator set in EL table (depth 3) is not a valid value. </p> <p>M264 = 460004: <ul style="list-style-type: none"> - Trigger TRS command defines too many events. </p> <p>M264 = 460005: <ul style="list-style-type: none"> - Trigger TRE, TRM, and TRM2D commands defines too many events. </p> <p>M264 = 460006: <ul style="list-style-type: none"> - PG1 or PG2 is defined by the two axes. This error will occur only when the TRE, TRM, TRM2D or SPG is executed. - Example: K342:0.0 = <value> and K342:0.1 = <value>. </p>		ON	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
			<p>M264 = 460007: - Both K342 and KF342 are set for the same PG: Situation not allowed.</p> <p>M264 = 460008: - Both K343 and KF343 are set for the same PG: Situation not allowed.</p> <p>M264 = 460009: - Both K339 and KF339 or K344 and KF339 are set for the same PG: Situation not allowed.</p> <p>M264 = 460011: - KF353 is set to 67. It is not allowed to use ML register to modulate the pulse generator.</p> <p>M264 = 460013: - Depth 0 of k337 is out of range. (1 to 512 included)</p> <p>M264 = 460014: - Depth 1 of k337 is out of range. (0 to 15 included)</p> <p>M264 = 460015: - TRE, TRM, TRS or TRM2D is enable when TTL_EMU is already running</p>				
49	ETH. NOT CONNECT	Ethernet connection lost with open communication port	<p>General case: - Check Ethernet connection.</p>	ON	1	All	
50	TCP OUT BUF FULL	The output TCP buffer is full	<p>General case: - Check if the network or your computer is not too busy.</p>	ON	1	All	
51	TCP IN BUF FULL	One of the TCP/IP input buffer is full	<p>General case: - Check in your application if there are not too much asynchronous commands.</p>	ON	1	All	
53	TRIGGER ENABLED	TRE or TRM command sent when trigger function is active	<p>General case: - Check in your application.</p>	ON	1	All	
54	TRIGGER MISSED	Trigger: detect a missed event	<p>General case: - Check missed event configuration and trigger definition.</p>	ON	1	300, 400/ 600, 48, VHP 100, VHP 48	
55	TIME OUT KEEPALIV	Time-out keep-alive, too long time without Ethernet communication	<p>General case: - Check Ethernet connection. - Check if your application is not stopped by a break point in debug mode.</p>	ON	1	All	
56	TIMEOUT TRANSEN	The controller does not receive TransnET messages	<p>General case: - Check the daisy chain of the TransnET.</p>	ON	1	All	
57	DIPSWITC BAD CFG	Error on dip switch configuration, put at 62 (axis number)	<p>General case: - Check node number must be < 62.</p>	x	ON	1	All
58	BAD SLOT TRANSNET	Bad slots configuration on TransnET for real time value	<p>General case: - Check you RTV configuration.</p>	ON	1	All	
60	LEAVEREF ERROR	Error in homing process when leaving the homing reference (home switch or limit switch or reference mark). This error occurs when after the trip from KL48 the home switch or limit switch or reference mark is still present.	<p>General case: - Check the distance you need to leave the switch. - Check if you have the correct logic on the switch. - Check cable. - If "Movement to leave a switch (KL47)" is smaller than tracking error parameter ==> see also TRACKING ERROR.</p>	ON	1	All	

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
61	MULT IDX SEARCH	Error in multi-index homing process. This error occurs when the motor has reached the two mechanical end stops (detections are done by KL43 and KF44) or the two limit-switch without having finished the homing process (for linear motors)	If the motor moves: - The parameters for the mechanical end stop detection are too low. - See also TRACKING ERROR. If the motor goes in both mechanical end stops: - Bad setting of the encoder head (no index detected). - The scale is dirty and the indexes cannot be seen by the head encoder. - The distance between the index is not set correctly. If the motor does not move: - Check the motor cable. - Check the regulation. - Check the encoder signals. - Check the phasing procedure.		ON	1	All
62	SING IDX SEARCH	Error in single index homing process. This error occurs when the motor has reached the two mechanical end stops (detections are done by KL43 and KF44) or the two limit-switch without having finished the homing process (for linear motors)	If the motor moves: - The parameters for the mechanical end stop detection are too low. - See also TRACKING ERROR. If the motor goes in both mechanical end stops: - Bad setting of the encoder head (no index detected). - The scale is dirty and the indexes cannot be seen by the head encoder. If the motor does not move: - Check the motor cable. - Check the regulation. - Check the encoder signals. - Check the phasing procedure.		ON	1	All
63	SYNCHRO START	Error in start synchro on input (STI command)	General case: - The time-out parameter (K164) has to be adjusted.		ON	1	All
64	FOUND MEC STOP	Mechanical end stop found by mistake during homing process	General case: - Check the cable and the position of the limit switch.		ON	1	All
65	OUT OF STROKE	Error when the real position (given by ML7) is out of the limits defined by KL34 and KL35 parameters when bit#1 of K36 is set	General case: - The command of position has been sent over the possible stroke.		ON	1	All
66	REF OUT OFSTROKE	Error when the reference position is out of the limits defined by KL34 and KL35 parameters when bit#2 of K36 is set	General case: - Check the reference position, as well as the software limits parameters (KL34 , KL35 and K36 parameters).		ON	1	All
67	MVT NOT ALLOWED	Movement not possible because the axis not powered	General case: - Switch on the motor!		ON	1	All
69	HOME NOT POSSIBLE	Homing is not possible due to the configuration of the controller and the setting	General case: - Check your homing type (K40 parameter). - Check your limit switch configuration (K58 parameter). - Some homing types are not compatible when driving a stepper motor in open loop. Please refer to the controller documentation.		ON	1	All
70	BAD CMD PARAM	Error when a bad register (type or value) is associated with a command	General case: - Refer to software manual for the specific command.		ON	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
71	HOMING NOT DONE	Movement not possible because the homing is not done	General case: - Homing must be done before any motion.		ON	1	All
72	NO PWR ON	Power On not possible because the controller is not properly configured	General case: - Restart your drive setting.		ON	1	All
73	STRETCH ERROR	Encoder Stretch configuration not possible because axis is in interpolation mode	General case: - The Encoder Stretch function must be enabled before to start the interpolation mode.		ON	1	All
74	ER SCALE MAPPING	Scale mapping configuration not possible because axis is in power ON	General case: - The Scale Mapping must be enabled before switching on the motor.		ON	1	All
75	STAGEMAP ACTIVE	This action is not possible when the stage error mapping compensation is active	General case: - This error occurs when mapping configuration commands (like ASP,...) or IND command are executed when the stage error mapping compensation is active.		ON	1	All
76	STAGEMAP CFG	Stage error mapping functionality is not properly set	General case: - This error occurs when the number of mapping steps is wrongly defined (less than 1 step) or when the number of mapping dimensions is incorrect (must be either 1, 2 or 3).		ON	1	All
77	STAGEMAP STEP ERR	Stage error mapping calculation not possible due to bad setting or steps too small	General case: - Reduce the maximum speed or increase the step size. The maximum speed is defined by the step divided by a MLTI.		ON	1	All
78	NO POWER DUAL FDB	Power ON not possible in Dual encoder Feedback	General case: - Power ON not possible in Dual encoder Feedback.		ON	1	All
79	TRACKING DUAL FDB	Tracking error in Dual encoder Feedback	General case: - Reserved.		ON	1	All
80	ERR SOFT LIMIT	Soft limit not available in rotary movement	General case: - Reserved.		ON	1	All
81	TRACES BAD CFG	The value for K123 must be a multiple of the PLTI frequency / 25 us	General case: - Reserved.		ON	1	All
82	ERR CFG DUAL FDB	Bad configuration for Dual Encoder Feedback	General case: - K76:0 and K76:1 could not be both different than 0		ON	1	All
86	USB CHECK-SUM	USB checksum error	General case: - Check the EMC environment.		ON	1	All
87	USB BUFF IN FULL	USB buffer in full	General case: - Check in your application if there are not too much asynchronous commands.		ON	1	All
88	IDX ONLY IN 1 DIR	Index mark could be detected only in one direction	General case: - Check setting of the encoder head		ON	1	All
89	SETTING FCTRL	Bad setting for the Force Control loop	General case: - Bad setting for the Force Control loop		ON	1	All
90	FORCE CTRL CFG	Bad configuration for the Force Control mode	General case: - The command SETFC is sent with K302.<axis> = 0.		ON	1	All
91	MVT NOT ALLOWED	Movement not possible because the axis is in Force Control mode	General case: - When a MVE, RMVE, STE_ABS, STE_ADD, STE_SUB, STA STI, IND, or SLS command is sent after the SETFC command but before a RESETFC command. - When a MVE, RMVE, STE_ABS, STE_ADD, STE_SUB, STA STI, IND, or SLS command is sent after the FCON command but before a FCOFF command.		ON	1	All
92	MVT NOT ALLOWED	Movement not possible because the axis in ITP mode	General case: - Indicates the receipt of a command generating motion (MVE, RMVE, STE_ABS, STE_ADD, STE_SUB, STA STI, FCON, FCOFF, SETFC, RESETFC) while the axis is in interpolated mode.		ON	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
93	MVT NOT ALLOWED	Movement not possible because the axis in external reference mode	General case: - Indicates the receipt of a command generating motion (MVE, RMVE, STE_ABS, STE_ADD, STE_SUB, STA_STI, FCON, FCOFF, SETFC, RESETFC) while the axis is in external reference mode(K61!=1 or K63!=0).		ON	1	All
98	HOME SPEED	The speed during homing will generate an encoder signal frequency higher than 500kHz	General case: - Homing speed to high (KL41).		ON	1	VHP 100, VHP 48, ZxT
99	EXTBOARD CORRUPT	Optional board flash is corrupted	General case: - Please contact your provider.	x	ON	1	300, 400/600, 48
100	F TO F ERROR	Internal error (any F2F errors)	General case: - Please contact your provider.	x	OFF	3	VHP 100, VHP 48, ZxT
101	CLOCK WATCHDOG	FPGA1 and 2 any watchdog errors	General case: - Please contact your provider.	x	OFF	3	VHP 100, VHP 48, ZxT
102	ERR VOLT BOARD	FPGA2 2.5v or other voltages errors	General case: - Please contact your provider.	x	OFF	3	VHP 100, VHP 48, ZxT
103	ERROR CLOCK	FPGA2 PLL RX clock or oscillator errors	General case: - Please contact your provider.	x	OFF	3	VHP 100, VHP 48, ZxT
104	ERROR PLL	FPGA2 PLL RX clock errors	General case: - Please contact your provider.	x	OFF	3	VHP 100, VHP 48, ZxT
105	FPGA2 INIT ERR	FPGA2 initialization failure	General case: - Please contact your provider.	x	OFF	3	VHP 100, VHP 48, ZxT
106	AD BAD SETTING	Setting error for current measurements	General case: - Please contact your provider.		OFF	1	All
116	EXTERNAL ERROR	This error is generated by the ERR command	General case: - Another controller of the same error propagation group is in error. - The command "ERR" has been sent.		ON	1	All
118	GANTRY ERROR	Error from the other axis from the gantry	General case: - Check the error on the other axis.		ON	1	All
119	GANTRY TRACKING	Tracking error from the gantry	General case: - The position difference between the two motors is bigger than K30:1. Check your machine, your setting or K30:1.		ON	1	All
120	GANTRY PWR ERR	Error when not all axes of the gantry are in Power On	General case: - In gantry level 1 the power on must be done on both axis in the same time. - If the advanced gantry feedforward is used, the RTV configuration must be set correctly.		ON	1	All
121	GTRY BAD CMD LVL2	Command cannot be executed in gantry mode level2	General case: - Mask of the axes are not correct for the gantry mode 2.		ON	1	All
122	GTRY ERR PARAM	Parameters are different between the two gantry axes when in level 1 or 2	General case: - Check M265 to know which parameter is different. Depth 0: parameter number, depth 1: parameter type, depth 2: parameter depth.		ON	1	All
123	GTRY NO SEQ LVL2	No sequence on slave gantry axis in gantry level 2	General case: - No possibility to have sequence on the second axis of a gantry in mode 2.		ON	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
124	GTRY ERR HM OFFST	The difference between the two home offsets is too big	General case: - Check your mechanical alignment because KL45_master - KL45_slave is bigger or lower than KL49:0 and KL49:1.		ON	1	All
128	BAD GAIN SCHEDUL	Error on the setting for the gain scheduling	General case: - Check value parameter K10 just after reboot		ON	1	All
130	LEAK OVER CUR	Leak overcurrent error	General case: - Can be come from a short- circuit against PE, check your cable.	x	OFF	3	300, 400/ 600
131	BRIDGE FAULT	Error on the bridge	General case: - Check the EMC environment. - Remove the motor cable and try to do a power on. If the error comes again contact your provider; if not, check the motor and the motor cable.	x	OFF	3	300
132	POWER RELAYERR	This error occurs when the control signal coming from the power relay is at 0V and the controller is in Power On	General case: - Check if the pin PSR of the connector X103 on the controller has 24VDC.		OFF	1	300, 400/ 600
135	BRIDGE OVERCUR1	Bridge hardware overcurrent on first axis	General case: - Check the EMC environment. - Remove the motor cable and try to do a power on. If the error comes again contact your provider; if not, check the motor and the motor cable.	x	OFF	3	400/600, 48, VHP 100, VHP 48, ZxT
136	BRIDGE OVERCUR2	Bridge hardware overcurrent on second axis	General case: - Check the EMC environment. - Remove the motor cable and try to do a power on. If the error comes again contact your provider; if not, check the motor and the motor cable.	x	OFF	3	400/600, 48, VHP 100, VHP 48, ZxT
137	LIN AMP FAULT	Error on the linear amplifier (thermal protection)	General case: - Check if no short circuit on the motor. - Check if current loop is not oscillating. - Please contact your provider.	x	ON	2	VHP 100, VHP 48, ZxT
138	LINEAR OVCUR1	The current provided by linear amplifier PH1 is too high	General case: - Check if no short circuit on the motor. - Check if current loop is not oscillating.		ON	2	VHP 100, VHP 48, ZxT
139	LINEAR OVCUR2	The current provided by linear amplifier PH2 is too high	General case: - Check if no short circuit on the motor. - Check if current loop is not oscillating.		ON	2	VHP 100, VHP 48, ZxT
140	LINEAR OVCUR3	The current provided by linear amplifier PH3 is too high	General case: - Check if no short circuit on the motor. - Check if current loop is not oscillating.		ON	2	VHP 100, VHP 48, ZxT
141	PLTI WATCHDOG	Watchdog error on the PLTI	General case: - Please contact your provider.	x	OFF	2	All
144	ERROR OSCILLAT	Oscillator error	General case: - Please contact your provider.	x	OFF	3	All
149	K90 = 1 BANNED	Init with K90 = 1 not allowed on VHP	General case: - This init mode is not allowed on VHP.		ON	1	VHP 100, VHP 48, ZxT
150	INITIALI MOTOR 1	Bad measure during phasing process when K90=1	General case: - The current loop may be set with too high gains. - The pulse current level may be set too high. - The motor moves during the phasing procedure.		OFF	2	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
151	INITIALI MOTOR 2	Bad time measurement during phasing process when K90=1	General case: - The current loop may be set with too high gains. - The pulse current level may be set too high. - The motor moves during the phasing procedure.		OFF	2	All
152	INITIALI POWER OF	Error when the controller is disabled during phasing process	General case: - An !HLO has been sent during the phasing process.		OFF	2	All
153	INITIALI LOW CUR	Movement is too small with small movement phasing	General case: - The current pulse value is too small (increase KF91). - The motor can not move (break active). - Motor cable unconnected. - Encoder cable not connected or the encoder is damaged.		OFF	2	All
154	INITIALI HIGH CUR	Movement is too important with small movement phasing	General case: - The current pulse value is too high (decrease KF91). - This error may appear if the motor moves during the procedure. A second phasing procedure is needed.		OFF	2	All
156	TIMEOUT AUT CMD	Time-out during AUT command	General case: - Phasing in constant current has been made against the mechanical end stop. - KF92 may be set too low.		OFF	2	All
164	ETHERCAT FAULT	Invalid EtherCAT configured station alias	General case: - Set the station alias either with dip-switches or EtherCAT master. Choose any of both, but the other one must be set to 0.	ON	1	300, 400/ 600, 48	
165	ETHERCAT FAULT	RTV bad configuration	General case: - RTV bad configuration - The available register types are M=0, K=1, C=2, X=3. - The available data type are INT=0, LONG=1, FLOAT=2, DOUBLE=3. - Caution: data type DOUBLE is only available for X registers.	ON	1	300, 400/ 600, 48	
166	ETHERCAT FAULT	Over speed demanded in active profile mode	General case: - Over speed demanded in active profile mode	ON	1	300, 400/ 600, 48	
181	ENCODER NOT SUPP	Encoder type not supported	General case: - The encoder type is not supported. - Check the encoder cable.	x	ON	2	All
190	SWITCH OFF & ON	The controller has executed a save operation (SAV command or firmware download)	General case: - Each time the "SAV" command is used or a new FW is download over TransnET, this error occurs and the controller has to be reset with the command "RSD.X=255".	x	OFF	2	All
191	ERROR DIG HALL	Wrong value coming from the digital Hall sensor	General case: - No hall effect connected with K90 = 3, 4 or 5. - Check cable of hall effect.		OFF	2	All
210	STG PROT INIT ERR	This error occurs when the stage protection cannot be properly initialized when the controller starts-up	General case: - Redo the setting of the Stage Protection.		ON	1	All
211	CFG EXT1 ERROR	External error activated by a DIN. The DIN mask is specified by mask CL211	General case: - External error activated by a DIN. The DIN mask is specified by mask CL211.		ON	1	All
212	CFG EXT2 ERROR	External error activated by a DIN. The DIN mask is specified by mask CL212	General case: - External error activated by a DIN. The DIN mask is specified by mask CL212.		ON	1	All
213	CFG EXT3 ERROR	External error activated by a DIN. The DIN mask is specified by mask CL213	General case: - External error activated by a DIN. The DIN mask is specified by mask CL213.		ON	1	All

M64	Displayed message	Description	Solutions	HW res	Brk	Protection Type	Product
214	STG PROT OTHERAXI	Stage protection other axis error. This error occurs when another axis generates an error	General case: - Check the error of the other axis of the controller.		ON	1	All
215	STG PROT SETTING	This error occurs when the thresholds for "Stage protection low Speed" or "Stage protection minimum deceleration" are not properly configured or K79:0.axis has a wrong value	General case: - Check the setting of the Stage Protection.		ON	3	All
220	COMMAND NOTALLOW	Command not allowed on the current configuration	General case: - Command not allowed on the current configuration.		ON	1	All
221	ASSIGN NOTALLOW	Assignment not allowed on the current configuration	General case: - Assignment not allowed on the current configuration.		ON	1	All
240	ENCODER LOST TTL	A forbidden transition happens in TTL poor counter	General case: - The encoder's head is not correctly fixed. - The speed during a movement may be too high (The signal frequency is too high (if possible change K72)). - Unconnected or broken cable. - The encoder's head or/and the scale are dirty.		ON	2	All
400	SEQ BAD LABEL	Sequence: jump to an undefined label	General case: - A non-existent function has been called from the sequence.		ON	1	All
401	SEQ BAD REG IDX	Sequence: bad register index or depth	General case: - Sequence: bad register index or depth.		ON	1	All
402	SEQ BAD THREAD	Sequence: bad thread	General case: - A non-existent thread has been called from the sequence; the limitation is 2 on the AccurET and 3 on Ultimet.		ON	1	All
403	SEQ RUN TWICE	Sequence: thread already running	General case: - Sequence: Thread already running.		ON	1	All
404	SEQ DIV BY ZERO	Sequence: division by zero	General case: - Sequence: Division by zero.		ON	1	All
405	SEQ BAD CMD PAR	Sequence: bad command parameter	General case: - Sequence: Bad command parameter.		ON	1	All
406	SEQ BAD ACCESS	Sequence: bad register access	General case: - Sequence: Bad register access.		ON	1	All
407	SEQ BAD NODE	Sequence: bad node	General case: - Sequence: Bad node.		ON	1	All
408	SEQ FCT NOT AVAI	Sequence: call to a function that is not implemented in the firmware	General case: - Sequence: function not existing.		ON	1	All
410	STACK OVERFLOW	Sequence: stack overflow	General case: - Stack overflow on the Compiled Sequence. Modify your code to reduce stack usage or switch to a stack in external memory (K415 = 0) if not already done.	x	ON	1	All
460	FORMAT SEQ ERR	Sequence: incorrect sequence format in flash	General case: - Sequence: incorrect sequence format in flash.		ON	1	All

22. Units conversion

The controller does not accept ISO unit values system (meter, kilogram, amp, second) but those from a system with **increment**. In this chapter, the conversion formula are given to calculate an increment value from an ISO one and vice-versa. First, the **cinematic** quantities such as distance, speed, acceleration and jerk time will be looked at and then **time** quantities units.

22.1 Cinematic quantities units

There are always two cinematic quantities systems side by side in the controller. The first is the user increment [ui], and the second is the drive increment [di]. Some values have to be given in user increment and others in drive increment. All internal calculations done by the controller are in drive increment because the resolution is better.

In the following chapters, the different cinematic quantities are abbreviated as mentioned in the following table, otherwise the complete conversion formula will be given from case to case.

	Notation	Signification	Equivalent in the ISO system
Linear motors	[upi]	User Position Increment	[m]
	[usi]	User Speed Increment	[m/s]
	[uai]	User Acceleration Increment	[m/s ²]
	[dpi] [dpi2] [dpi3]	Drive Position Increment for main encoder Drive Position Increment for secondary encoder Drive Position Increment for auxiliary encoder	[m]
	[dsi]	Drive Speed Increment	[m/s]
	[dai]	Drive Acceleration Increment	[m/s ²]
Rotary motors	[rupi]	Rotary User Position Increment	[turn]
	[rusi]	Rotary User Speed Increment	[turn/s]
	[ruai]	Rotary User Acceleration Increment	[turn/s ²]
	[rdpi] [rdpi2] [rdpi3]	Rotary Drive Position Increment for main encoder Rotary Drive Position Increment for secondary encoder Rotary Drive Position Increment for auxiliary encoder	[turn]
	[rdsi]	Rotary Drive Speed Increment	[turn/s]
	[rdai]	Rotary Drive Acceleration Increment	[turn/s ²]

The offset between [dpi] and [upi] is given by the monitoring **M4** (ML4), then $[upi] = ([dpi] + M4)$.

22.1.1 Linear motors

22.1.1.1 User increments, linear motors

Abbreviation used in the table below:

$PCod$ = Encoder period [m] (given by parameter M239*1E-9, refer to [§7.3](#) for more information)
 h = Manager loop time interrupt [s] (refer to [§1.](#) for more information)

Unit	Quantity	Conversion formulae	Concerned quantities
[upi]	Distance user position increment	$[m] \rightarrow [upi]$ $Distance [upi] = Distance [m] \cdot \frac{M241}{PCod}$ $[upi] \rightarrow [m]$ $Distance [m] = Distance [upi] \cdot \frac{PCod}{M241}$	<ul style="list-style-type: none"> KL27, KL34, KL35, K37, K39, KL45, KL46, KL47, KL48, KL49, KL208, KL210 M6, ML6, M7, ML7, M36, ML36, M37, ML37, ML165, ML166, M340, ML340, ML399, ML434, ML435 CL240, CL241
[usi]	Speed user speed increment	$[m/s] \rightarrow [usi]$ $Speed [usi] = Speed [m/s] \cdot \frac{256 \cdot M241 \cdot h}{PCod}$ $[usi] \rightarrow [m/s]$ $Speed [m/s] = Speed [usi] \cdot \frac{PCod}{256 \cdot M241 \cdot h}$	<ul style="list-style-type: none"> KL41, KL211, KL181 M211, ML211
[uai]	Acceleration user acceleration increment	$[m/s^2] \rightarrow [uai]$ $Acceleration [uai] = Acceleration [m/s^2] \cdot \frac{65536 \cdot M241 \cdot h^2}{PCod}$ $[uai] \rightarrow [m/s^2]$ $Acceleration [m/s^2] = Acceleration [uai] \cdot \frac{PCod}{65536 \cdot M241 \cdot h^2}$	<ul style="list-style-type: none"> KL42, KL206, KL212 M212, ML212

Remark: In the above-mentioned formulas, monitoring M241 corresponds to $1024 * 2^{K77}$ (for an analog and EnDat2.1 encoder), $64 * 2^{K77}$ (for a TTL encoder) and 2^{K77} (for an EnDat2.2 encoder).

Refer to [§7.3](#) to know the meaning of the different depth of the parameter K77.

The position given in [upi] is available on 48 bits.

The speed given in [usi] is limited to $(2^{31} * (MLTI/PLTI))-1$

The acceleration in [uai] is limited to $(2^{31} * (MLTI/PLTI)^2)-1$

Example:

If a 12 cm movement is wanted with a linear motor, the MVE command is used. The value has to be calculated in increments corresponding to 12 cm with the above table. With a 40µm period of analog encoder and the parameter K77 equal to 4, the value in increment corresponding to a 12cm movement is obtained as follows:

$$Distance [upi] = Distance [m] \cdot \frac{M241}{PCod} = Distance [m] \cdot \frac{1024 \cdot 2^{K77}}{PCod} = 0.12 \cdot \frac{1024 \cdot 2^4}{40 \cdot 10^{-6}} = 49152000$$

The value which has to be programmed is 49152000 increments for this movement.

22.1.1.2 Drive increments, linear motors

Abbreviation used in the table below:

PCod = Encoder period [m] (given by parameter M239*1E-9, refer to [§7.3](#) for more information)

k = Position loop time interrupt [s] (refer to [§1.](#) for more information).

Unit	Quantity	Conversion formulae	Concerned quantities
[dpi]	Distance drive position increment	<p>[m] → [dpi]</p> $\text{Distance [dpi]} = \text{Distance [m]} \cdot \frac{M241:0}{PCod:0}$ <p>[dpi] → [m]</p> $\text{Distance [m]} = \text{Distance [dpi]} \cdot \frac{PCod:0}{M241:0}$	<ul style="list-style-type: none"> KL43, K30, KL166, KL167, K305, K307, K308, KL370, KL371, K373, K374 M0, ML0, M1, ML1, M2, M3, M4, ML4, M5, ML5, M9, ML9, M17, ML17, M18, M19, ML19, M21, M22, ML39, M104, ML104, M117, ML117, M119, M138, M139, M251, ML251, M252, ML252, M391
[dpi2]	Distance drive position increment	<p>[m] → [dpi2]</p> $\text{Distance [dpi2]} = \text{Distance [m]} \cdot \frac{M241:1}{PCod:1}$ <p>[dpi2] → [m]</p> $\text{Distance [m]} = \text{Distance [dpi2]} \cdot \frac{PCod:1}{M241:1}$	<ul style="list-style-type: none"> M13, ML13, M33, ML33, ML40, M105, ML105, ML106
[dpi3]	Distance drive position increment	<p>[m] → [dpi3]</p> $\text{Distance [dpi3]} = \text{Distance [m]} \cdot \frac{M241:2}{PCod:2}$ <p>[dpi3] → [m]</p> $\text{Distance [m]} = \text{Distance [dpi2]} \cdot \frac{PCod2}{M241:2}$	<ul style="list-style-type: none"> M15, ML15, ML33, M35
[dsi]	Speed drive speed increment	<p>[m/s] → [dsi]</p> $\text{Speed [dsi]} = \text{Speed [m/s]} \cdot \frac{256 \cdot M241 \cdot k}{PCod}$ <p>[dsi] → [m/s]</p> $\text{Speed [m/s]} = \text{Speed [dsi]} \cdot \frac{PCod}{256 \cdot M241 \cdot k}$	<ul style="list-style-type: none"> K19, K31, K380 M10, M11, M109, M111, M118, ML118
[dai]	Acceleration drive acceleration increment	<p>[m/s²] → [dai]</p> $\text{Acceleration [dai]} = \text{Acceleration [m/s2] \cdot} \frac{65536 \cdot M241 \cdot k^2}{PCod}$ <p>[dai] → [m/s²]</p> $\text{Acceleration [m/s2] = Acceleration [dai]} \cdot \frac{PCod}{65536 \cdot M241 \cdot k^2}$	<ul style="list-style-type: none"> M14, M114, M115 K381

Remark: In the above-mentioned formulas, monitoring M241 corresponds to $1024 * 2^{K77}$ (for an analog and EnDat2.1 encoder), $64 * 2^{K77}$ (for a TTL encoder) and 2^{K77} (for an EnDat2.2 encoder). Refer to [§7.3](#) to know the meaning of the different depth of the parameter K77.

22.1.2 Rotary motors

22.1.2.1 User increments, rotary motors

Abbreviation used in the table below:

NPCod = Encoder periods number per turn [p/r] (given by parameter M239, refer to [§7.3](#) for more information)

h = Manager loop time interrupt[s] (refer to [§1.](#) for more information).

Unit	Quantity	Conversion formulae	Concerned quantities
[rupi]	Distance rotary user position increment	<p>[turn] → [rupi] $\text{Distance [rupi]} = \text{Distance [turn]} \cdot (M241 \cdot \text{NPCod})$</p> <p>[rupi] → [turn] $\text{Distance [turn]} = \text{Distance [rupi]} \cdot \frac{1}{M241 \cdot \text{NPCod}}$</p>	<ul style="list-style-type: none"> KL27, KL34, KL35, K37, K39, KL45, KL46, KL47, KL48, KL49, KL208, KL210 M6, ML6, M7, ML7, M36, ML36, M37, ML37, ML165, ML166, M340, ML340, ML399, ML434, ML435 CL240, CL241
[rusi]	Speed rotary user speed increment	<p>[turn/s] → [rusi] $\text{Speed [rusi]} = \text{Speed [turn/s]} \cdot (256 \cdot M241 \cdot \text{NPCod} \cdot h)$</p> <p>[rusi] → [turn/s] $\text{Speed [turn/s]} = \text{Speed [rusi]} \cdot \frac{1}{256 \cdot M241 \cdot \text{NPCod} \cdot h}$</p>	<ul style="list-style-type: none"> KL41, KL211, KL181 M211, ML211
[ruai]	Acceleration rotary user acceleration increment	<p>[turn/s²] → [ruai] $\text{Acceleration [ruai]} = \text{Acceleration [turn/s}^2\text{]} \cdot (65536 \cdot M241 \cdot \text{NPCod} \cdot h^2)$</p> <p>[ruai] → [turn/s²] $\text{Acceleration [turn/s}^2\text{]} = \text{Acceleration [ruai]} \cdot \frac{1}{65536 \cdot M241 \cdot \text{NPCod} \cdot h^2}$</p>	<ul style="list-style-type: none"> KL42, KL206, KL212 M212, ML212

Remark: In the above-mentioned formulas, monitoring M241 corresponds to $1024 * 2^{K77}$ (for an analog and EnDat2.1 encoder), $64 * 2^{K77}$ (for a TTL encoder) and 2^{K77} (for an EnDat2.2 encoder). Refer to [§7.3](#) to know the meaning of the different depth of the parameter K77.
The position given in [rupi] is available on 48 bits.
The speed given in [rusi] is limited to $(2^{31} * (\text{MLTI}/\text{PLTI})) - 1$
The acceleration in [ruai] is limited to $(2^{31} * (\text{MLTI}/\text{PLTI})^2) - 1$

22.1.2.2 Drive increments, rotary motors

Abbreviation used in the table below:

NPCod = Encoder periods number per turn [p/r] (given by parameter M239, refer to [§7.3](#) for more information)

k = Position loop time interrupt [s] (refer to [§1](#) for more information).

h = Manager loop time interrupt [s] (refer to [§1](#) for more information).

Unit	Quantity	Conversion formulae	Concerned quantities
[rdpi]	Distance rotary drive position increment	<p>[turn] → [rdpi] Distance [rdpi] = Distance [turn] · M241:0 · NPCod:0</p> <p>[rdpi] → [turn] Distance [turn] = Distance [rdpi] · $\frac{1}{M241:0 \cdot NPCod:0}$</p>	<ul style="list-style-type: none"> KL43, K30, KL166, KL167, K305, K307, K308, KL370, KL371, K373, K374 M0, MLO, M1, ML1, M2, M3, M4, ML4, M5, ML5, M9, ML9, M17, ML17, M18, M19, ML19, M21, M22, ML39, M104, ML104, M117, ML117, M119, M138, M139, M251, ML251, M252, ML252, M391
[rdpi2]	Distance rotary drive position increment	<p>[turn] → [rdpi2] Distance [rdpi2] = Distance [turn] · M241:1 · NPCod:1</p> <p>[rdpi3] → [turn] Distance [turn] = Distance [rdpi2] · $\frac{1}{M241:1 \cdot NPCod:1}$</p>	<ul style="list-style-type: none"> M13, ML13, M33, ML33, ML40, M105, ML105, ML106
[rdpi3]	Distance rotary drive position increment	<p>[turn] → [rdpi3] Distance [rdpi3] = Distance [turn] · M241:2 · NPCod:2</p> <p>[rdpi3] → [turn] Distance [turn] = Distance [rdpi3] · $\frac{1}{M241:2 \cdot NPCod:2}$</p>	<ul style="list-style-type: none"> M15, ML15, ML33, M35
[rdsi]	Speed rotary drive speed increment	<p>[turn/s] → [rdsi] Speed [rdsi] = Speed [turn/s] · 256 · M241 · NPCod · k</p> <p>[rdsi] → [turn/s] Speed [turn/s] = Speed [rdsi] · $\frac{1}{256 \cdot M241 \cdot NPCod \cdot k}$</p>	<ul style="list-style-type: none"> K19, K31, K380 M10, M11, M109, M111, M118, ML118
[rdai]	Acceleration rotary drive acceleration increment	<p>[turn/s²] → [rdai] Acceleration [rdai] = Acc [turn/s²] · 65536 · M241 · NPCod · k²</p> <p>[rdai] → [turn/s²] Acceleration [turn/s²] = Acc [rdai] · $\frac{1}{65536 \cdot M241 \cdot NPCod \cdot k^2}$</p>	<ul style="list-style-type: none"> M14, M114, M115 K381

Remark: In the above-mentioned formulas, monitoring M241 corresponds to $1024 * 2^{K77}$ (for an analog and EnDat2.1 encoder), $64 * 2^{K77}$ (for a TTL encoder) and 2^{K77} (for an EnDat2.2 encoder). Refer to [§7.3](#) to know the meaning of the different depth of the parameter K77.

22.1.3 Resolution

The **resolution** is the value of a single increment expressed in the ISO unit system A distinction between the user resolution and the controller resolution as in the unit system is also necessary.

Remark: The resolution is high when the value of the resolution is small and vice-versa.

For example (with an analog encoder), a user resolution value for speed is:

$$\text{Speed [m/s]} = \underbrace{\text{Speed [usi]}}_{=1} \cdot \frac{PCod}{256 \cdot 1024 \cdot 2^{K77} \cdot h} \Rightarrow \text{User resolution [m/s]} = \frac{PCod}{256 \cdot 1024 \cdot 2^{K77} \cdot h}$$

And for a controller resolution:

$$\text{Speed [m/s]} = \underbrace{\text{Speed [dsi]}}_{=1} \cdot \frac{PCod}{256 \cdot M241 \cdot k}$$

There is a better resolution with the drive increment than the user increment. In the above example, the motor

$$\Rightarrow \text{Controller resolution [m/s]} = \frac{PCod}{256 \cdot M241 \cdot k}$$

can reach a maximum speed (SPD command) with a certain amount of user resolution, but the controller will calculate the speed trajectory with a higher resolution.

22.2 Current units

The current is in increment too.

	Notation	Signification	Equivalent in ISO unit
Current	[ci]	Current increment	[A]

Abbreviation used in the table below: $I_{\max, \text{controller}} = [A]$ maximum current delivered by the controller = M82/100

Unit	Quantity	Conversion formulae	Concerned quantities
[cur]	Current increment	$[A] \rightarrow [ci]$ $\text{Current [ci]} = \text{Current [A]} \cdot \frac{32768}{I_{\max, \text{controller}}}$ $[ci] \rightarrow [A]$ $\text{Current [A]} = \text{Current [ci]} \cdot \frac{I_{\max, \text{controller}}}{32768}$	<ul style="list-style-type: none"> • KF6, KF13, KF44, KF59, KF60, KF83, KF91, KF92, KF247, KF248, KF302, KF304 • MF20 to MF22, MF27, MF28, MF30 to MF33, MF116, MF230 to MF234, MF250, MF391, MF394, MF398, MF400, MF401, MF402, MF406, MF407

22.3 Time quantities unit

Some parameters and commands are time-related. The formulae which calculates the time in ISO unit from the time in increments are given below.

In the next part of the chapter, the time units will be abbreviated as follows and the conversion formulae given from case to case.

Notation	Signification	Equivalent in SI Unit
[mlti]	Manager Loop Time Increment	[s]
[plti]	Position Loop Time Increment	[s]
[clti]	Current Loop Time Increment	[s]

Abbreviation used in the table below: h = Manager loop time interrupt [s] (refer to [§1](#) for more information).

k = Position loop time interrupt [s] (refer to [§1](#) for more information).

m = Current loop time interrupt [s] (refer to [§1](#) for more information).

Unit	Quantity	Conversion formulae	Concerned quantities
[mlti]	Time Manager loop time increment	$[mlti] \rightarrow [s]$ $\text{Time [mlti]} = \frac{\text{Time [s]}}{h}$ $[s] \rightarrow [mlti]$ $\text{Time [s]} = \text{Time [mlti]} \cdot h$	<ul style="list-style-type: none"> K38, K164, K180, K194, K204, K213, K229, K299, K346, K347 M38
[plti]	Time Position loop time increment	$[plti] \rightarrow [s]$ $\text{Time [plti]} = \frac{\text{Time [s]}}{k}$ $[s] \rightarrow [plti]$ $\text{Time [s]} = \text{Time [plti]} \cdot k$	<ul style="list-style-type: none"> K190, K192, K193, K251, KF314 M395

22.4 Volt vs. increment

The conversion formula to know the measured encoder signal value in ISO units is:

$$\text{Encoder value } [\hat{V}] = \frac{\text{Encoder value [inc]}}{4096 \cdot 0.74}$$

23. 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. Dolnemecholupská 12b 102 00 Praha 10 - Hostivice 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 Storsätragränd 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.

Index

A

ACC, 112
 Acceleration feedforward, 39, 128
 Advanced gain scheduling, 126
 Advanced movements, 140
 Advanced reference mode, 79, 133
 Alias, 22
 Analog I/O, 161
 Analog input/output, 161
 Anti-windup, 124
 ASG, 283
 ASP, 272
 ASR, 264
 ASW, 264, 272
 AUT, 88
 Automatic setting, 88
 AXI, 47
 Axis number, 47

B

Back-EMF, 80
 Basic movements, 111
 Basic programming, 248
 Bit fields, 24
 Block statement, 320
 Break statement, 323
 BRK, 155

C

C common parameters

C1, 25
 C2, 26
 C3, 26
 C5, 159
 C6, 160
 C7, 163
 C10, 253
 C11, 253
 C12, 253
 C13, 253
 C17, 162
 CF18, 162
 C19, 30
 C27, 164, 165
 CF28, 164, 165
 C30, 165
 C31, 165
 C32, 165
 C40, 238
 C41, 238
 CF60, 304
 C100, 121
 C101, 121
 C102, 121
 CF117, 166, 167
 CF118, 166

CL211, 253
 CL212, 253
 CL213, 253
 CL240, 284
 CL241, 284
 C245, 278, 279, 281, 289, 290, 308, 330, 331
 C246, 284
 C305, 237
 C306, 236
 C307, 237
 C353, 160
 C354, 160
 C355, 160
 C358, 161
 C359, 160
 C360, 236
 C361, 236
 C362, 237
 C490, 296
 C495, 304
 CF490, 304
 CAM, 141, 153
 CEC, 26
 CFGPGFIX, 175
 CFGPGMOD, 178
 CFGTRIGEXTPOS, 191
 CH_BIT_REG32, 129, 234
 Cinematic quantities, 153
 units, 400
 CLRWAIT, 110
 CLTI, 18
 CLX, 46
 Cogging compensation, 132
 ComET software installation, 33
 Commands
 ACC, 112
 ASG, 283
 ASP, 272
 ASR, 264
 ASW, 264, 272
 AUT, 88
 AXI, 47
 BRK, 155
 CAM, 141, 153
 CEC, 26
 CFGPGFIX, 175
 CFGPGMOD, 178
 CFGTRIGEXTPOS, 191
 CH_BIT_REG32, 129, 234
 CLRWAIT, 110
 CLX, 46
 DIN, 157, 158
 DOUT, 159, 160
 END, 328
 ERR, 330
 HLB, 110
 HLO, 110
 HLT, 110
 IND, 91
 INI, 81
 JMP, 327

- JRT, 112
list, 334
LTI, 143
LTN, 143
MMC, 141, 152
MMD, 141
MVE, 113
MVETILT, 301
NEW, 45
NEWFC, 287
OFFSETAUX, 112
OFFSETSEC, 111
POSCAPT, 168
PWR, 90
RES, 44
RESETFC, 288
RMVE, 114
RMVETILT, 301
RSD, 74
RST, 74, 255
SAF, 129
SAFRX, 303
SAFRY, 303
SAV, 44
SCI, 70
SD1, 229
SD2, 229
SER, 117
SET, 111
SETFC, 286
SETFCI, 286
SETTILT, 301
SLS, 67
SMP, 149
SPD, 112
SPG, 189
SSR, 234
STA, 149
STE, 154
STETILT_ABS, 302
STETILT_ADD, 302
STETILT_SUB, 302
STI, 149
STP, 155
TARGET, 112
TRE, 182, 193, 194
TRM, 182, 193, 194
TRM2D, 188, 193, 194
TRR, 189
TRS, 189
UST, 232, 234
VALPOS, 60
VER, 117
WBC, 108
WBS, 108
WPG, 109
WPL, 109
WSG, 109
WSL, 109
WTD, 247
WTF, 289
WTM, 105
WTP, 105
WTS, 107
WTT, 105
WTW, 106, 170
WTWZXT, 303
XAIN, 162
XAOA, 165
XAOO, 165
XAOUT, 163
XSRA, 165
XSRI, 165
XSRT, 165
ZTE, 245
Commands description, 248
Commands syntax, 20
Commutation look-up table, 82
Concatenated movements, 152
Continue statement, 324
Controller
 installation, 32
 status parameters, 229
Current
 increment, 405
 limit, 68
 regulator, 37
 regulator tuning, 40
Current loop time increment, 406
Current reference generator, 129
- D**
- Digital Hall effect sensor, 85
Digital inputs/outputs, 157
DIN, 157, 158
Dip switch, 47
Do statement, 322
DOUT, 159, 160
Drive increments
 table for linear and rotary, 400
Drive increments, linear
 dai, 402
 dpi, 402
 dsi, 402
Drive increments, rotary
 rdai, 404
 rdpi, 404
 rdsi, 404
Dynamic braking, 76
- E**
- Encoder
 amplitude correction, 52, 55
 analog, 51
 EnDat, 54
 Hall effect sensor, 86
 interpolation, 230
 monitorings, 59, 60, 66
 offset correction, 51, 54
 parameters, 49
 resolution, 51, 53, 54, 56

TTL, 53, 65	KF5, 124
Encoder scale mapping, 230	KF6, 124
Encoder Stretch, 232, 234	K10, 124
END, 328	K11, 53
EnDat, 54	KF12, 125
ERR, 330	KF13, 125
Errors	KF14, 253
description, 387	KF15, 253
handling, 74	K19, 68, 125, 128
management, 328	KF20, 128
reset, 74, 255	KF21, 39, 128
ETEL-Bus-Lite2, 27	KF23, 124
External reference	K24, 128
modes, 133	KL27, 67, 115
	K30, 67
	K31, 67
	K32, 92, 105, 263, 264, 265, 271, 273
Feedforward, 127	K33, 74, 76, 77
Firmware download, 34	KL34, 67
For statement, 323	KL35, 67
	K36, 67
	K37, 229
G	K38, 170
Gain scheduling, 124	K39, 170
Gantry, 277	K40, 93
General protection parameters, 71	KL41, 93
	KL42, 93
H	KL43, 93
Hall effect sensor, 85, 86	KF44, 93
HLB, 110	KL45, 93
HLO, 110	KL46, 93
HLT, 110	KL47, 93
Homing, 91	KL48, 93
modes, 94	KL49, 281
modes summary table, 94	K52, 82
	K53, 82
I	K54, 48
i2t type error, 68	K55, 50
If statement, 320	KL55, 50
IND, 91	K56, 49, 88
Indexation	K58, 92
parameters, 93	KF59, 118
Infinite rotary movement, 144	KF60, 68, 118, 124
INI, 81	K61, 79, 133
Initialization, 79	KF61, 305
Inputs, 157, 161	K62, 229
Installation of a controller, 32	KF62, 305
Interpolation, 52	K63, 133
Interpolation factor, 53, 54, 56	KF64, 305
In-window, 170	K66, 118
IsMoving bit, 229	K68, 50
	K69, 49
J	K70, 50
JMP, 327	K71, 50
JRT, 112	KF72, 50
	KF73, 50
K	K75, 50
K parameters	K76, 58, 60
KF1, 38, 124	K77, 53, 54, 56
KF2, 38, 124	K78, 118
KF4, 38, 124	K79, 50

- KF80, 37, 129
KF81, 37, 129
KF83, 68
KF84, 68
KF85, 68
K89, 48
K90, 82
KF91, 82, 87
KF92, 83
K93, 83
K94, 83
K97, 83
K98, 82
K101, 87
KF104, 129
KF105, 129
KF106, 129
KF107, 129
KF108, 129
KF114, 305
KF115, 305
KF116, 305
KF117, 305
KF118, 305
K119, 132
K120, 242
K121, 243
K122, 243
K123, 243
K124, 243
K125, 243
K126, 243
K127, 243
KD127, 126, 243
KF127, 243
KL127, 243
K128, 243
K129, 243
K133, 89
K140, 75
K141, 75
K142, 329
K144, 328
K145, 67
K146, 73
K147, 73
K148, 73
K149, 73
K164, 151
K165, 231
K166, 78
KL166, 106, 113, 230
KL167, 231
K171, 159
K177, 229
K180, 107
KL181, 107
K182, 168
K190, 292
KF191, 292
K194, 118
K198, 24
K200, 330
K201, 141, 152
K202, 141
K203, 143
K204, 143
KF205, 141, 154
KL206, 141
K207, 142
KL208, 142
K209, 115
KL210, 112
KL211, 112
KL212, 112
K213, 112
K220, 133
K221, 133
K222, 133
K223, 133, 138
KF224, 133
K225, 133
K226, 126
KF226, 126
KL226, 126
KF227, 138
K229, 147
K230, 147
K240, 48
K241, 50
K242, 65
K243, 66
KF247, 128
KF248, 128
K249, 127
K250, 127
KL250, 304
K251, 128
K291, 156
K292, 156
K293, 26, 30
K294, 106, 113, 230
K297, 308
K298, 155
KF299, 291
K300, 151
KF300, 291
K301, 168
KF301, 291
K302, 71, 290, 330, 331
KF302, 291
KL302, 71
K303, 75
KF303, 291
K304, 72
KF304, 291
KF305, 291
KL305, 71
KF306, 291
KL306, 138, 139
K307, 291
K308, 289

- KF308, 291
 KF309, 291
 KF310, 291
 K311, 291
 K312, 278
 KF312, 291
 KF313, 291
 KL313, 278
 K314, 279
 KF314, 291
 KF316, 291
 KF317, 291
 KF318, 291
 KF319, 291
 K320-K335, 174
 K336, 172, 190
 K337, 194
 K338, 185
 K339, 174
 KF339, 178
 K340, 174
 K341, 175
 K342, 175
 KF342, 178
 K343, 175
 KF343, 178
 K344, 175
 K345, 175
 K346, 248
 K347, 192
 K348, 192
 K349, 192
 K350, 160
 K351, 160
 K352, 160
 K353, 178, 190, 193
 K354, 178, 190, 193
 K355, 178, 190, 193
 K356, 178, 190, 193
 K357, 72
 K358, 151
 K359, 160
 K360, 190, 193
 KF360, 191
 K363, 248
 K364, 249
 K365, 273
 KL370, 132
 KL371, 132
 K372, 132
 K373, 291
 KF373, 291
 K374, 291
 KF374, 291
 K380, 254
 K381, 254
 K415, 331
 K420, 57
 K421, 57
 K422, 57
 K423, 57
 K424, 57
 KF446, 61
 KF453, 61
 K460, 262
 K480, 304
 K481, 304
 K482, 304
 K483, 304
 KF490, 304
 KF492, 304
 KF493, 304
 K495, 304
 K parameters, homing parameters, 91
 K parameters, setting
 current loop regulator parameters, 129
- L**
- Limits, 66
 Look-up table movement commands, 141
 LTI, 143
 LTN, 143
- M**
- M monitorings
 M0, 130, 131, 234
 ML0, 130, 131, 234, 293
 M1, 130, 131, 137, 271
 MF1, 130, 131
 ML1, 58, 60, 130, 131, 137, 271
 M2, 130, 131
 MF2, 130, 131
 ML3, 155
 M4, 400
 ML4, 130, 137, 400
 M5, 91
 ML5, 91
 M6, 130, 131
 ML6, 130, 131
 M7, 130, 131, 137
 ML7, 130, 131, 137
 M9, 130, 131, 232
 ML9, 131, 232
 M10, 130, 131
 M11, 130, 131
 M13, 131
 ML13, 131
 M14, 130, 131
 M15, 131
 ML15, 131
 M17, 130, 131, 232, 271
 ML17, 130, 131, 232, 233, 240, 271
 M18, 139
 M19, 130, 131, 232
 ML19, 130, 131, 232, 233
 MF20, 130, 131
 M21, 130, 131, 271
 MF21, 130, 131
 M22, 140
 MF22, 130, 131

MF24, 130, 131	MF110, 59, 60
M25, 130, 131	M111, 130, 131, 290
M26, 301, 305	MF112, 59, 60
MF26, 130, 131	M114, 290
MF27, 130, 131	ML114, 60
MF28, 130, 131	M115, 290
MF29, 130, 131	ML115, 60
MF30, 130, 131	MF116, 131, 137
MF31, 130, 131, 290	M117, 135, 136
MF32, 130, 131	ML117, 131, 135, 136
MF33, 130, 131	M118, 131, 136
ML33, 111	ML118, 131, 136
MF34, 130, 131	M119, 131
ML35, 112	M120, 244
M38, 106, 170	M121, 244
M40, 66	MF121, 130, 284
M41, 66	M122, 244
M43, 66	M123, 244
M44, 92	M124, 244
M48, 85	M125, 244
M50, 157	M126, 244
MF51, 166	M127, 244
M52, 157	MF127, 244
M53, 166	ML127, 244
M54, 157	M128, 244
M55, 158	M129, 244
M56, 162	M131, 244
M58, 168	M134, 244
M60, 229	ML134, 244
M61, 229	M135, 244
M63, 229	M136, 244
M64, 74	M138, 106
M66, 74	M139, 106
MF67, 68	M140, 73
M70, 117	M141, 75
M71, 117	M160, 168
M72, 117	M161, 168
M73, 47, 117	M163, 113
M82, 68, 305	ML165, 168
MF82, 68	M166, 78
ML83, 47	ML166, 168
MF84, 68	MF167, 68
M85, 117	ML167, 169
M86, 117	ML168, 169
M87, 117	M170, 160
M88, 117	M171, 159
M89, 117	M172, 159
M90, 117	M173, 164, 165
M91, 73, 117	M174, 167
M95, 117	M180, 70
MF95, 87	ML181, 70
M97, 310	M201, 330
M101, 105	M202, 330
M102, 49, 61	M204, 26
M103, 61	M205, 26
M104, 59	M211, 154
ML104, 59	ML211, 26, 154
M105, 59	M212, 154
ML105, 59	ML212, 26, 154
ML106, 59, 60	M228, 246
M109, 130, 131, 290	ML228, 246

- MF230, 130
 MF231, 130
 MF232, 130
 MF233, 130
 M239, 50
 M240, 48
 M241, 51
 M243, 18
 M244, 18
 M245, 18
 M247, 139
 M248, 139
 MF249, 305
 MF250, 130, 132, 290
 M251, 233
 ML251, 233, 240
 M252, 233
 ML252, 233
 MF253, 233
 M254, 233
 M260, 240
 M261, 240
 M262, 240
 M264, 195
 M265, 279
 M274, 254
 M282, 46
 M289, 278
 M330, 271, 273
 M340, 181
 ML340, 181
 M341, 181
 M345, 174
 M346, 181
 M349, 194
 M350, 194
 M351, 181
 M363, 249
 M370, 121
 M389, 289
 M390, 292
 M391, 291
 MF391, 291
 MF392, 291
 M395, 292
 MF395, 292
 ML396, 293
 MF399, 291
 MF400, 291
 MF401, 291
 MF402, 291
 M417, 331
 M418, 331
 ML434, 111
 ML435, 111
 M443, 264
 M447, 265
 M448, 264
 M449, 264
 M450 to M465, 265
 MF450 to MF465, 265
 ML450 to ML457, 265
 M480, 305
 M481, 305
 M491, 305
 MF496, 305
 MF497, 305
 MF500, 305
 MF510, 305
 M520, 264
 M524, 263
 M525, 263
 M526, 263
 Magnetic brake, 72
 Manager loop time increment, 406
 Mapping, 230, 267
 Master/slave, 138
 Master/slave position compensation, 134, 138
 MLTI, 18
 MMC, 141, 152
 MMD, 141
 Monitorings
 diagram, 130
 list, 372
 Motor
 maximum force, 81
 parameters, 48
 Movements
 advanced types, 140
 basic, 111
 calculation, 18
 concatenated, 152
 infinite rotary, 144
 limits, 66
 look-up table, 141
 predefined profile, 147
 rotary S-Curve, 115
 Movements commands
 LTI, 143
 LTN, 143
 MMC, 152
 MVE, 113
 MVETILT, 301
- ## N
- NEW, 45
 NEWFC, 287
 Normal mode, 79
 Numerical values, 24
- ## O
- OFFSETAUX, 112
 OFFSETSEC, 111
 Operators, 315
 Outputs, 157, 161
- ## P
- Parameters list, 350, 368
 Phase adjustment, 79
 Phase shift, 79
 PLTI, 18

POSCAPT, 168
Position capture, 168
Position counter's limits (KL27), 115
Position loop time increment, 406
Position reference mode, 135
Position regulator, 38
 tuning, 40
Position window, 170
Predefined functions, 325
Programming
 basic, 248
 structured code, 311
Protection registers, 68
Protection signals on DIN, 71
Protection signals on DOUT, 72
PWR, 90

R
Real-time channel, 261
Reference
 advanced modes, 79, 133
 external modes, 133
 normal mode, 79
Registers
 bit fields, 24
 examples of use, 24
 K parameters, 21
 L look-up table, 142
 M monitoring, 21
 maximum value, 24
 minimum value, 24
 numerical values, 24
 value attribution, 23
 X user variables, 21
Regulation loops, 18
Regulator
 current, 37
 position, 38
Regulators parameters
 advanced, 124
Relative movement, 114
RES, 44
Reset errors, 74, 255
RESETFC, 288
Resolution, 404
Return statement, 324
RMVE, 114
RMVETILT, 301
Rotary movement, 115
RSD, 74
RST, 74, 255
RTV, 261

S
SAF, 129
SAFRX, 303
SAFRY, 303
SAV, 44
Scale mapping, 230
SCI, 70

SD1, 229
SD2, 229
SER, 117
SET, 111
Set point generator, 18
Set several registers, 234
SETFC, 286
SETFCI, 286
SETTILT, 301
SLS, 67
SMP, 149
SPD, 112
Speed feedforward, 128
Speed reference mode, 136
SPG, 189
SSR, 234
STA, 149
Stage error mapping, 267
Stage protection, 250
Start movements, 149
State regulator, 124
STE, 154
Stepper in open loop, 118
STETILT_ABS, 302
STETILT_ADD, 302
STETILT_SUB, 302
STI, 149
Stop a sequence, 110
STP, 155
Switch statement, 321
Syntax, 20

T
TARGET, 112
Thread, 310, 328
Traces management, 242
Trajectory filter, 242
TRE, 182, 193, 194
Trigger, 171
 1D trigger, 181
 2D trigger, 183
 continuous trigger, 193
 missed event, 191
TRM, 182, 193, 194
TRM2D, 188, 193, 194
Troubleshooting, 74
TRR, 189
TRS, 189
Tuning, 40

U
Units conversion, 324, 400
User increments
 table for linear and rotary, 400
User increments, linear
 uai, 401
 upi, 401
 usi, 401
User increments, rotary
 ruai, 403

rupi, 403
rusi, 403
UST, 232, 234

V

VALPOS, 60
VER, 117

W

Wait and pause commands, 242

Warnings

 handling, 74

WBC, 108

WBS, 108

While statement, 322

WPG, 109

WPL, 109

WSG, 109

WSL, 109

WTD, 247

WTF, 289

WTM, 105

WTP, 105

WTS, 107

WTT, 105

WTW, 106, 170

WTWZXT, 303

X

XAIN, 162

XAOA, 165

XAOO, 165

XAOOUT, 163

XSRA, 165

XSRI, 165

XSRT, 165

Z

Zero machine, 111

ZTE, 245

ZxT functions, 295