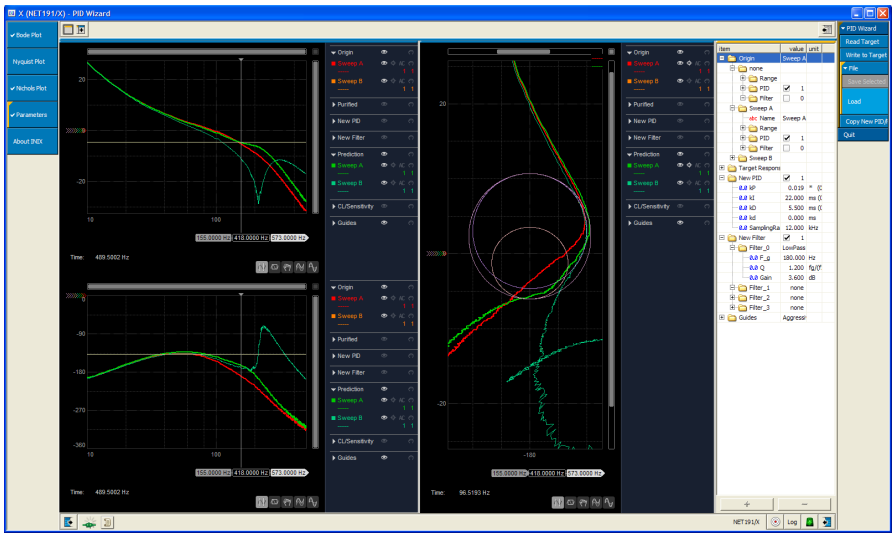
INDEL AG
INDUSTRIELLE ELEKTRONIK

Indel SAC-Drives

Commissioning Manual



Made in Switzerland

Table of contents

1 Introduction

1.1 General notes

Please read this documentation, and other documentation to which it refers, fully before installation and commissioning. Incorrect handling of the modules can lead to personal injury or property damage. Ensure that the technical details and information on connection conditions, as well as all provisions, are complied with.

1.2 Overview of the document

Section 2 provides a rough description of how to operate the new INIX tools from Indel.

Section 3 goes into detail on the many individual parameters of the motor configuration file.

Section 4 contains information on operation with Safe Torque Off, which is provided by the SAC3 controllers.

Section 5 illustrates the configurations needed on the software side for trouble-free operation. In particular, it highlights the parameters that must be identical in the SAM configuration and on the controller and/or in the motor configuration file.

Section 6 provides brief information on the controller configuration file. This is usually not of interest to the user.

Section 7 describes error messages in the drive that were not directly evaluated in the past. However, today they are evaluated directly on the controller and are visible to the user.

Section 8 discusses the position controller and its individual commands.

Section 9 briefly describes the ACS-Show tool and the settings for the trapezoid controller.

Section 10 offers instructions for the commissioning of an axis. You should go through this section step by step when commissioning.

Section 11 describes how to carry out a bode sweep with the axis tool. The Indel Axis Tuner tool, which is also described in this section, is used to adjust PID parameters and configure current filters. This means that even complex control routes can now be stabilised with little effort.

Section 12 describes how to commission open-loop stepper motors, i.e. stepper motors that are run without feedback.

2 INIX-Motion

2.1 Operation

The Inix-Motion is required for commissioning of the motor, together with the logger tool Inix-Varlog.

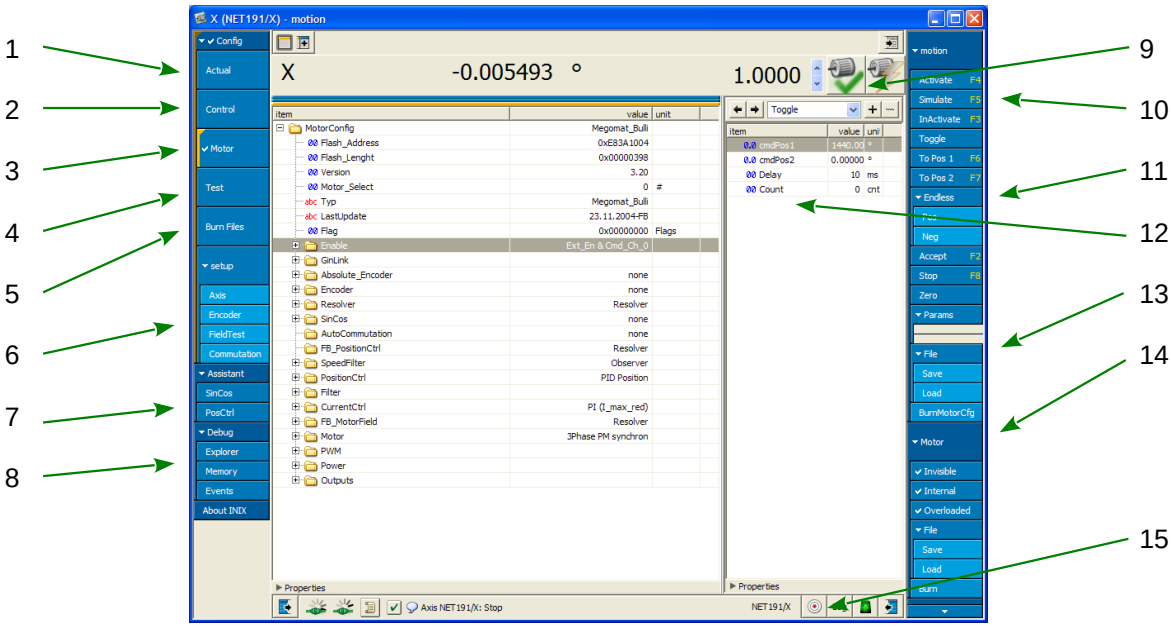


Fig. 1: INIX Motion

1) All actual values for the axis
2) Configuration of the drive (incl. drive sampling rate)
3) Motor configuration
4) Test window: current mode, voltage mode, commutation, etc.
5) Loading of configuration files and drive software
6) Various commissioning aids
7) Commissioning assistants
8) Debug: variable explorer, memory dump, etc.
9) Symbol for external release, axis active
10) Move commands
11) Endlos fahren
12) Parameter window: predefined parameter sets, own parameter sets
13) Loading/saving of motor and/or controller configuration files
14) Display/hide internal or invisible parameters
15) Target selection

2.2 Logger

The Indel variables logger is integrated into the motion tool:

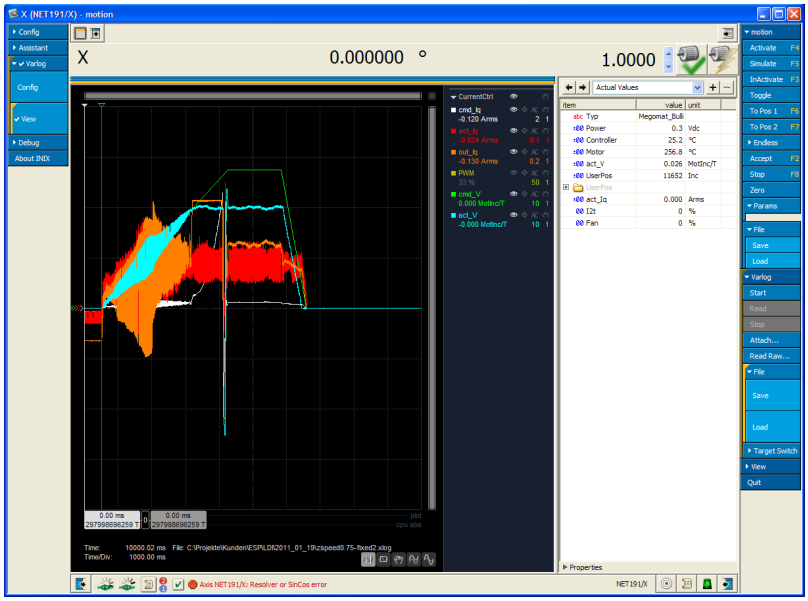


Fig. 2: Variables logger

The variables logger displays the speed in MotInc/T.

MotInc	Motor increments: The encoder resolution in the controller is standardised at 4096 increments per motor revolution. In the case of high-resolution SinCos encoders with, for example, 1'048'576 increments resolution, you will get motor increments with decimal places.
T	Sampling period: Depending on the settings, the sampling frequency in the controller is 8kHz, 12kHz, 16kHz, 24kHz oder 32kHz

Calculation of speed in °/s

v	Speed in	°/s
f	Sampling frequency in	Hz

$$v = \frac{MotInc/T \cdot f \cdot 360}{4096}$$

Calculation of the following error

s	Path in	mm
K	Spindle pitch in	mm
P	Pole spacing in	mm
G	Gear factor; only required when the feedback system is attached to the motor and the gearbox is downstream.	

for rotational motors:

$$s = \frac{MotInc \cdot 360 \cdot G}{4096}$$

for spindle motors:

$$s = \frac{MotInc \cdot K \cdot G}{4096}$$

for linear motors:

$$s = \frac{MotInc \cdot P}{4096}$$

2.3 Loading firmware, motor configuration, controller configuration

The motion tool allows the motor configuration, controller configuration and firmware to be loaded into all motion boards and servo controllers.

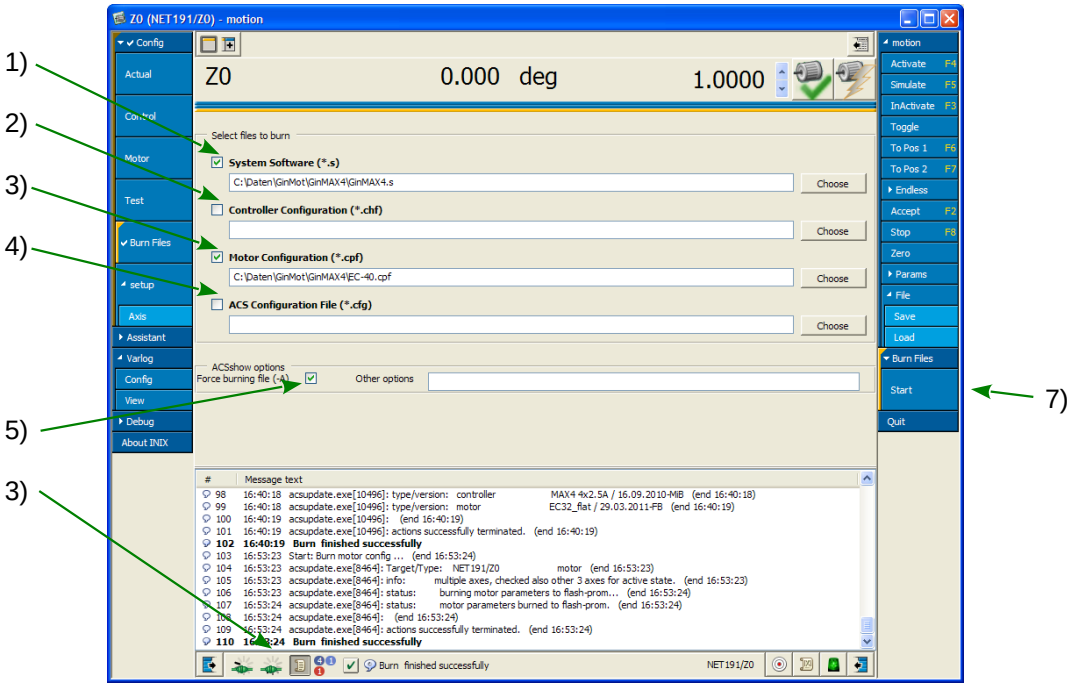


Fig. 3: Loading firmware, motor/controller configuration

1) Selection of firmware, e.g. GinMAX4.s
In the case of GinLink targets, you can also specify the motor_gin.zip file so that the right software is automatically selected and burned to the target.

2) Selection of controller configuration

The controller configuration contains adjustment data for the current-voltage measurement, SinCos inputs and the details on the maximum currents of the IGBTs.

Under no circumstances may the controller configuration for a specific motion board or servo drive be loaded into another device. In the worst case scenario, this can lead to the drive being damaged.

3) Selection of motor configuration

4) See section 13.1.6 Automating flash PROM updates

5) -A if this flag is activated, the selected files are always burned to the flash prom.

If the flag is not selected, a check is carried out to find out whether the file to be loaded is more up to date than the file in the flash prom.
In the case of firmware, attention is paid to the version; in the case of configuration files, attention is paid to the file date in the parameter: LastUpdate ; see below.

6) Message-History Show window

7) Start Button for displaying file info

Saving and burning the motor configuration

Once changes have been made to the motor configuration, these need to be saved in a file and burned to the flash prom in the controller. Otherwise, the changes will be lost when the unit is powered off. If an error occurs whilst the parameters are being burned and the drive can no longer be booted, there is an option to boot the drive within the emergency system. Please see section 13.2 Emergency system.

The motor configuration can also be saved and/or burned with the tool ACSUpdate.exe; see section: 13 Firmware update, parameter update

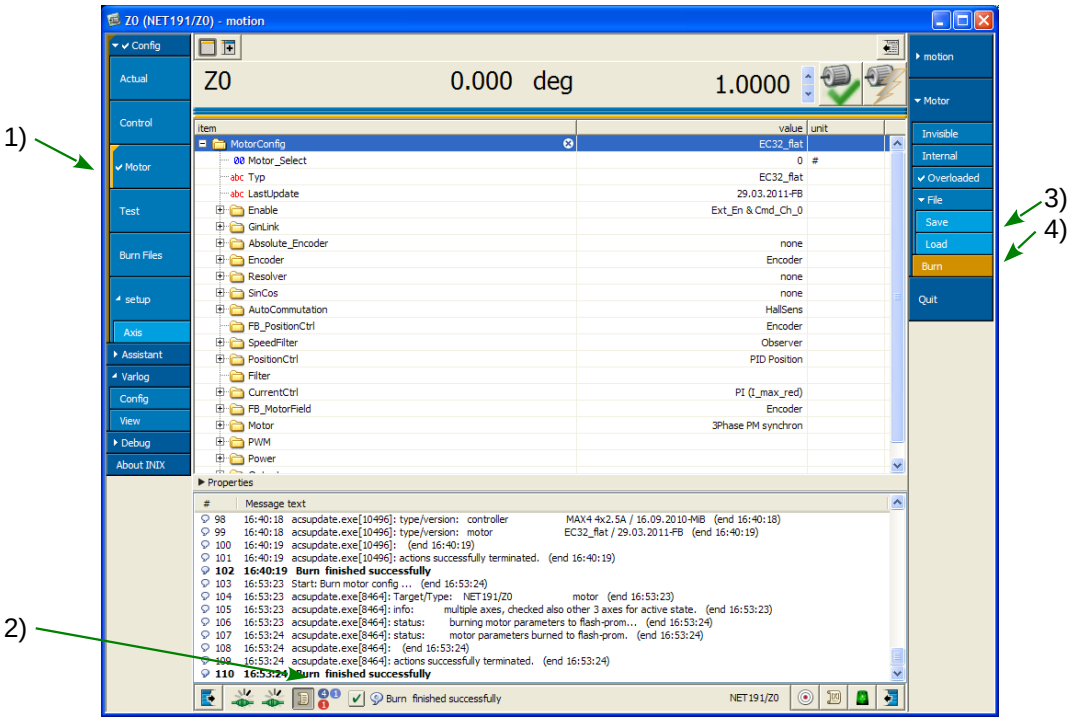


Fig. 4: Burning and saving motor configuration parameters

1) Motor configuration

2) Message-History Show window. Make sure that you observe the text in the message window in order to be sure that the parameters have actually been burned.

The parameters cannot be burned if the axis is still active!

3) Save, Load motor configuration into a file. With Load the motor parameters are loaded into the drive's RAM. In order to save the parameters permanently,
they also need to be burned to the flash prom using Burn .

4) Burn Button for loading the motor configuration into the flash prom. The Burn button turns orange as soon as a motor parameter is changed.

Motor configuration data can also be burned and saved via a context menu, which can be brought up using the right-hand mouse button.

Burning parameters to the flash:

right-hand mouse button in Inco-Tree → Motor → BurnMotorCfg
or right-hand side: File: BurnMotorCfg

Saving parameters

right-hand mouse button in Inco-Tree → Motor → File → Safe/Load
or right-hand side File: File: Safe/Load

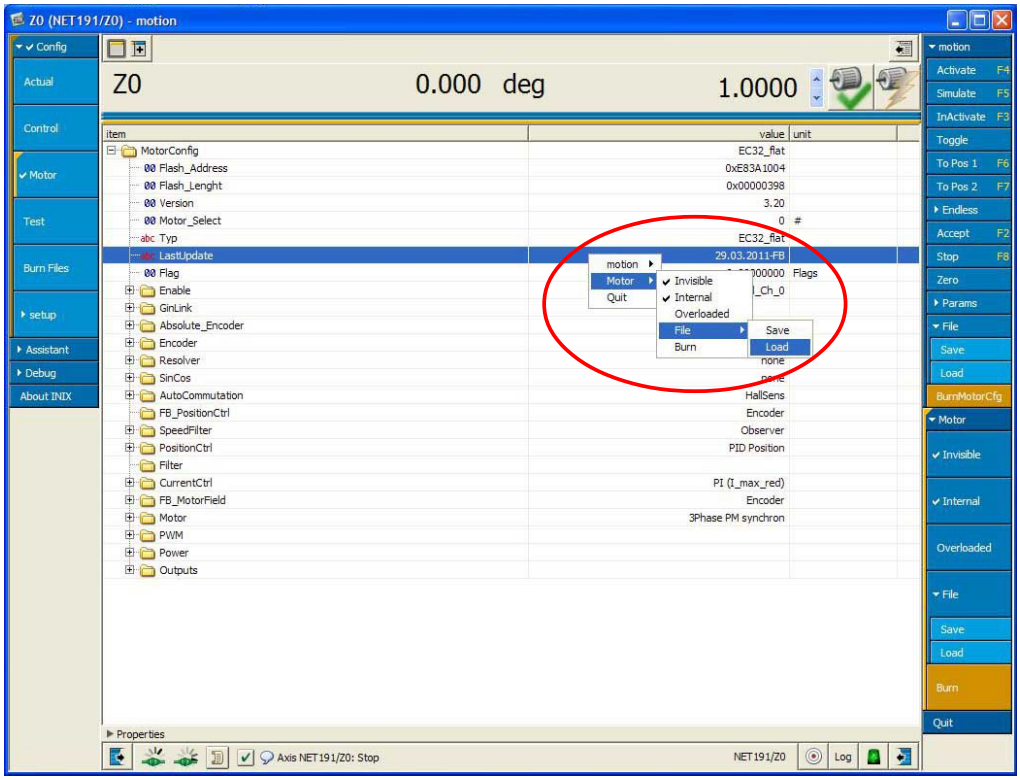


Fig. 5: Saving and burning motor configuration parameters

2.4 Burning dt2 configuration files

Motion boards require configuration for the analogue and digital periphery on the motion board, as well as the configuration of the ramps for all axes.

To do this, a dos box is opened.

Burning an entire dt2 configuration:
trans32 SAM192\AX0 -k AX_Config\dt2config -G -B

Burning an individual configuration file:
trans32 sio -k -l max-inp.dt2 -b
trans32 sio -k -l config\maxbus\digital\ip_adr.dt2 -b

Deleting an entire configuration:
trans32 SAM192\AX0 -G -B

Creating a backup of the configuration:
trans32 sio -b -W backup

Also see: http://doc.indel.ch/doku.php?id=software:application:trans32

2.5 Test Modi

There are various test modes available for commissioning:

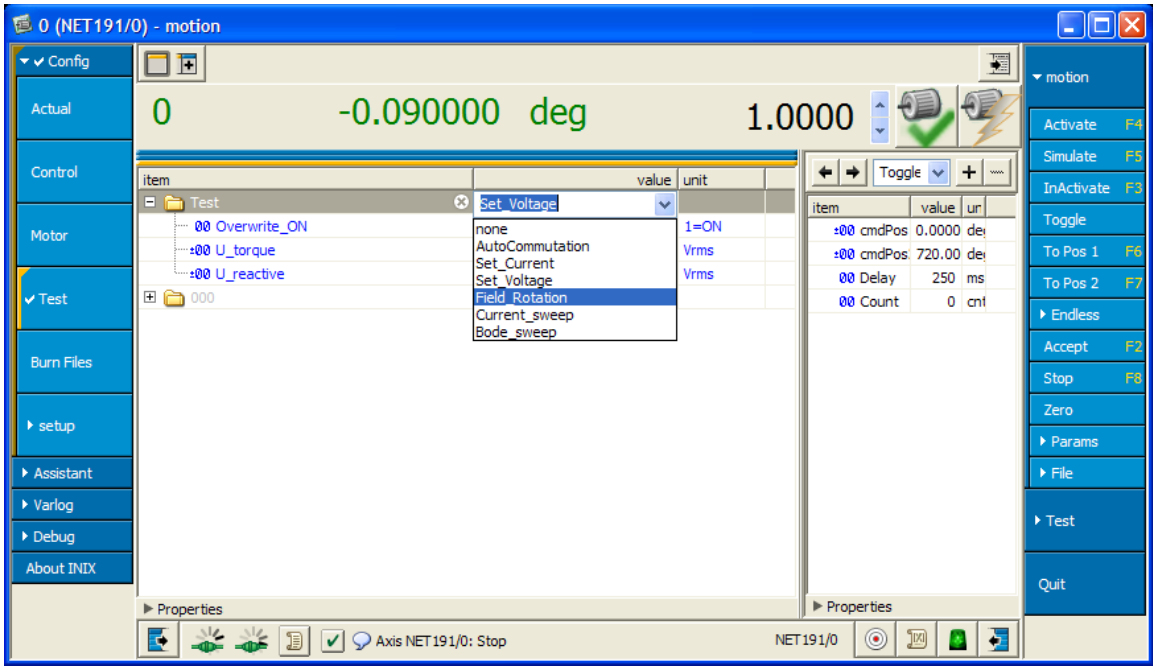


Fig. 6: Test-Modi

AutoCommutation	Auto-commutation: automatic adjustment of the field offset.
Set_Current	Current mode: setting of a constant active or idle current. Commutation is not yet required in this mode.
Set_Voltage	Voltage mode: setting of a constant active or idle voltage. Commutation must have been carried out for this mode.
Field_Rotation	Field mode. Commutation is not yet required in this mode.
Bode_sweep	Plotting of a bode diagram.

The individual test modes are described in detail in the following sections.

3 Motor configuration

3.1 Version control for motor configuration

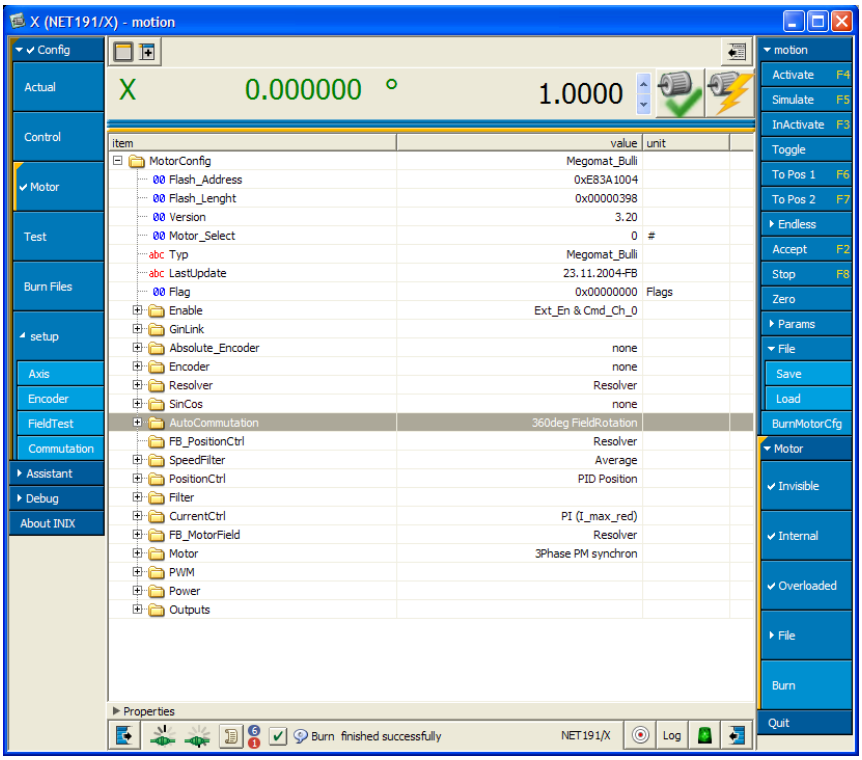


Fig. 7: INIX Motion

LastUpdate This parameter is used for the customer's version of the file.

Type Enter motor type here.

Version: Internal version number of motor configuration file structure.

> The parameter Version must not be adjusted! This is the version number of the motor configuration file structure.
>
> **This number is needed by the drive software in order to interpret various versions of the file (older generations of controllers).**

In the case of multiple drives or motion boards such as SAC3x3, MAX2, MAX4 and AX4 with a firmware version older than Rev. 6.413 - 847 (Rev. 6.4D - 847), a motor configuration file must be loaded into the drive for all motors.

If only one motor configuration file has been loaded, the analogue and digital I/Os will not be affected!

From firmware version Rev. 6.413 – 847, a motor configuration file must at least be loaded for axis 0.

3.2 Firmware Version

The firmware version can be viewed via the motion tool or in Inco-Explorer:

In the motion tool under Burn Files deactivate all check boxes, then press Start . In theMessage-History the firmware version is displayed, along with the version of the motor configuration and controller configuration.

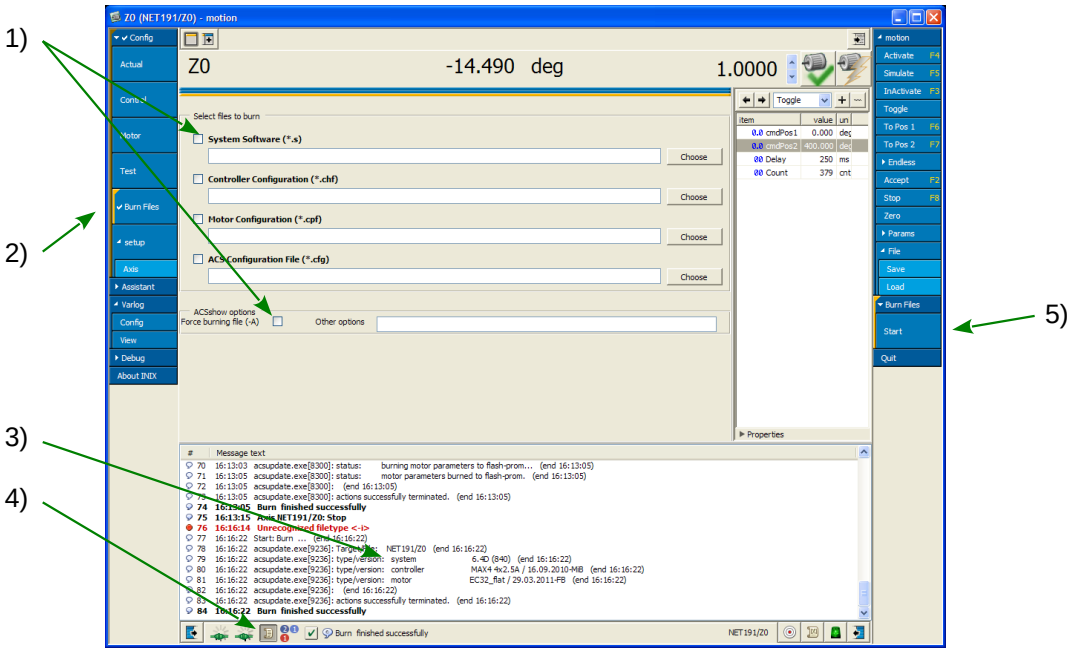


Fig. 8: Firmware version in the motion tool

1) Check-Boxen deactivate
2) Burn Files menu
3) File versions
4) Message-History Show window
5) Start Button for displaying file info

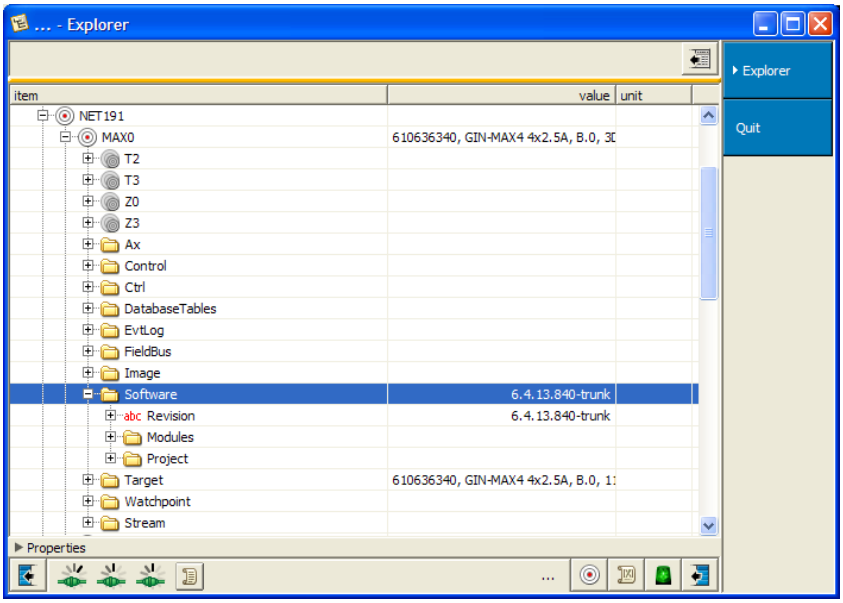


Fig. 9: Firmware version in Inco-Explorer

3.3 Absolut Encoder

The refresh rate of all absolute encoders in 1ms.

3.3.1 Specification of absolute encoders

Selection of encoder systems:

Endat	max. 26-bit data (increments per revolution + revolutions) Clock frequency: 400kHz
Hiperface	max. 48-bit data (increments per revolution + revolutions) Clock frequency 9600 bits / s
SSI	max. 32-bit data (increments per revolution + revolutions) Clock frequency 400kHz
Used as Encoder	An incremental encoder can also be connected to the absolute encoder input. This applies to the following drives: SAC3x3 und AX4 This allows even fast digital encoder signals to be processed. For more details, see Hardware-Manual-Motion-Boards.pdf and Hardware-Manual-SAC3.pdf The configuration of the encoder stays in the sub-folder Encoder.

3.3.2 Configuration of absolute encoders

Inco path for configuring absolute encoders: Ctrl.MotorConfig.Absolute_Encoder

IncPerMotorTurn	Number of increments per motor revolution. For linear motors, enter the number of increments per pole spacing. This parameter is only required for auto-commutation.
BitsPerEncTurn	Number of bits per encoder revolution. Take this value from the encoder data sheet.
NrOfEncTurns	For multi-turn encoders: enter the number of turns here. For single-turn encoders, you must enter 1 here.
Flags: GrayCode	0 for encoder without gray code 1 for encoder with gray code
direction	0 for CW 1 for CCW

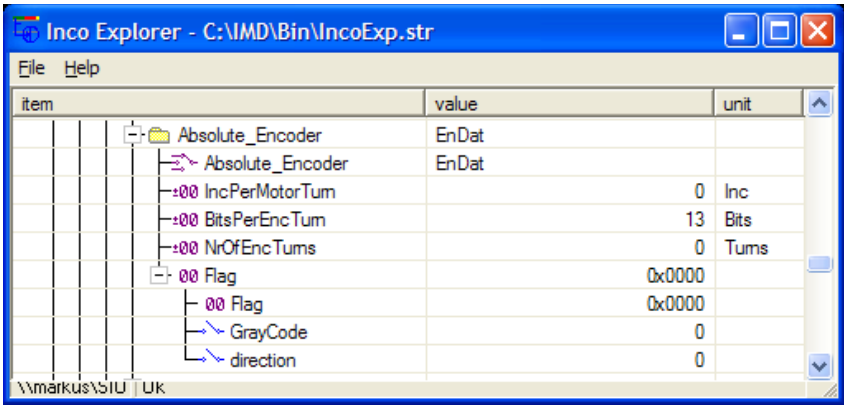


Fig. 10: Absolute encoder configuration

3.3.3 Actual values for absolute encoders

Inco path for actual values of absolute encoders: Ctrl.Actual.Absolute_Encoder

Status	0	Request data
	1	Read data
	2	Reset
	3	Hardware Reset
Turns	Number of turns (only for multi-turn encoders)	
Position	Position in increments	
Error	Error code; see below	
Ok_Reads	Number of messages received	
Error_Reads	Number of messages with errors received	

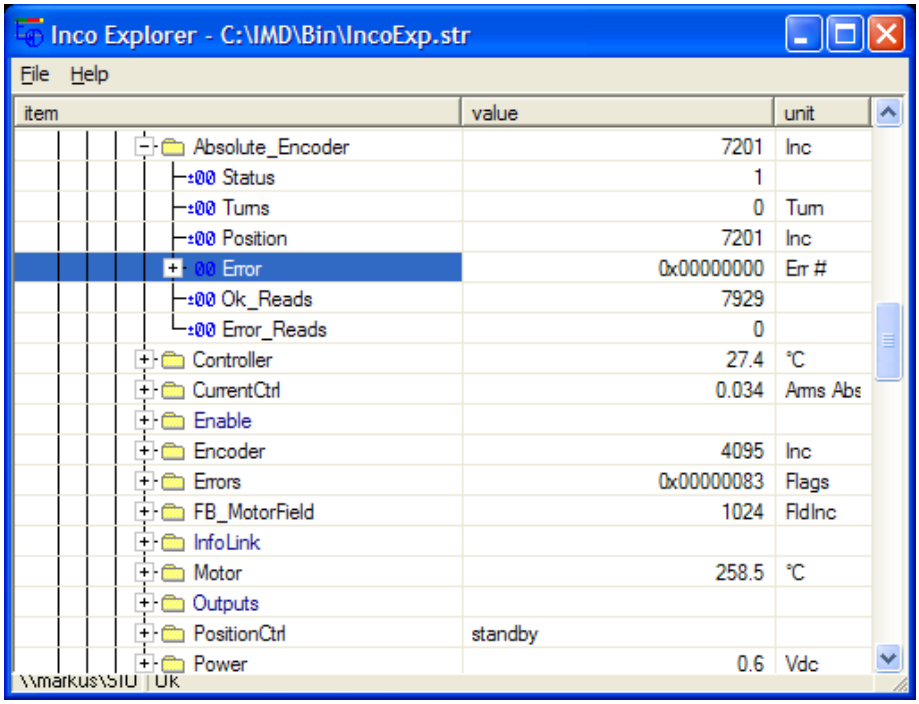


Fig. 11: Actual values, absolute encoder

3.3.4 Errors

Error flags from Endat

Busy	Interface is busy
Overrun	Error from UART
Framing	Error from UART
Parity	Error from UART
Timeout	The entire message was not received during the timeout time of 1ms
Alarm	Encoder alarm bit; see specification of encoder
CRC	Checksum error; occurs if, for example, the number of bits is incorrect.

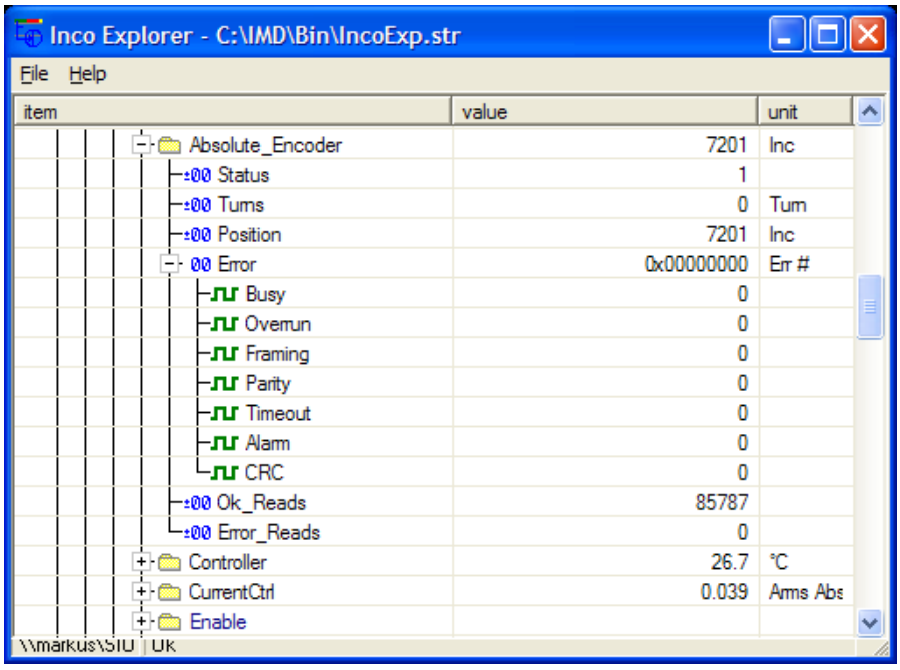


Fig. 12: Absolute encoder error, Endat

Error flags from Hiperface encoder

Busy	Interface is busy
Overrun	Error from UART
Framing	Error from UART
Parity	Error from UART
Timeout	The entire message was not received during the timeout time of 1ms
Alarm	Encoder alarm bit; see specification of encoder
Csum	Checksum error; occurs if, for example, the number of bits is incorrect.

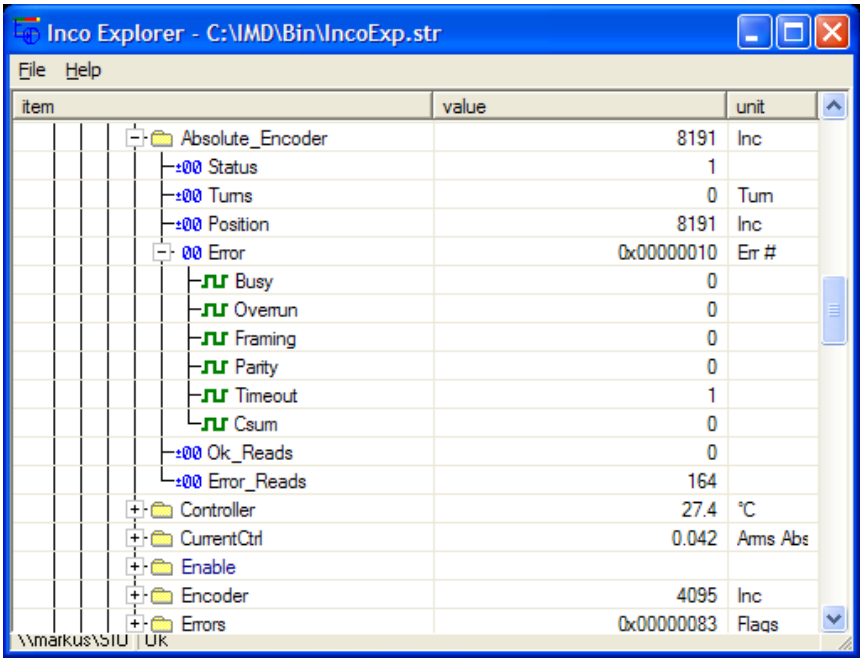


Fig. 13: Absolute encoder error, Hiperface

Error flags from SSI (synchronous serial interface)

Busy — Interface is busy

Ok_Reads — The SSI protocol cannot carry out a check such as parity bit or CRC. For this reason, all read-in values are recognised as ok, even if no encoder is plugged in.

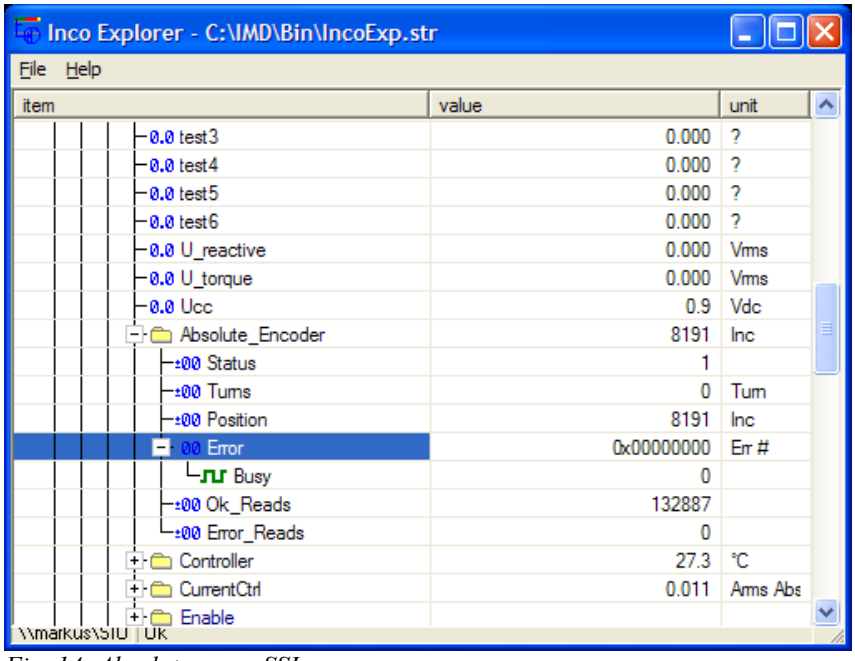


Fig. 14: Absolute error, SSI

3.3.5 Examples

Endat interface with 26 bits, single-turn, no auto-commutation:

Absolute_Encoder	**EnDat**
IncPerMotorTurn	0
BitsPerEncTurn	26
NrOfEncTurns	1
Flags:	
GrayCode	0
direction	0

SSI interface with 24 bits, multi-turn, with gray code, no auto-commutation:

Absolute_Encoder	**SSI**
IncPerMotorTurn	0
BitsPerEncTurn	12
NrOfEncTurns	4096
Flags:	
GrayCode	1
direction	0

3.4 Encoder

3.4.1 Configuration of incremental encoder at encoder feedback

Path in Inco-Tree for incremental encoder configuration: Ctrl.MotorConfig.Encoder

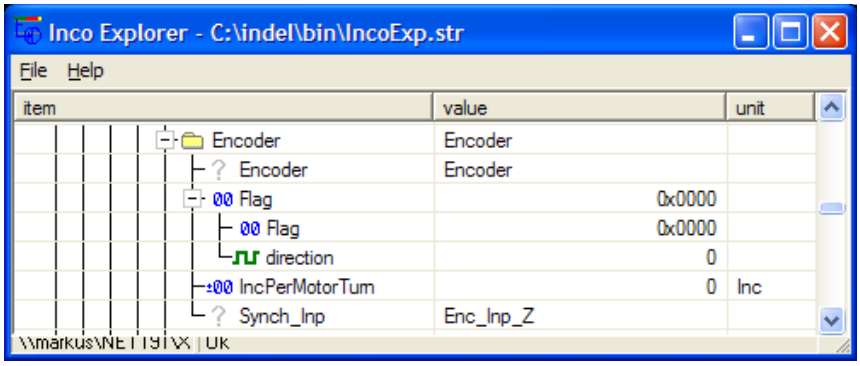


Fig. 15: Encoder

Encoder	Encoder, zero pulse from incremental encoder
Ref_Inp time	Only zero pulse
IncPerMotorTurn	Number of increments per motor revolution. Including 4-quadrant evaluation: in the case of an encoder with 1024 strokes, 4096 must be entered here.
Synch_Inp	no zero pulse or zero pulse from incremental encoder
Flag.Direction	Flag = 0: CW Flag = 1: CCW, inverted counting direction

3.4.2 Configuration of incremental encoder at SinCos interface

If an incremental encoder is connected to and run on the SinCos interface, the configuration is carried out on the SinCos interface (see 3.6.2).

3.4.3 Actual values for incremental encoders

Path in Inco-Tree for actual values for the incremental encoder: Ctrl.Actual.Encoder

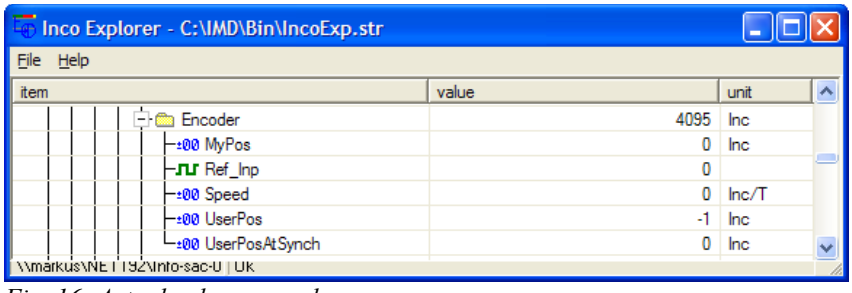


Fig. 16: Actual values, encoder

3.5 Resolver

The position of the resolver is read in at 16 bits, fixed.

3.5.1 Configuration of resolver

Path in Inco-Tree for the resolver: Ctrl.MotorConfig.Resolver

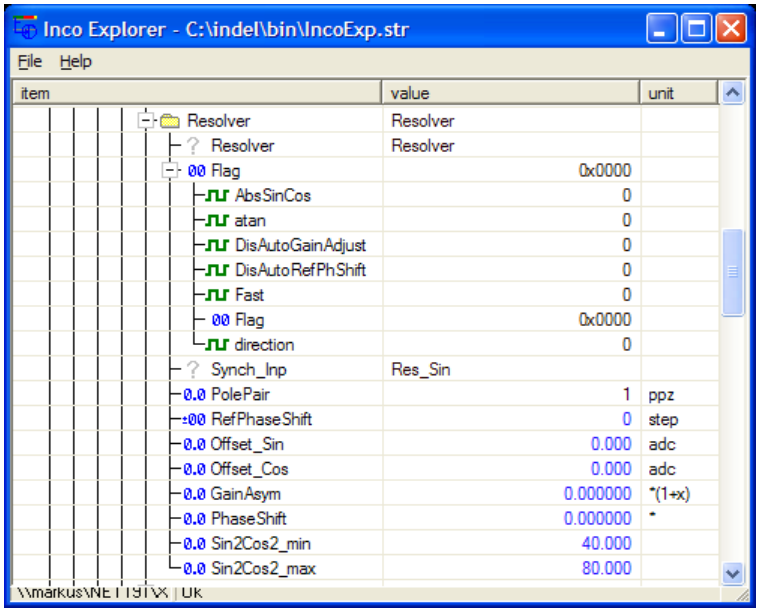


Fig. 17: Resolver

PolePair		Number of pool pairs
Sin2Cos2_max		Maximum of sine2 Cosinus2. Default value: 80
Sin2Cos2_min		Minimum of sine2 Cosinus2. Default value: 40
DisAutoGainAdjust		The amplitude of the resolver generator (reference output) is automatically set by the controller in such a way that Sin2Cos2 is around 60 wherever possible. Correction is switched off with this flag, and primarily only serves the purposes of troubleshooting.
DisAutoRefPhShift		The phase shift of the resolver generator (reference output) to the measured sin and cos signals is automatically measured and corrected by the controller. Correction is switched on with this flag, and primarily only serves the purposes of troubleshooting.
Flag	Direction	Direction of rotation of the resolver. (Still wire correctly!)
	Fast	0: Low-pass filter on a resolver with a cut-off frequency of approx. 400Hz 1: Low-pass filter on a resolver with a cut-off frequency of approx. 600Hz
	atan	0: no impact 1: low-pass filter switched off. The arctang is calculated and applied directly from the sine and cosine track.

If the low-pass filter is set higher or switched off, more interference may occur. You should always work with the deepest filter possible.

Note
When working with a resolver, the sampling rate set in the servo drive must be 12kHz as a maximum. Otherwise there will be too much attenuation of the level of the sine and cosine values.

3.5.2 Actual values for resolvers

Path in Inco-Tree for actual values for the resolver: Ctrl.Actual.Resolver

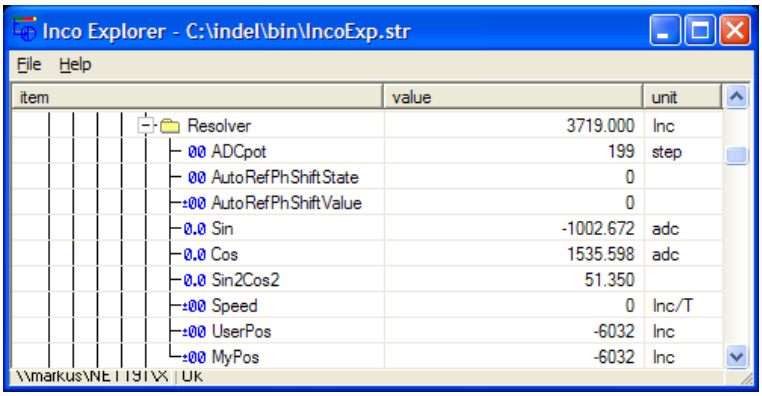


Fig. 18: Actual values, resolver

In the event of a “resolver error”, the Sin2Cos2 value must be checked first. This must be within the configured limits.

3.6 SinCos

3.6.1 Configuration of SinCos

Path in Inco-Tree for SinCos: Ctrl.MotorConfig.SinCos

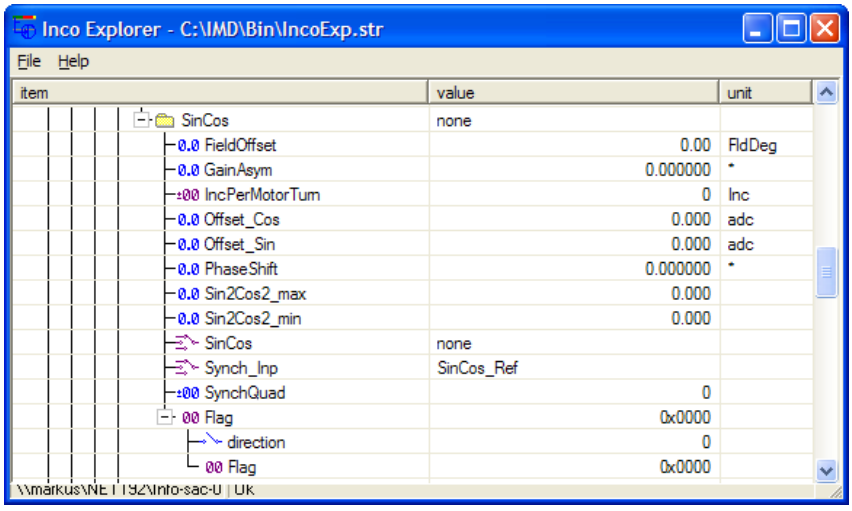


Fig. 19: SinCos

IncPerMotorTurn	Number of increments per motor revolution Number of periods* 1024. The resolution of the sine cosine values is 10 bit. Example: SinCos encoder with 2048 strokes: 2048 * 1024 = 2'097'152 Inc/T	
Sin2Cos2_max	Maximum of sine2 Cosinus2. Default value: 80	
Sin2Cos2_min	Minimum of sine2 Cosinus2. Default value: 20	
Flag	Direction	Direction of rotation of the SinCos encoder

3.6.2 Configuration of incremental encoder at SinCos interface

If an incremental encoder is connected to and run on the SinCos interface, the configuration is carried out on the SinCos interface.

IncPerMotorTurn	Number of increments per motor revolution * 1024 (Without 4-quadrant resolution)	
Sin2Cos2_min	Minimum of sine2 Cosinus2. Default value: 20	
Sin2Cos2_max	Minimum of sine2 Cosinus2. Default value: 400	
Flag	Direction	Direction of rotation of the SinCos encoder

Warning: As the incremental encoder is now configured at SinCos, the SinCos must also be defined for position control and GinLink feedback.

3.6.3 Actual values for SinCos

Path in Inco-Tree for actual values for SinCos: Ctrl.Actual.SinCos

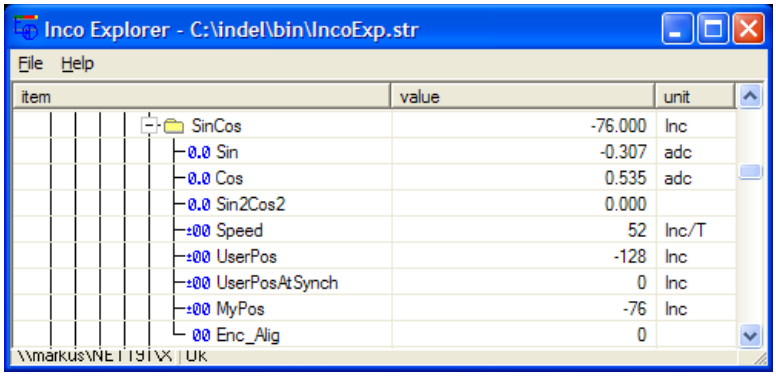


Fig. 20: Actual values, SinCos

In the case of a “resolver error” or “SinCos error”, the Sin2Cos2 value must be checked first. This must be within the configured limits.

3.7 Checking the sine cosine / resolver level

Change the Inco path to the actual values for the resolver/SinCos:
Ctrl.Actual.Resolver
Ctrl.Actual.SinCos

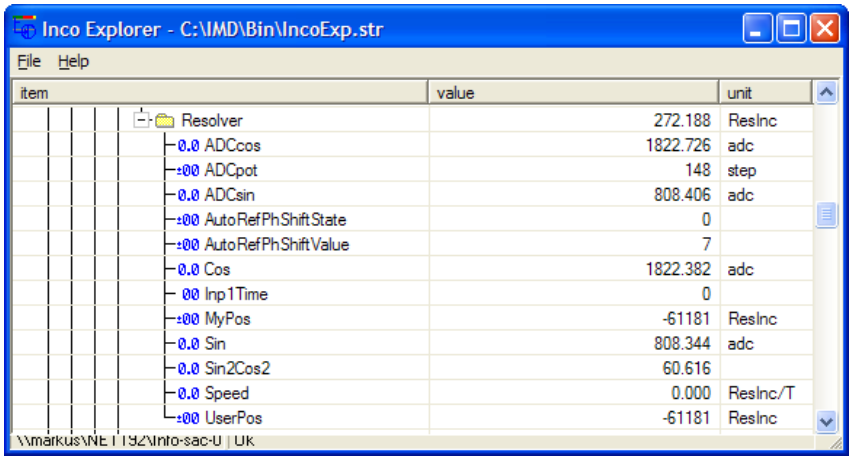


Fig. 21: SinCos

The following parameters must be right:

Supply		Resolver		SinCos	
		Maximum	Minimum	Maximum	Minimum
ADCcos	bits	2048	-	2048	-
ADCsin	bits	2048	-	2048	-
ADCpot	bits	255	-	255	-
Sin2+Cos2	(V^2)	80	40	80	20

Should certain parameters be near their limit, this can be the cause of sporadic errors in the encoder.

3.8 Auto-commutation

3.8.1 Configuration of the auto-commutation

Inco path for auto-commutation: Ctrl.MotorConfig.AutoCommutation

There are various procedures available for auto-commutation:

360deg FieldRotation	Auto-commutation with 360° field rotation
UVW pulse	3 voltage pulses
Two-Phase Stepper	Voltage pulse, motor moves, also suitable for 3-phase motors with suspended load
Hiperface	Digital interface, motor does not move
EnDat	Digital interface, motor does not move
SSI	Digital interface, motor does not move

Flags

ON_If_Ok=1	If the commutation was successful, the axis remains active. Otherwise the output stage is switched to inactive.
Again=0	In normal operation, the axis is switched to active after commutation. The operating mode changes from Commutation to Active; Again=0 When commissioning, it is necessary to repeat the commutation a few times. To do this, the flag is set to Again = 1 so that the operating mode always remains AutoCommutation.

Never activate the axis (either in simulation mode or active mode) if the commutation is not working perfectly!

3.8.2 Auto-commutation with UVW pulse

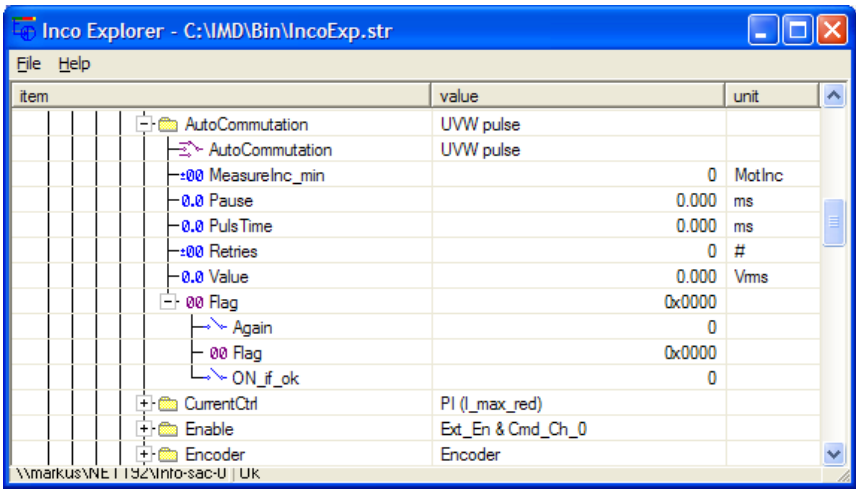


Fig. 22: Auto-commutation, UVW pulse

The UVW pulse procedure generates a short voltage pulse at each phase. The field offset is calculated from the resulting movement.

With this commutation method, the motor moves very little. This method is suitable for dynamic motors with low mass and low friction.

MeasureInc_min 20 ... 30 [MotInc]
The sum of the three movements of U, V, W must be greater than MeasureInc_min so that the commutation is successful.

Pause 20 ... 100 [ms]
Pause between the individual pulses

PulsTime 1 ... 5 [ms]
Pulse duration of the three pulses

Retries 0 ... 3 [#]
Retries in the case of unsuccessful commutation attempts.

Value=1 ... 3 [Vrms]
Voltage value for the pulse

3.8.3 Auto-commutation with the two-phase stepper method

This type of commutation is suitable for stepper motors with feedback and for 3-phase motors with a suspended load. The motor moves jerkily by up to 60 field degrees.

This method is also suitable for motors with an incremental encoder with low resolution (500 strokes per revolution).

The voltage pulse is applied to the motor for the time PulsTime. This causes the rotor to be dragged to a defined position.

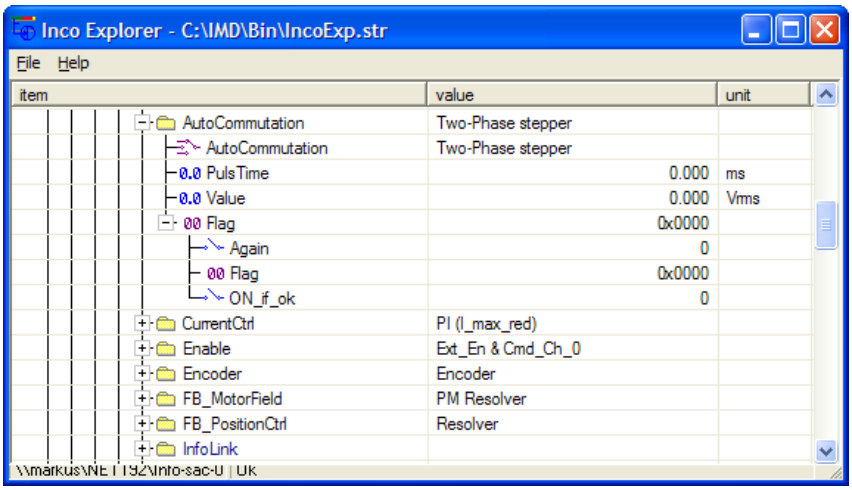


Fig. 23: Auto-commutation Two-Phase Stepper

PulsTime 1000 [ms]
Pulse duration for the voltage pulses

Value 0.5 ... 3 [Vrms]
Voltage value for the pulse. Choose the voltage in such a way that IMAX/2 ... IMAX flows.

3.8.4 Auto-commutation with absolute encoders

In the case of axes that must not move before activation, or axes that are held in place with a securing brake, the auto-commutation can be carried out using an absolute encoder.

- Hiperface
- EnDat
- SSI Synchronous serial interface

3.8.5 Auto-commutation with 360° field rotation

This type of commutation is the most precise of them all and should be used wherever possible. The motor moves by 360 field degrees. In the case of motors with one pole pair, this corresponds to one motor revolution; in the case of motors with 10 pole pairs, this corresponds to 0.1 revolutions.

This method is suitable for axes with a large load and high friction, as well as for highly dynamic axes.

Flag.Direction
0, 1 [ ]
The direction flag can be used to define the direction in which the field should be turned (positive or negative direction)

Flag.not_unwind
0, 1 [ ]
Following commutation, the axis can be disconnected for a period of approx. 380ms
This prevents the motor from becoming overloaded
if the commutation is carried out against a mechanical restriction.
Do NOT set this flag so that the unloading (unwinding)
druchgeführt wird.
See section 3.8.6 “Not Unwind” flag (360° commutation).

Max_Delta
10 ... 20 [deg]
Following evaluation of the position information, the difference of all offset angles of the measured segments must not exceed the Max_Delta value.

TurnTime
1000 ... 4000 [ms]
Time for field rotation, 360° in positive and 360° in negative direction.

The time should be adapted to match the pole pairs of the motor. Allow motors with just one pole pair to rotate at at least 2000ms.

Configure longer times for motors with a large load.

Value
1 ... 3 [Vrms]
Voltage value for the field rotation. Choose the voltage in such a way that I_{NENN} is not exceeded.

3.8.6 “Not Unwind” flag (360° commutation)

The flag not_unwind allows the behaviour to be influenced immediately after the 360° commutation.

If the flag is not set and a pause of 380ms is entered after the 360° field rotation, the active current is set to zero during this time. The position control is activated after the pause and a certain holding current is set.

This mode (not_unwind=0) is not suitable for suspended Z axes, as the axis cannot be held during the pause.

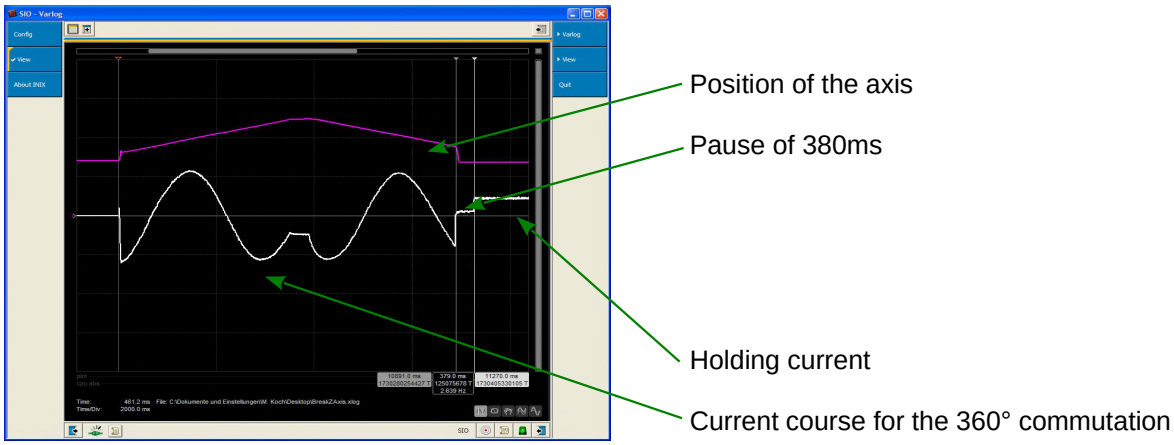


Fig. 24: Flag not_Unwind not set

If the flag is set, there is no pause to relieve the axis. If the axis is pressed against a mechanical restriction after the commutation, an increased current may be applied to the motor after the commutation. See fig. 25.

The mode with the set not_unwind flag requires the I2t control to be optimally set. Otherwise there is a risk that the motor will become overloaded.

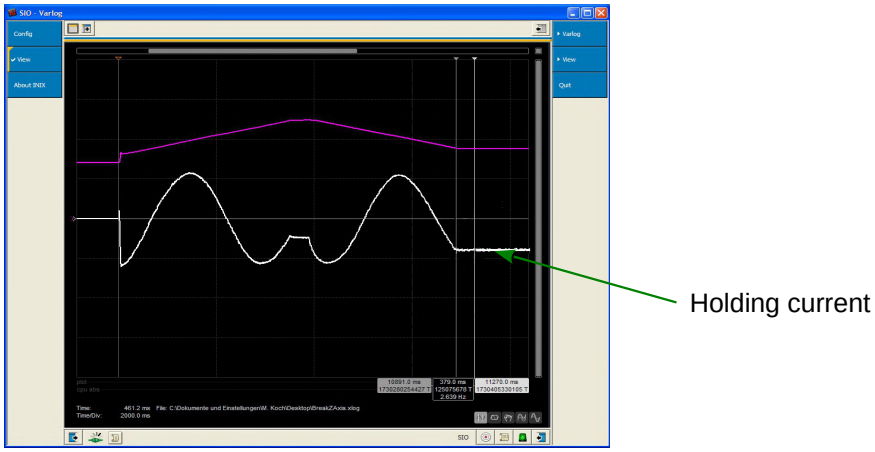


Fig. 25: Flag not_Unwind set

3.8.7 Auto-commutation with hall sensors for Maxon motors

HallInp_Seq — Expected sequence of the hall sensor input signals

HallInp_0 — Bit number of the first hall sensor input. The three hall sensors must be connected one after another:
e.g. input 0, 1, 2

Wire	hall sensor 1	to input Inp No
Wire	hall sensor 2	to input Inp No + 1
Wire	hall sensor 3	to input Inp No + 2

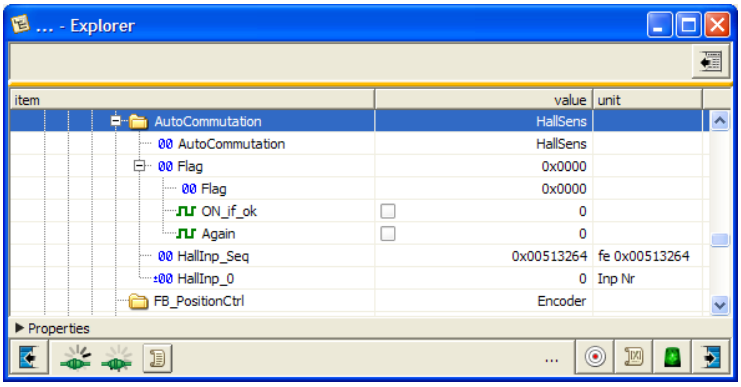
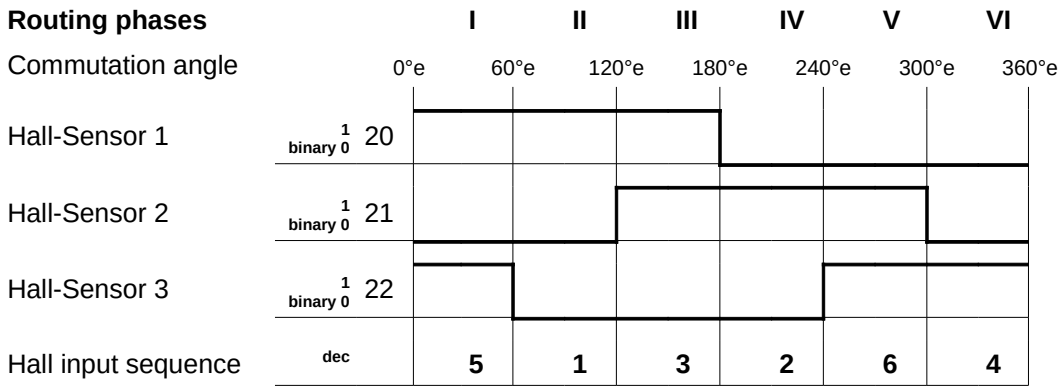


Fig. 26: Hall sensor commutation

Block commutation



The commutation angle is given in field degrees (°e). Warning: in motors with more than one pole pair, the degrees on the motor shaft do not match the field degrees.

Example:
In a motor with 7 pole pairs, the motor field rotates 7 x 360°e = 2520°e for one revolution of the shaft.

Configuration of the block commutation

These boards offer block commutation with hall sensors: MAX2, MAX4, AX-4x2, AX-4x4.

An initial rough commutation can be carried out (approx. +-30degrees exactly) with the help of the hall sensors.

An exact commutation value will also be stored at the encoder ref mark. The final commutation will then be determined the first time the mark is run over. This means that two commutations need to be prepared.

Preparation

Commission the motor fully. The following points must be right before the block commutation can be configured:

- The motor's direction of rotation must be correct:
  Motor winding 1 to U, motor winding 2 to V, motor winding 3 to W
- The direction of rotation of the motor and the encoder system must be correct, i.e. the direction flags for the motor and the encoder in the motor configuration must be correct
- Invert hall sensor inputs if there are no open collector outputs available on the hall sensor:

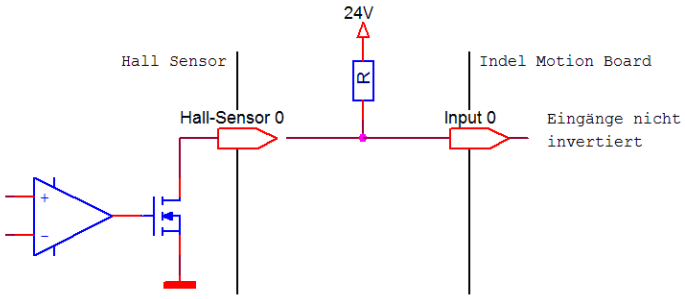


Fig. 27: Wiring with non-inverted inputs

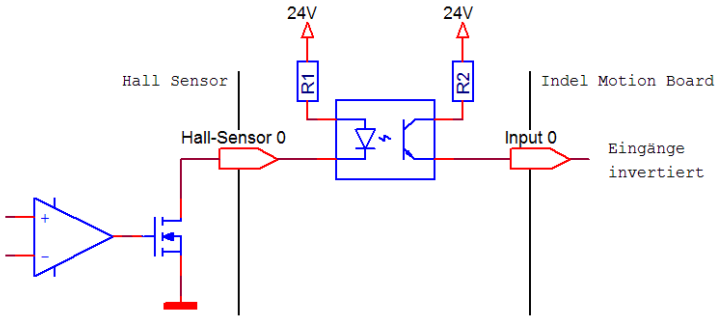


Fig. 28: Wiring with inverted inputs

The inputs can be inverted in the dt2 configuration; to do this, set the "Inverted" bit.

Procedure

1. Precise commutation at synch mark (zero pulse)

To do this, determine the field offset for the motor as precisely as possible:
e.g. with the mode 360deg FieldRotation followed by fine-tuning in current or voltage mode as per the manual) See section: 10.16.3 Adjusting the resolver offset by hand

- Auto-commutation, e.g. with 360° method, fine-tuning by hand

2. Determine field offset at reference mark

- Set Ctrl.Actual.FB_MotorField.SynchDone = 0
- Set Ctrl.MotorConfig.FB_MotorField.FieldOffset_at_Ref = -1
- Set flag Ctrl.MotorConfig.FB_MotorField.Flag.FieldSynchWithRefInp = 1

- move the motor over the synch mark (zero pulse), either by hand or by allowing the motor to turn slowly.
  -> the flag Ctrl.Actual.FB_MotorField.SynchDone becomes 1
  -> in Ctrl.MotorConfig.FB_MotorField.FieldOffset_at_Ref
  the exact field offset is stored and can thus be burned.

3. Read out hall sensor sequence

The sequence of the hall sensors changes depending on how the motor is connected, or how the direction of rotation is configured.

- select commutation modeCtrl.MotorConfig.AutoCommutation = HallSens
- configure hall sensor inputs: Ctrl.MotorConfig.AutoCommutation.HallInp_0 = Inp_Nr
  The three hall sensors must be connected one after another.

- Allow the motor to run in test mode Field_Rotaion
  At the same time, record the following parameters with the Varlog:
  - Ctrl.Actual.FB_MotorField
  - Ctrl.Actual.AutoCommutation.HallSens_1
  - Ctrl.Actual.AutoCommutation.HallSens_2
  - Ctrl.Actual.AutoCommutation.HallSens_3

- From -2048 Inc e (field angle), read out the sequence of the inputs from left to right:
  one of the following sequences should appear within a field rotation, depending on the motor direction flag:
  - 0x0062'3154
  - 0x0051'3264

  The following value must always be entered in the hall input sequence for the wiring U, V, W – motor winding 1, 2, 3:
  Ctrl.MotorConfig.AutoCommutation.HallInp_Seq = 0x0051'3264

- Set the parameter Ctrl.MotorConfig.AutoCommutation.FieldOffset = 0 .

- Set Ctrl.Actual.FB_MotorField.SynchDone = 0

Hall sensor sequence

Motor-Direction Flag: 1
Motor connections: U, V, W – motor winding 1, 2, 3

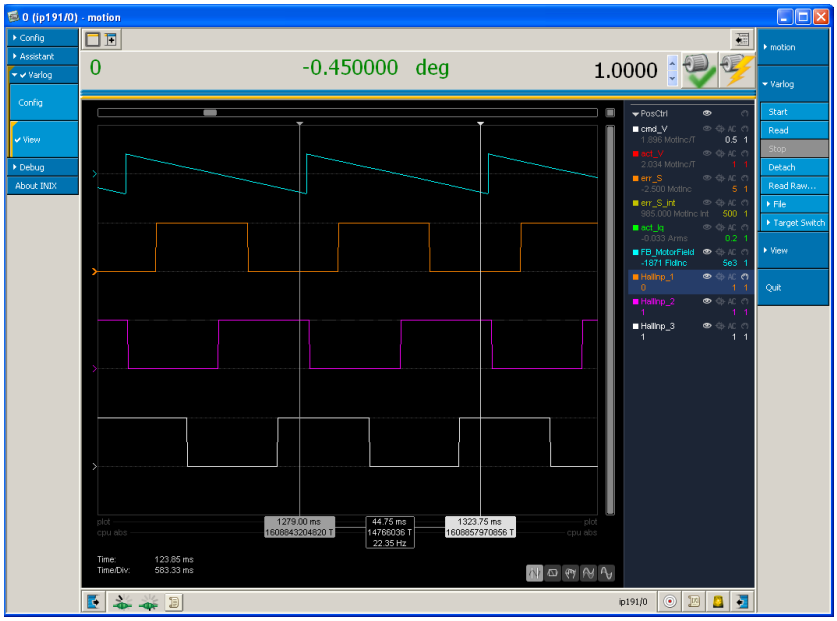


Fig. 29: Hall sensor sequence, standard direction of rotation CCW

Motor-Direction Flag: 0
Motor connections: U, V, W – winding 1, 2, 3

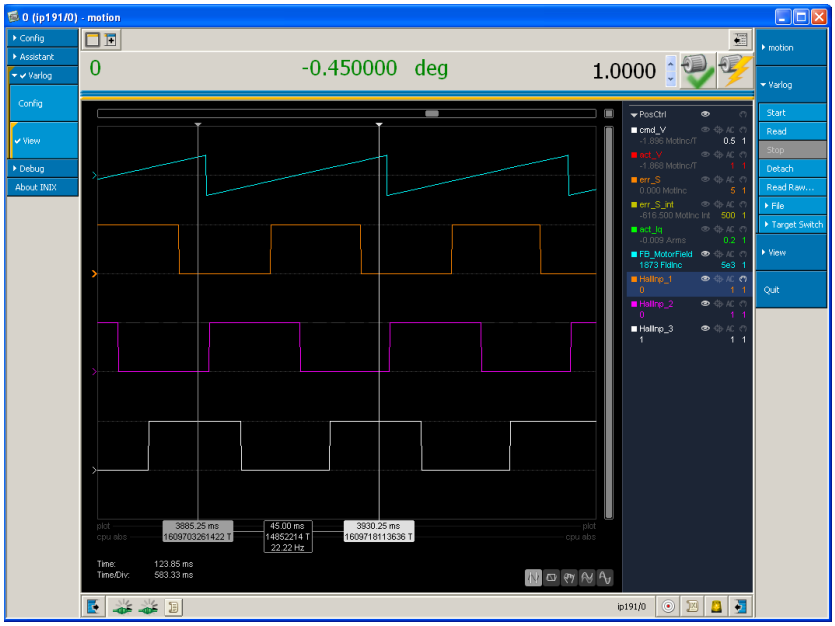


Fig. 30: Hall sensor sequence, standard direction of rotation CW

Channel 1:	light blue	Field angle in increments
Channel 2:	orange	Hall sensor input 1
Channel 3:	pink	Hall sensor input 2
Channel 4:	white	Hall sensor input 3

4. Test the block commutation

- In order to test the block commutation, the flag needs to be set to zero temporarily:
  Ctrl.MotorConfig.FB_MotorField.Flag.FieldSynchWithRefInp = 0
- Burn motor parameters to the flash prom
- Switch off/on several times with various starting positions
  The axis must be able to at least reach the
  synch mark each time. It may be necessary to use reduced PID settings.

> Caution: If the commutation is not right, high currents will flow into the motor and it may turn in the wrong direction! The IMAX can be reduced to protect the motor. The value for I2t-down can be set to 0.98.

- When everything is working, do not forget to put the flag back into the correct position and burn it. Ctrl.MotorConfig.FB_MotorField.Flag.FieldSynchWithRefInp = 1

Possible problems

If the hall sensors do not display exactly 60°e sectors, the accuracy of the commutation deteriorates considerably. This means that the route many not be able to be travelled to the synch mark without the motor resonating.

In order to combat this problem, an adapted PID parameter set can be used to reach the first synch mark:

- reduce kP to 50 ... 80%
- reduce Pos_Int_Max to 1000 ... 5000

In order to check the evenness of the segments, a log should always be made with the individual hall sensor inputs (HallInp_0, 1, 2) and the field angle. Drive the axis at a constant speed.

All signal edges must have the same spacing. In fig. 29 and 30 the segments are very irregular.

3.8.8 Actual values for auto-commutation

Inco path for actual values for auto-commutation: Ctrl.Actual.AutoCommutation
The actual values are shown as follows, regardless of the auto-commutation method:

UVW method

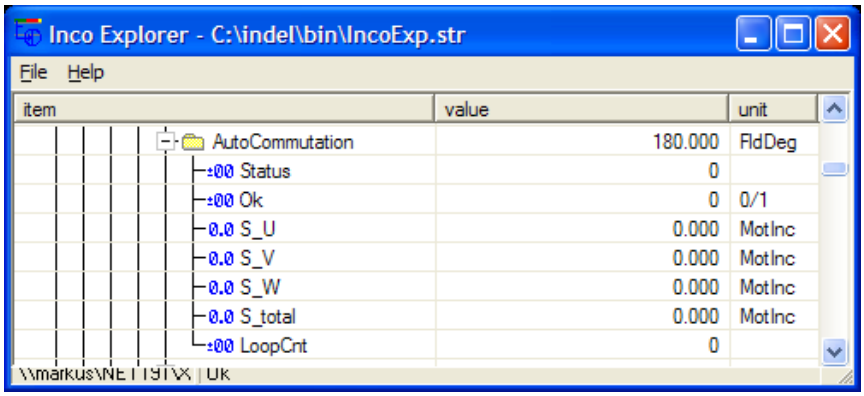


Fig. 31: Actual values, auto-commutation UVW

Status	status of the auto-commutation
OK	if the commutation was successful, this value is 1, otherwise 0
S_U, S_V, S_W	create the route for the individual U,V,W pulses
S_Total	entire route covered during the commutation. If the S_Total exceeds the value of MeasureInc_min , the commutation is successful.
LoopCount	The loop counter can be used to determine how many attempts the commutation needed before it was successful. See Retries.

Two-phase stepper and absolute encoder method

These two procedures always end successfully.

360° field rotation method

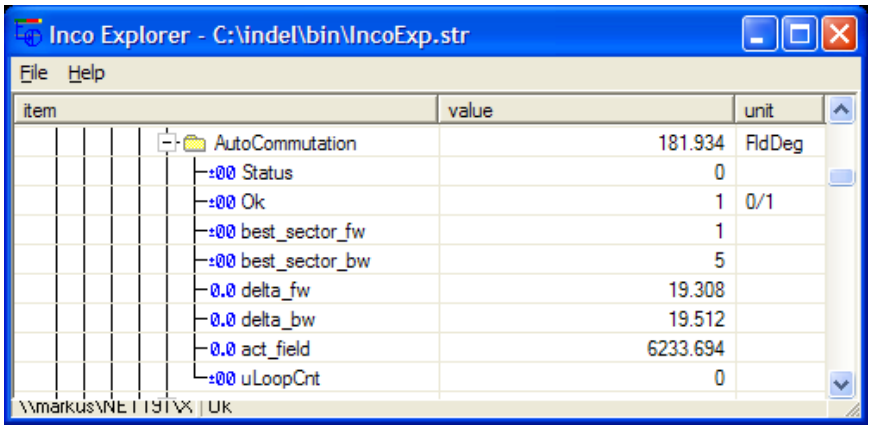


Fig. 32: Actual values, 360° commutation

The field angle under auto-commutation is assumed for the field control in the case of successful commutation.

3.9 Current controller: Current Control

3.9.1 Current controller variants

The target value for the current controller always comes from the upstream PID controller (speed and/or position controller). This target value can be adapted with the following variants.

Output 1 corresponds to the bit I_Red (current reduction). The bit I_Red is set by the fieldbus controller via the fieldbus.

none

PID # CmdIq 1 = 1: Ausgang 1 = 0: Target value for current controller is output from PID controller Output

Target value is output from PID controller; beneath this is the torque curve CmdIq. Any torque curve can be driven with this function.

PI Target value for current controller is output from PID controller (position controller), output 1 is not considered

PI (I_max_red) Ausgang 1 = 0: Current controller limited to Imax Output 1 = 1:
Current controller controls on Ired -> constant torque **Default setting**

PID (CmdIred) Ausgang 1 = 0: Target value for current controller is output from PID controller
Ausgang 1 = 1: Current controller limited to torque curve Ired, operating mode "Ired"

PID + CmdIq The curve CmdIq is above the target current (pilot control).

CmdIq Apply current curve, without position control

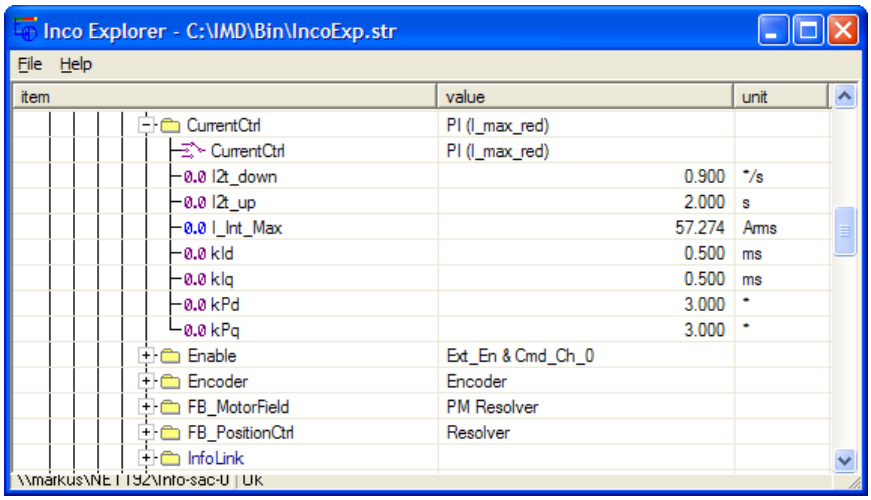


Fig. 33: Current controller

3.9.2 Current controller parameters

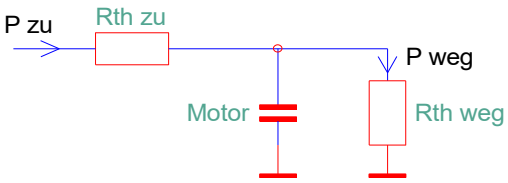
I_Int_Max [Arms]	Limitation of the I component in the current controller. **Dimensioning: I_Int_Max = 3 mal I_{MAX}**
Overload protection I2t	
I2t [%]	Shows how much of the power dissipation that the motor absorbs can be re-emitted within the set time constant (I2t_down).
I2t_up_run: [s@I_max]	Time constant for the absorbed power dissipation of the motor while the motor is moving.
I2t_up_halt: [s@I_nom]	Time constant for the absorbed power dissipation of the motor while the motor is at a standstill.
I2t_down: [*/s]	Unloading behaviour of the heat stored in the motor (heat emission via housing, cooling, etc.).
kId [ms]	I component of idle current control
kPd [*]	P component of idle current control
kIq [ms]	I component of active current control
kPq [*]	P component of active current control

3.9.3 I²t control

In order to protect the motor from overload, the power dissipation that the motor absorbs is integrated. The power dissipation that the motor emits via the housing and any other ventilation holes is always deducted from this sum. The remaining value must not exceed a certain threshold.

> If there is no temperature sensor in the motor winding, the I2t control is the only protection for the motor against thermal overload!

Equivalent circuit diagram for I2t



Default values

I²t_up_run	[s]	0.5 ... 2	for motors with low overload capability
I²t_up_run	[s]	2 ... 4	for motors with high overload capability
I²t_down	[*]	0.90 ... 0.95	rotational motors
I²t_down	[*]	0.95 ... 0.98	linear motors, motors with poor heat emission

Loading behaviour

If, for the time I²t_up_run the current I_max is applied to the motor without it being cooled, the value for I2t =100%. This value applies whilst the motor is moving.
If, for the time I²t_up_halt the current I_nom is applied to the motor without it being cooled, the value for I2t =100%. This value applies whilst the motor is at a standstill.

Unloading behaviour

The value for I²t_down describes the temperature absorption in the motor:
In the case of a value of I²t_down = 0.9 the value for I2t becomes 10% smaller every second
In the case of a value of I²t_down = 0.98 the value for I2t becomes 2% smaller every second

Calculation principles

For the calculation of the I2t control, the I_max of the motor is used.
The bigger I_max is, the slower the I2t value increases.

If the I_max is much greater than I_nenn (I_ max / I_nenn > 5), this can lead to distortion of the calculation of the I2t value because the value for I2t may increase too slowly.

Linear motors

Special attention must be paid to cooling in the case of linear motors. As the coils are often sealed in epoxy, the emission of heat is not optimal. This can lead to slight heat build-up in the motor winding, particularly when the motor is not moving or is only moving very slowly. (Standstill, suspended loads, ...)

In the case of linear motors that only carry out very small movements, it must be ensured that a separate temperature sensor is available for each winding!

> The values for the I2t control must be determined empirically, as appropriate, and adapted to the physical conditions, such as motor data, cooling and driving profiles.
> **Precise data for overload operation must always be obtained from the motor manufacturer.**
> In the case of continuous operation of the motor at I2t = 100% the maximum motor temperature must not be exceeded.

3.10 Extern Enable

3.10.1 Configuration of the external enabler input

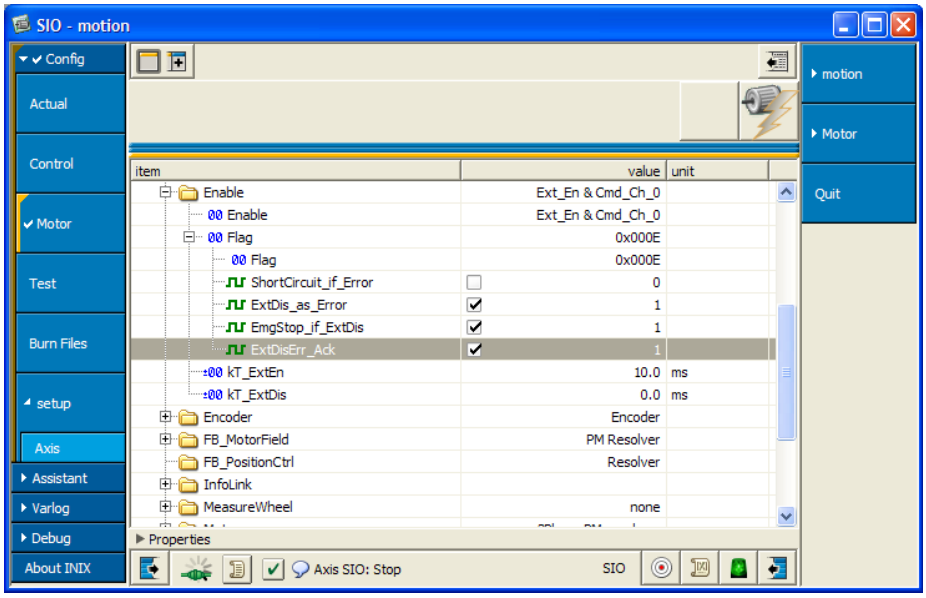


Fig. 34: Extern Enable

Ext + Ch_0 Controller becomes active when the external release is in place and is “active” and connected via target value channel 0

Ext + Ch_1 Controller becomes active when the external release is in place and is “active” and connected via target value channel 1

Ext + (Ch_0#Ch_1) Controller becomes active when the external release is in place and is “active” and connected via target value channel 0 or channel 1

kT_ExtDis Safety function allowing for the motor to be slowed down in the case of loss of the external release. The external release is removed with a delay by time kT_ExtDis. During this time, this motor can still be slowed down by the user software. This means that the external release can be accepted into the emergency stop circuit.

kT_ExtEn Software filter for the external release (10ms) to eliminate interference through spikes.

For STO (safe torque off) category 3 as per EN ISO 13849-1, the two safety inputs at X100 must be used.
The external enabler is a non-secure input.

For configuration of the safety pulse inhibitor see section: 4 Safety configuration.

3.10.2 Configuration of the emergency stop braking ramp

Flags — These flags are used to define how the axis will be switched off if the control goes into error.

A braking resistor may need to be used for braking in the event of an emergency stop. When braking, the motor feeds the kinetic energy back into the intermediate circuit!

ShortCircuit_if_Error = 1
The controller switches the axis back on after the control has gone into error. The axis is braked with Imax until it comes to a full stop.

If the flag ShortCircuit_if_Error = 0 is (not set) when the external enabler is lost, braking is carried out with an emergency stop ramp. The emergency stop is set out in the axis configuration.

Dangerous_OvrOn
This bit must be zero.

ExtDis_as_Error
When the external enabler is switched off, an error occurs. This bit must be set for operation with STO!
In the case of **GinLink**, the bit is **always** set and is no longer visible in the configuration

EmgStop_if_ExtDis
Emergency stop braking ramp is started after the external enabler is switched off. This bit must be set for operation with STO!
In the case of **GinLink**, the bit is **always** set and is no longer visible in the configuration

ExtDisErr_Ack — This bit must be set in order to prevent unwanted restart.
This bit is not secure!
In the case of **GinLink**, the bit is **always** set and is no longer visible in the configuration

Maximum braking power

The maximum permissible current for the short-circuit braking ramp is calculated as follows:

$$I_{MAX\ Drive} > 0.7 * (\frac{U_{CC}}{R_{PP}})$$

$I_{MAX\ Drive}$	Maximum permissible peak current of the drive	A
U_{CC}	Intermediate circuit voltage	V
R_{PP}	Phase-phase winding resistance	Ohm

The drive can be destroyed with the flag ShorCircuit_if_Error! The max. permissible short-circuit current is used for braking. The IGBTs/FETs can be destroyed after 1 ... 10 braking ramps.

This flag should only be used when the protection of the mechanics is more important than the protection of the drive!

This operating mode is excluded from the warranty!

3.10.3 Actual values for the external enabler

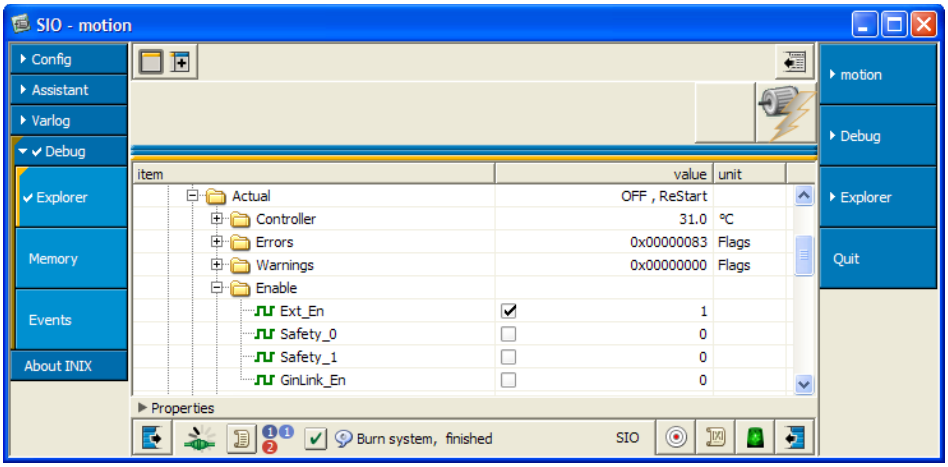


Fig. 35: Actual values, external enabler

Ext_EN	Status of input +En, -En at pin X15
Safety_0, Safety_1	Status of input 24V_R1, 24V_R2 at pin X100
GinLink_EN	Status of software enabler of GinLink fieldbus

3.11 Feedback Motor Field

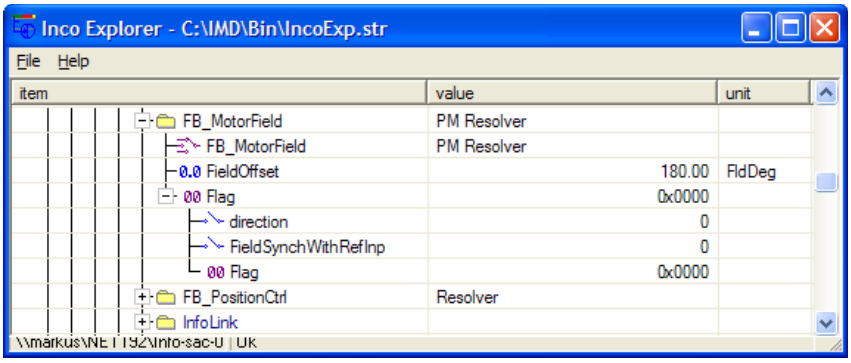


Fig. 36: Motor Field Feedback

Feedback Motor Field	
PM Resolver	Feedback system is resolver on permanent magnet synchronous or DC motor
AC I-model	Encoder on AC asynchronous motor
PM SynCos	Sine cosine interface
PM SynCos+Enc	Sine cosine interface with additional encoder to compensate for relative movements above this.
PM Encoder	Incremental encoder
Stepper without FB	No-feedback operation for stepper motors
Flag.Direction	Flag = 0: CW Flag = 1: CCW, inverted counting direction This flag can never be 1!!

No field control is required for DC motors.

A feedback system that is rigidly connected to the motor shaft is required for the field feedback. A gauge after a long spindle with a small incline or a feedback system after a gearbox is not suitable for field control. In this case, a second feedback system directly on the motor shaft is required.

The error of the field offset must not be greater than +-10 ... 15°!

Calculation example with a 30cm-long steel spindle with 5mm incline:

a =	12um/K/m	Coefficient of linear expansion of steel
dT =	10K	Temperature difference
l =	0.3m	Spindle length
pp =	5	Polpaare

5mm ≙ 360 motor ° ≙ 360 * 5 field °
5mm ≙ 1800 Feld °

Length change at dT=10K: 36um

36um ≙ 12.96 Feld °

→ The 36um length change of the spindle at a temperature difference of 10K corresponds to a field angle of 13°. This means that the entire tolerance for the field offset is already given.

To this, you must add deviations in the auto-commutation measuring method: +- 5..10°,
inaccuracy of the spindle (length): +- 5...10um,
inaccuracy of the spindle (incline): +- 5...10um,

In the worst case scenario, the entire error thus amounts to over 30 field °. The axis can no longer be operated in an optimal manner with this deviation. Idle currents flow, the motor heats up and the dynamics reduce.

In the case of even greater errors, the axis may start to move itself!

3.12 Feedback position control

Fig. 37: Position Control

Feedback for the position controller in the servo drive:

Resolver

SinCos

Encoder

Stepper without FB

3.13 GinLink

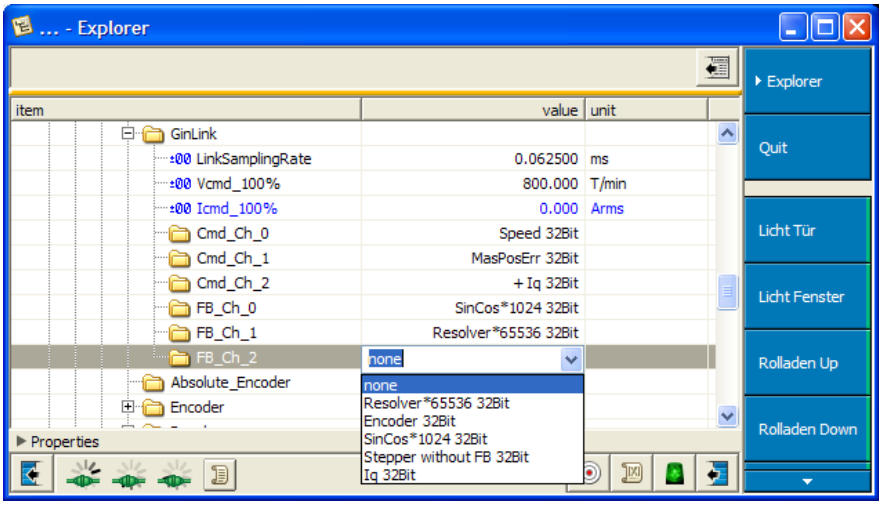


Fig. 38: GinLink

Vcmd_100%	Nominal speed of rotation of the control. Standardisation factor
LinkSamplingRate	Sampling rate of the superordinate controller or path control in the fieldbus master. This value is required in order to match the interpolation between two target values of the master to the sampling rate of the SAC controller, typically: 8 to 32kHz. The following sampling rates are provided via the fieldbus: 16 kHz 0.0625 ms 8 kHz 0.125 ms 4 kHz 0.25 2 kHz 0.5 1 kHz 1 0.5kHz 2 ms 0.25 kHz 4 ms
Icmd_100%	Nominal current for current control. Standardisation factor

Target value channels

There are a total of three target value channels: Cmd_Ch_0, Cmd_Ch_1, Cmd_Ch_2

Speed 32Bit	Target value is speed (for position control, the v is integrated into the controller.)
Iq Limit 32Bit	Limit of constant target current
Iq 32Bit	Curve form for target current
+ Iq 32Bit	Curve form for current lead value
MasPosErr 32Bit	Following error from superordinate position control

Actual value channels

There are a total of three actual value channels:
FB_Ch_0: Standard assignment for motor feedback, position detection on the SAM
FB_Ch_1:
FB_Ch_2: Standard assignment for active current Iq

23-bit values are transmitted on all channels.

Resolver*65536 32Bit	Resolver with 65536 increments per motor revolution as 32-bit wide value
Encoder 32Bit	Encoder as 12-bit wide value
SinCos*1024 32Bit	Number of SinCos periods per motor revolution * 1024 as 32-bit wide value
Stepper without FB	Calculated feedback for no-feedback operation of stepper motors. 4096 increments are returned per motor revolution.
Iq	Actual current value

In the case of analogue feedback systems, the resolution of the feedback system is given along with the selection (resolver*65536 32 bit):

- Resolver: The resolution is 16 bit; 65'536 increments are generated per resolver revolution.
- SinCos: The resolution is 10 bit; number of SinCos periods per revolution * 1024 gives the number of increments that are transmitted per revolution.

The following parameters must match the configuration in IMD:	
Link Sampling Rate	LinkSamplingRate
Increments per motor revolution	IncsPerTurn
Nominal speed of rotation	TrunsPerMin
See dt2 files	

3.14 Motor

3.14.1 Motor configuration

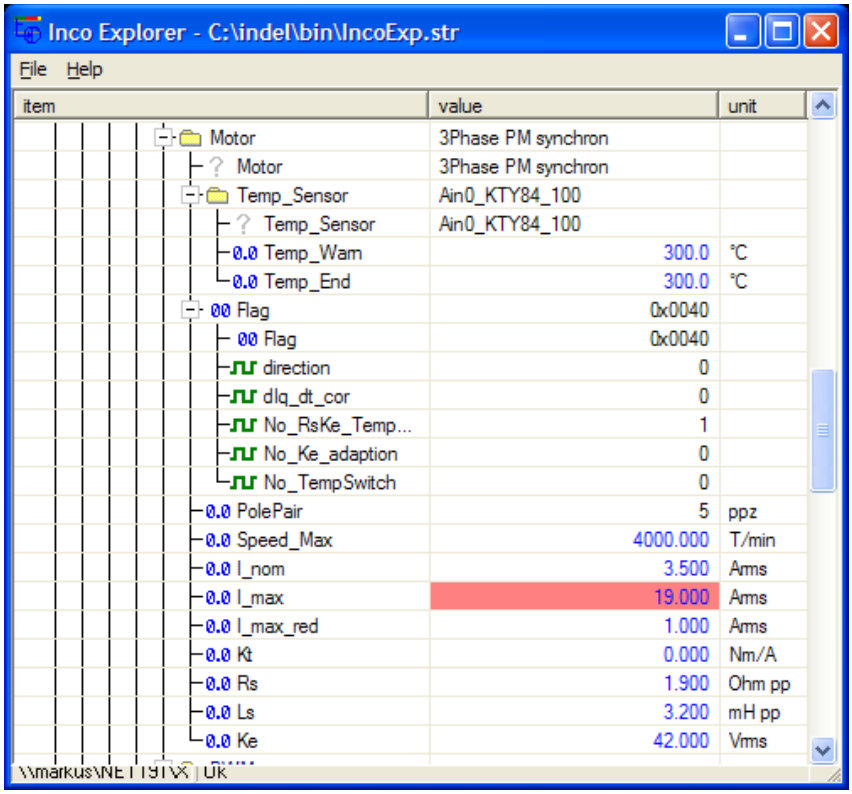


Fig. 39: Motor

Use values from the motor data sheet. The values may need to be verified by means of measurement. Please observe the standardisation of the values: Ls, Rs are given as either phase-phase or as strand resistance or inductance. The Ke is also often given differently.

If it is unclear whether the Rs, Ls, Ke values are correct, they can be measured as per section 10.21 .

Motor-Flags

Flag	direction	Direction of rotation of the motor field
	dIq_dt_cor	Lead value for current calculated from inductance of the motor The current required for the coming PWM pulse is already included in the calculation. However, if the phase reserve brings more than approx. 200Hz, this can cause additional noise.
	No_Ke_Adaption	No automatic correction of Ke during a journey with a high speed of rotation. As standard, bit = 1
	No_RsKe_TempComp	No temperature compensation of R and Ke These three flags are set if a temperature sensor that shows a measurement value in °C is not being used. As standard, bit = 1

Physical motor parameters

Motor	3-phase AC asynchronous motor 3-phase PM synchronous motor, rotational or linear (permanent magnet motor) 2-phase stepper motor: stepper motor with or without feedback DC motor with feedback DC motor without feedback
I_{MAX} [A_{RMS}]	Maximum current of the motor; this current is limited by the servo drive.
I_{NOM} [A_{RMS}]	Nominal current of the motor
I_{RED}: [A_{RMS}]	Current value for the operating mode of `current reduction`. In this mode, the controller limits the maximum current to I_{RED}. The operating mode is set by the fieldbus master.
Ke [V]	Voltage constant of the motor (counter-EMF) [Vrms/1000rpm]
Ls [mH]	Inductance of the motor winding (phase-phase)
PolePair	Number of pole pairs in the motor. (Number of pole pairs = number of poles / 2)
Rs [Ω]	Ohmic resistance of the motor winding (phase-phase)
Speed_Max [U/min]	Maximum permissible mechanical speed of rotation of the motor

Should there be any uncertainty, Rs and Ls should be measured.
Warning: Look out for measuring errors in the measuring tools used!

Temperature sensors

When configuring, you must pay attention to the encoder pin to which the temperature sensor is connected: input Ain0 is assigned to the resolver and input Ain1 to the SinCos encoder.

The characteristic line of the KTY sensors is linearised:

KTY-84-100:

Resistance at 0°	475 Ohm
Resistance ratio 100°/20°	0.190476
Resistance at 100°	1000 Ohm
Resistance at 20°	580 Ohm

KTY-84-110:

Resistance at 0°	778 Ohm
Resistance ratio 100°/20°	0.1667
Resistance at 100°	1667 Ohm
Resistance at 25°	1000 Ohm

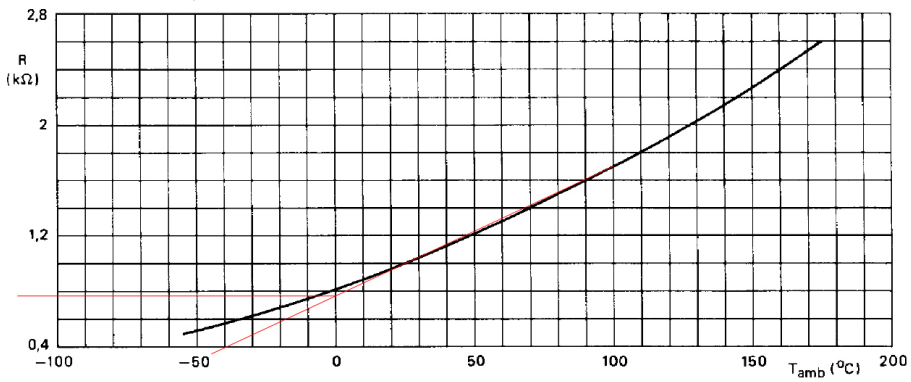


Figure 3.1: Linearisation of KTY-110

Temp_Sensor	Ain0_KTY84_100	KTY-100 to analogue input 0 resolver pin
	Ain0_KTY84_110	KTY-110 to analogue input 0 resolver pin
	Ain1_KTY84_100	KTY-100 to analogue input 1 SinCos pin
	Ain1_KTY84_110	KTY-110 to analogue input 1 SinCos pin
	Ain0_PTC	PTC to analogue input 0 resolver pin
	Ain1_PTC	PTC to analogue input 0 resolver pin
	Ain0_10kGT2	
	Ain1_10kGT2	

KTY-84-130 sensors can also be connected. To do this, configure a KTY-84-100 sensor and wire a 27 kOhm resistor parallel to the sensor.

Temp_Warm — At this temperature (100°C) the servo controller emits the warning Motor Temp warm.

Temp_End — At this temperature (120°C) the servo controller emits the error message Motor Temp max.

3.14.2 Converting Ke for linear motors

$$K_{e\ \ Rotativ} = \frac{K_{e\ \ Translativ} * Magnetabstand * 1000}{60}$$

$K_{e\ translational}$	V/m/s
$K_{e\ rotational}$	v/1000U/min
Magnet spacing	m

3.14.3 Converting Ke for Maxon motors

$$K_{e\ \ Rotativ} = \frac{1000}{K_{e\ \ Maxon}}$$

$K_{e\ rotational}$	v/1000U/min
$K_{e\ Maxon}$	min-1 V-1

3.15 PWM settings

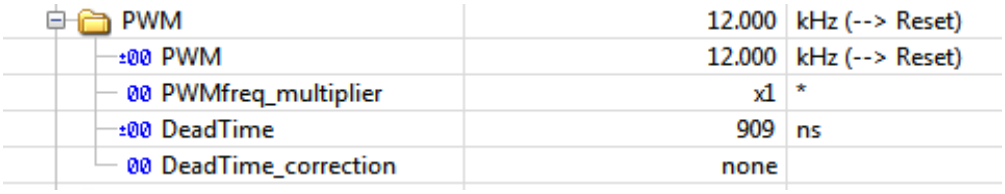


Figure 3.2: PWM settings

PWM

Sampling and PWM frequency of the controller: **8kHz, 12kHz, 16kHz, 32kHz**

Warning: A high frequency also means a higher power dissipation and therefore higher heat build-up at the output stage. Indel does not provide any guarantee in the case of defects due to incorrect configurations.

After changing the PWM frequency you need to reset the drive's hardware so that the changes are accepted!

PWMfreq_multiplier

x1, x2, x3, x4
Motors with minimal inductance that are run on MAX boards with low sampling rates (8kHz); the PWM can be increased in order to improve the current behaviour.

The higher PWM frequency allows a more even current flow to be achieved in the windings.

However, higher switching losses also generate more power dissipation and waste heat.

It is only the PWM frequency that is changed; the position loop remains the same.

x0.5
The setting x0.5 is designed for large SAC3, SAC3x3 drives (INENN: 24A) . With PWM x 0.5 the drive controls with, for example, 16kHz; the PWM, however, only runs at 8kHz. The output values are changed in each edge. This means that the controllers become considerably less hot.

DeadTime

Dead time of the IGBTs. This parameter is controller-dependent.

DeadTime_correction

none
In the case of linear motors with a high current amplification (kPq, kPd), the dead time compensation must be switched off (none).
This allows disruptive noises to be eliminated.

U_half, U_full, U_double
For drives with large motors or large currents, the controller behaviour when at a standstill can be improved if the dead time compensation is switched on.

3.16 Position controller

Indel servo controllers are equipped with a modifiable PID controller, comprising three different parameter sets. Modifiability means pilot control, or what is known as a booster. This means that speed and acceleration-dependent lead values can be added to the target value.

The lead values are purely target values and do not affect the control algorithm, i.e. the stability of the control route is not affected.

The three different PID parameter sets can be used simultaneously and completely independently from one another. As standard, the three parameter sets are provided for forwards, backwards and stand-by.

Driving forwards and backwards

This allows a load change, for example, to be reacted to in a targeted manner in handling tasks.

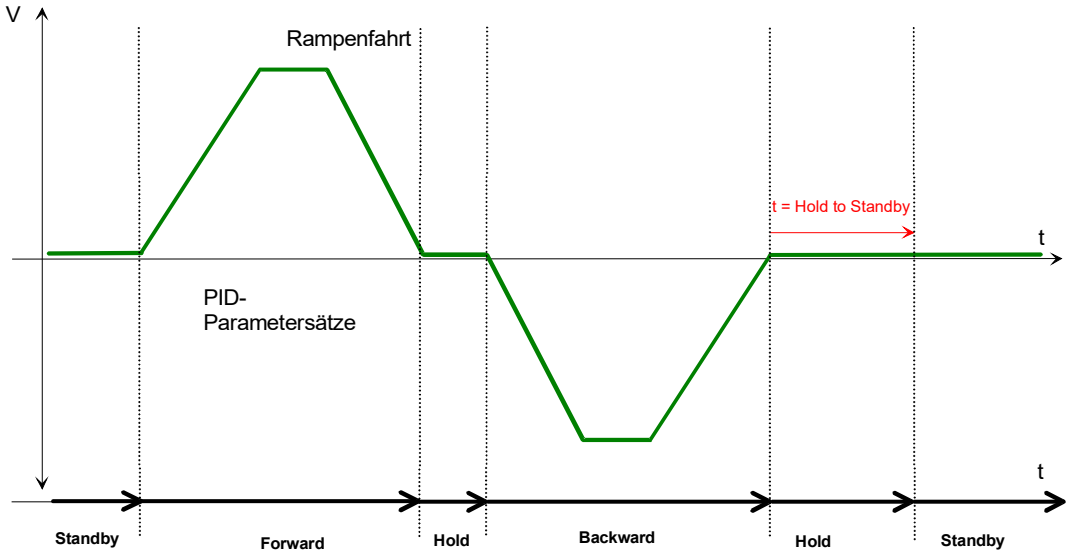


Fig. 40: Trapezoid controller

Stand-by

After a configurable period of time, the control is switched to stand-by. In stand-by mode, the motor can be operated in a power-saving mode, for example.

PID-Parameter

Altered motor parameters are only updated in the RAM of the controller. In order to maintain them long-term, they must be burned to the flash prom using `Burn Values to Target`. If the parameters are not saved or burned, they will be lost forever after switch-off!

PID values also can only be updated when the controller is active.

3.16.1 Configuration of the position controller

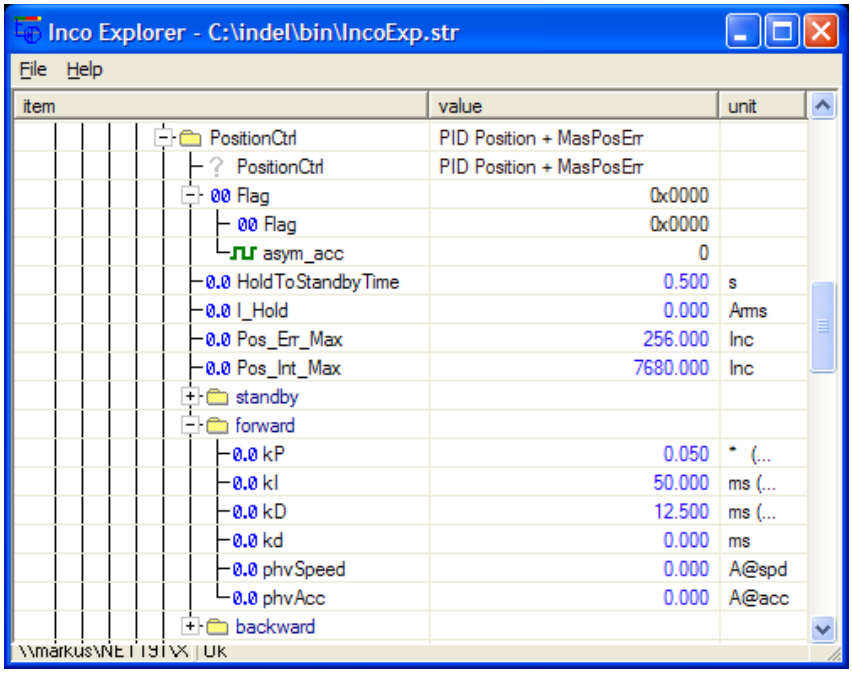


Fig. 41: Position controller

PositionCtrl

Motors with feedback

PID speed	Speed controller
PID position	Position controller with MasPos Error correction
PID position+MasPosErr	Position controller with MasPos Error correction (MasPos Error is always included in the control.)

Motors without feedback

3-phase stepper without FB	Position controller for 3-phase PM motors without feedback
2-phase stepper without FB	Position controller for 2-phase PM motors without feedback
DC motor without FB	Position controller for DC motors without feedback

HoldToStand-by [s] At the end of a ramp journey, the controller switches to “Hold Mode". After the configurable time “HoldToStandby”, the controller changes from hold mode to Standby-Mode. There is a separate PID parameter set for stand-by mode.
This function allows the drive to be operated in power-saving mode during a downtime.

I_Hold Lead value: For “suspended loads”, a constant value or constant force can be set using "I_Hold”.

Pos_Err_Max Limitation of the maximum following error. Typically: 256 MotInc

Pos_Int_Max Limitation of the I component of the PID control (position controller) Typically: 7680 MotInc

Flag	Acc_Filter	Acceleration filter
	asym_acc	Asymmetrical acceleration lead value 1: Acceleration lead value in forward applies to acceleration 1: Acceleration lead value in backward applies to braking 0: Acceleration lead value in forward applies to pos. direction 0: Acceleration lead value in backward applies to neg. direction

There are 3 PID parameter sets available for forward, backward and stand-by:

Value		Standardisation	Description
kP	*	A/Inc control deviation	Proportional value Position error
kI	ms		Integral component Integrator position error
kD	ms		Differential component Speed error
kd	ms		Acceleration error
phvSpeed	*	A/v	Lead value for speed
phvAcc	ms		Lead value for acceleration

A: *Ampere*
Inc: Increment
v: Speed

The parameter kd can only be used with high-resolution encoder systems.

Switching of the PID parameters

In the case of a following error of more than 10 MotInc, the parameter set is switched from stand-by to forward or backward.

$$10\,MotInc = \frac{1\,Turn}{N_{inc}} = \frac{360\,°}{4096} = 0.879\,°$$ Also see section 2.2.

Warning

If the PID values used for stand-by are different to those used for forward or backward, it may be that the axis “jerks” every so often, in order to compensate for cumulative path errors. This will particularly happen if the kP is considerably smaller in stand-by than for forward/backward.

3.17 Power Supply

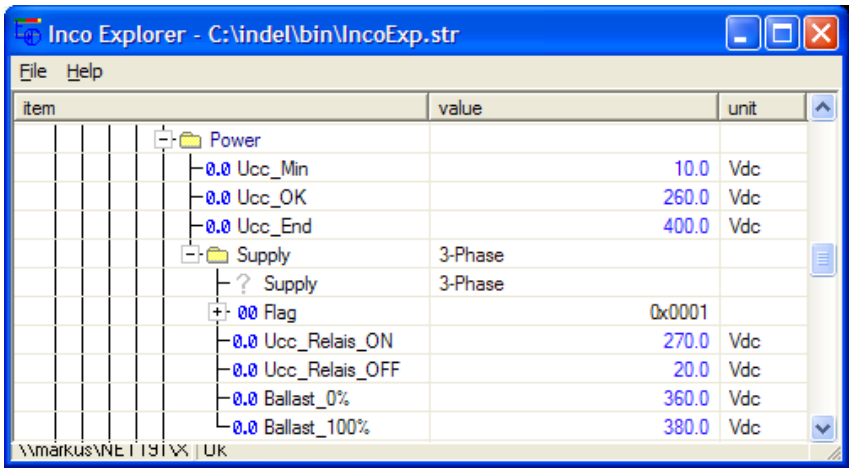


Fig. 42: Power Supply

Supply	1-phase or 3-phase supply, in MAX and AX boards Brake only	
Ucc_End	If the intermediate circuit voltage exceeds this value, the control goes into error.	
Ucc_Min	Below this limit value, the control goes into error. Between Ucc_Ok and Ucc_Min the control emits a warning: "Ucc low"	
Ucc_OK	Normal operation is between Ucc_End and Ucc_OK.	
Ballast_0%	With this intermediate circuit voltage, the PWM of the ballast IGBT is 0%	
Ballast_100%	With this intermediate circuit voltage, the PWM of the ballast IGBT is 100%	
Ucc_Relais_ON	When the controller is switched on, the intermediate circuit capacitors are loaded via a resistor. As soon as the intermediate circuit voltage exceeds the threshold, the loading resistors are bypassed.	
Ucc_Relais_OFF	Threshold for switching off the relays in the case of interrupted supply voltage	
Flags	No_PhaseFailure	Phase error evaluation is deactivated This flag must be one in the case of 1-phase supply and in the case of DC supply via the intermediate circuit.

Recommended values for the configuration of the voltage supply

In order to protect the intermediate circuit capacitors from premature aging, the value must not drop below Ucc Min.

Supply		3-phase 400V	1-phase 230V	1-phase 120V	48V DC	24V DC
UCC_End	V	800	400	220	56	30
UCC_Min	V	450	260	125	40	20
UCC_Ok	V	500	280	135	44	22
Ballast_0%	V	760	360	180	50	25
Ballast_100%	V	780	380	200	52	28
UCC_Relais_ON	V	480	270	110	(42)	(22)
UCC_Relais_OFF	V	470	260	100	(40)	(20)

The values for Ucc_Relais_ON/OFF in MAX and AX boards are irrelevant as there is no load switching on these boards.

3.18 Speed Filter

Inco path for the speed filters: Ctrl.MotorConfig.SpeedFilter

Also see section: 11.4 Procedure for optimising the control route.

3.18.1 Average Speed-Filter

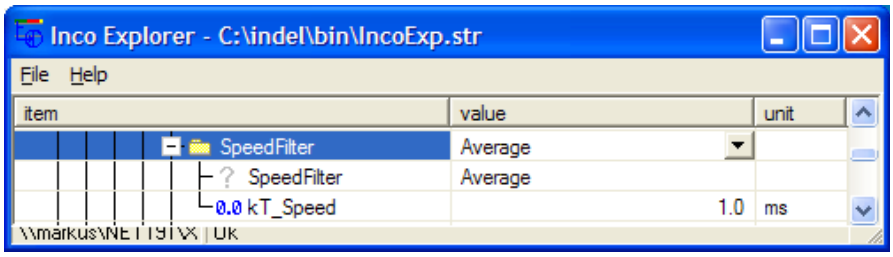


Fig. 43: Average Filter

3.18.2 Speed Observer

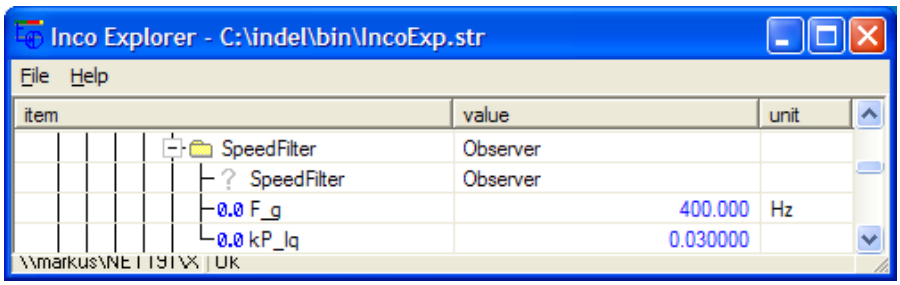


Fig. 44: Observer

3.19 Actual hardware values

Some actual values, e.g. the A-B tracks of the incremental encoder, can be viewed directly in the hardware.
Inco path for actual hardware values: Hardware

In the case of older firmware versions:
Inco path for actual hardware values: Ctr0.Hardware.Status

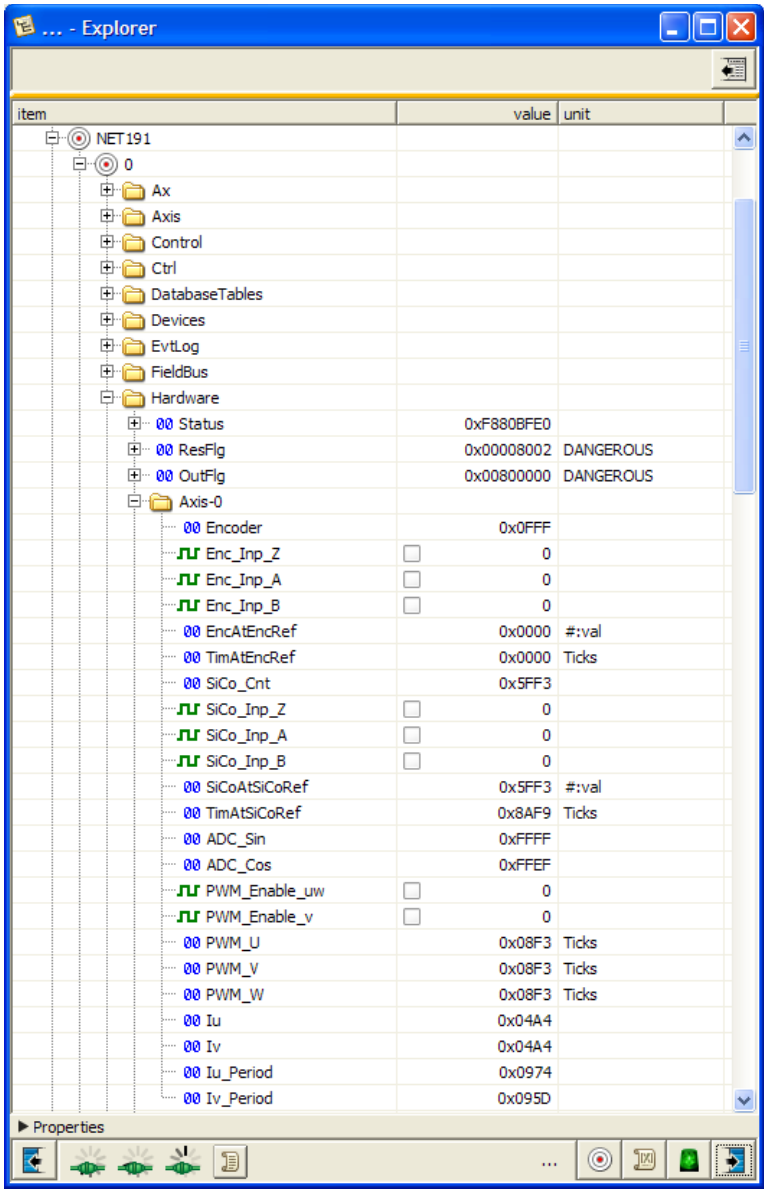


Fig. 45: Actual hardware values

3.20 Porting Info-link motor configuration files on GinLink

In order to port older motor configuration files to the latest standard, the missing parameters can be stored in a separate file. This partial configuration file can then be loaded into the drive.

It is recommended that you do not save any axis-specific parameters, such as GinLink configuration, etc., in the partial configuration files.

Example for 3x400V drive

Ctrl.MotorConfig.Version	3.199997
Ctrl.MotorConfig.PWM.DeadTime	909.088135
Ctrl.MotorConfig.PWM.DeadTime_correction	0.000000
Ctrl.MotorConfig.PWM.PWMfreq_multiplier	1.000000
Ctrl.MotorConfig.Power.Ucc_End	800.000000
Ctrl.MotorConfig.Power.Ucc_Min	0.000000
Ctrl.MotorConfig.Power.Ucc_OK	490.000000
Ctrl.MotorConfig.Power.Supply.Ballast_0%	700.000000
Ctrl.MotorConfig.Power.Supply.Ballast_100%	760.000000
Ctrl.MotorConfig.Power.Supply.Supply	2.000000
Ctrl.MotorConfig.Power.Supply.Ucc_Relais_OFF	470.000000
Ctrl.MotorConfig.Power.Supply.Ucc_Relais_ON	480.000000
Ctrl.MotorConfig.Power.Supply.Flag.Flag	0.000000
Ctrl.MotorConfig.Power.Supply.Flag.No_PhaseFailure	0.000000

4 Safety configuration

4.1 Operation with Safe Torque Off (STO)

Prerequisites

- The two safety relays must be controlled by the control using digital 24V outputs. (Test controller release)
- The main contactor must be controlled by the control using a 24V output.

Configuration of the external enabler in the servo drive

See section: 3.10 Extern Enable

Protecting the drive against overload

Important: for protection of the drive also see section:
3.9.3 I2t control**,**
10.1 Protecting the motor against overload

Configuration of the ballast resistance in the servo drive

Configure the ballast resistance in accordance with section 3.17 Power Supply

Configuration in the IMD configuration

The following parameters must be correctly set in the dt2 configuration:

X.Axis.dt2

Emergency Type	When the external enabler in the drive is switched off, the emergency stop braking ramp is activated: 0: simple stop without emergency stop braking ramp 1: stop with emergency stop braking ramp; controller is deactivated 2: controller is deactivated
Emergency Delay	Delay between Stop and Inactivate with emergency stop braking ramp in ms

X.PosCtrl.dt2

emgB	Retardation of the emergency stop braking ramp; this delay must be adapted to suit the mechanical conditions.

Monitoring the auxiliary contacts

The auxiliary contacts of the two safety relays (N/O switches) are displayed in INCO-Tree.
The part of the application that switches the safety relays on must check the condition of the two auxiliary contacts. The comparison of their condition must not last longer than 50ms.

In the event of a fault, an emergency stop must be triggered.

Testing controller release

The controller release must be tested cyclically. To do this, both safety relays and the external enabler are switched off and an attempt is made to activate the control and move the axis.

This procedure involves the following steps:

External enabler	Relay 1	Relay 2
off	off	off
off	on	on
on	on	off
on	off	on
on	on	on

If the testing of the controller release is unsuccessful, the drive must not be put into operation!

4.2 Engaging Safe Torque Off

Wiring examples

You can find several wiring examples in this manual:
Indel-Safety-Manual.pdf (see section on connection examples).

Configuration, wiring

Example for an axis that can brake to 0 from the maximum speed in 200ms:

- Wire safety door to SAC external enabler
- Wire safety door (two-channel) to SAC-STO via safety time relay (e.g. 300ms)
- MotorConfig.Enable.kT_ExtDis programme to, for example, 250ms
- in the IMD project, configure the SAC external enabler as an emergency input
- in the IMD project, select the emergency braking ramp in such a way that the unit can be stopped from the maximum speed in, for example, 200ms

Procedure in the SAC and fieldbus master

- Safety door opens, external enabler on the SAC drive is lost
- In the SAC, the disable timer starts, the axis remains active in position operation and continues its control
- In the master (SAM or PCI card), the emergency braking ramp is introduced automatically
- depending on the speed, Speed=0 is reached within 200ms at the latest
  the master deactivates the axis (red active LED goes out; if the active status is configured to a digital output, this output is lost)
- the SAC axis switches off after 250ms, regardless of whether Speed=0 has already been achieved
- after 300ms, the safety time relay switches off the hardware output stages via the STO Safety Torque Off.
  From this point onwards, the safety pulse inhibitor is active and the drive is no longer able to control the motor.

Checking the braking ramp

The entire procedure and the selected emergency braking lamp must be tested and checked using log files:

- all signals must appear as intended:
  Ctrl.Actual.Enable.Ext_En
  Ctrl.Actual.Enable.Safety_0
  Ctrl.Actual.Enable.Safety_1
  Ctrl.Actual.Enable.GinLink_En
  Ctrl.Actual.GinLink.FB_Status.Axis_Active
  Ctrl.Actual.PositionCtrl.cmd_V
  Ctrl.Actual.PositionCtrl.act_V
  Ctrl.Actual.PositionCtrl.cmd_A
  Ctrl.Actual.PositionCtrl.act_A
- the drive must not switch off prematurely and spin out due to overcurrent
- engage the STO at the highest speed to ensure that the speed can be reduced to zero within the set time

Notes

The external enabler does not need to be connected to a digital input again; this is clear in the process diagram under the corresponding position channel.

If needed, the external enabler can also be connected via digital outputs on the control. This is not compulsory.

With all Indel drives and motion boards, there is always only 1 external enabler and 1 STO input available, and it always has an impact on all axes at the same time.

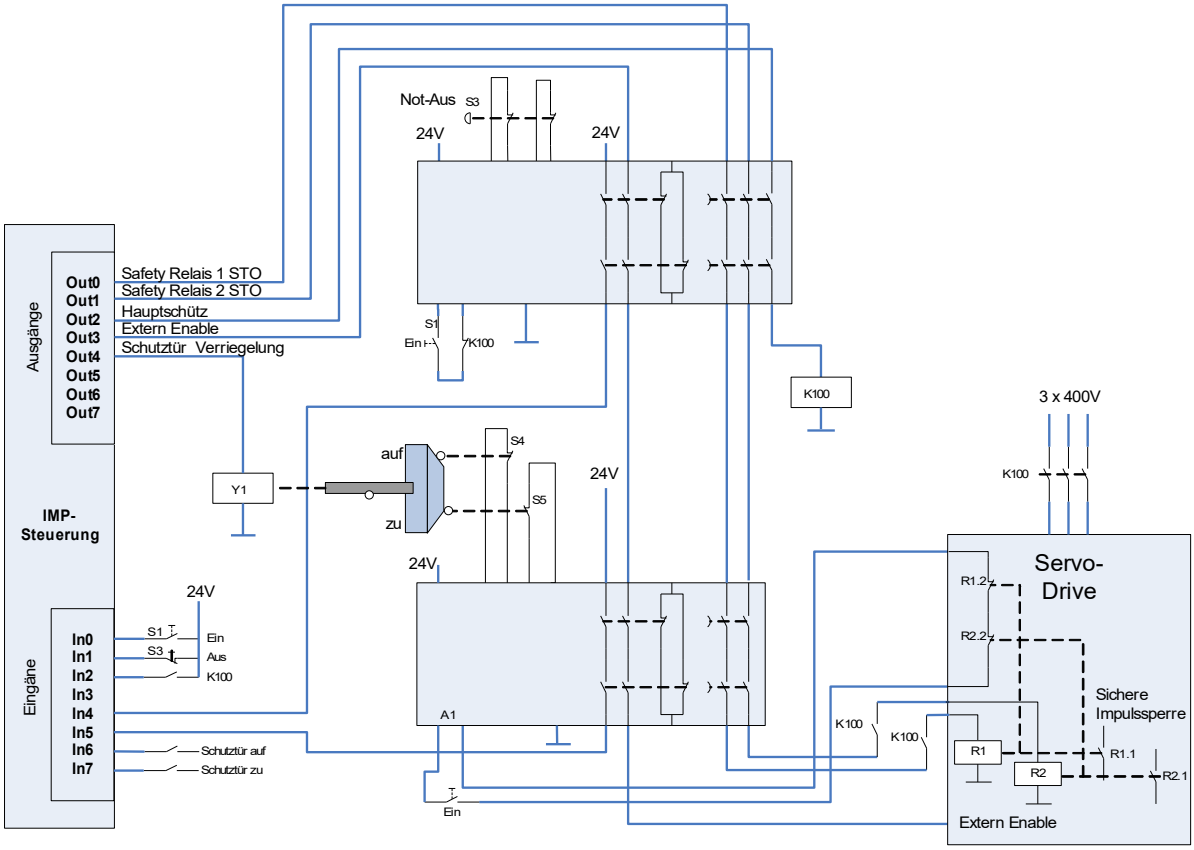


Fig. 46: Example: STO, locked safety door

The examples are non-binding. The user is responsible for the binding layout of the safety functions (SF). The user must observe the state of the art in the corresponding European standards, such as EN ISO 13849-1/-2, EN 62061, EN 1088, etc.

5 IMD configuration

5.1 GinLink configuration in IMD

Description of GinLink configuration in IMD with the GinLink.dt2 file.
The configured cycle time on the GinLink of each axis must match the configured cycle time in the motor configuration file under `Ctrl.MotorConfig.GinLink.LinkSamplingRate`.

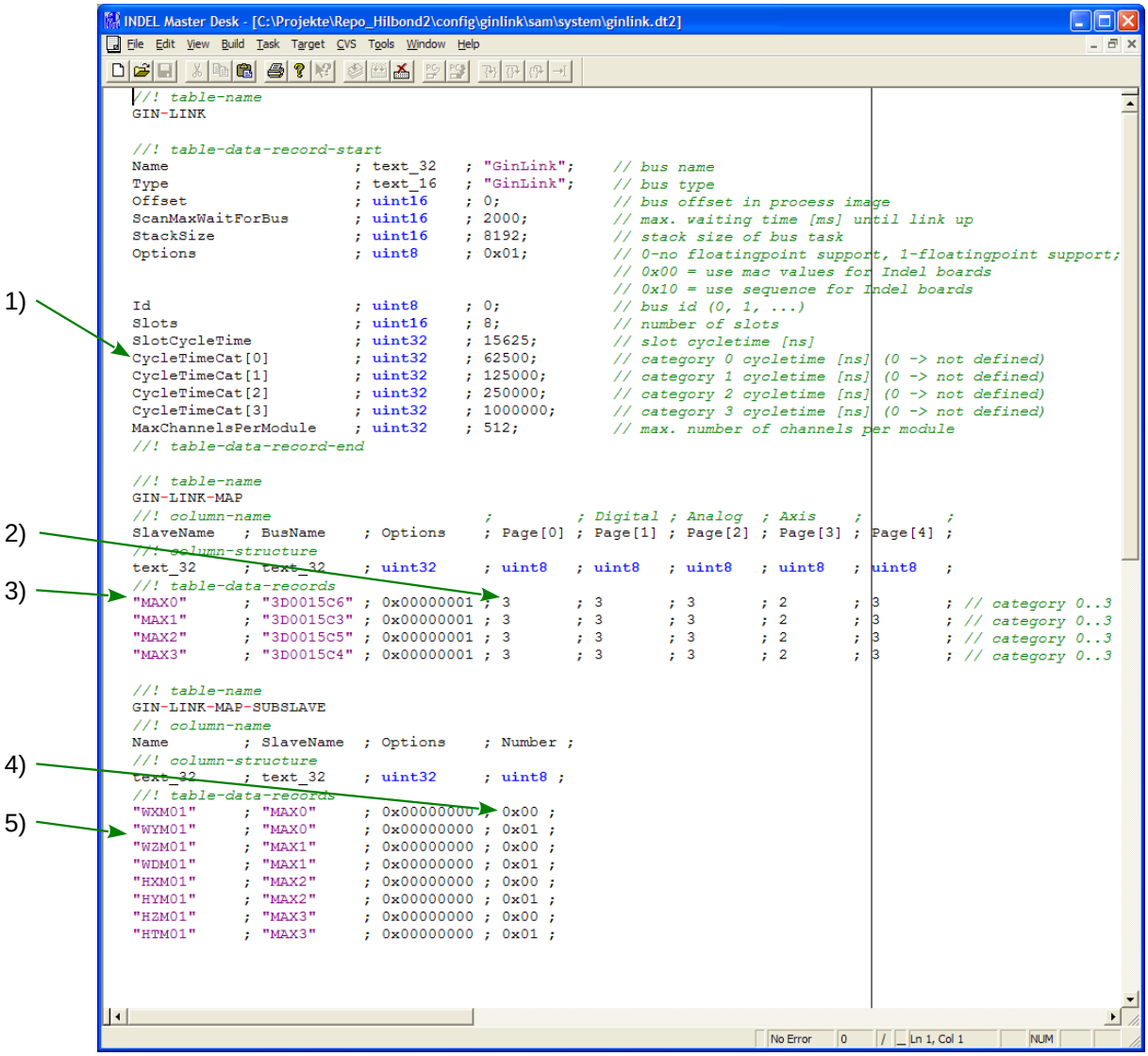


Fig. 47: GinLink configuration

Index of all GinLink cycle times 1)

The following cycle times are possible via the GinLink as standard:

Cycle time	Cycle rate	Index
• 1ms	1kHz	3
• 0.25ms	4kHz	2
• 0.125	8kHz	1
• 0.0625	16kHz	0

Cycle times of the different GinLink pages 2)
The entire periphery, incl. communication, is distributed across different data pages. Each of these pages can be transmitted via the GinLink with a configurable cycle time. The following pages are available:

- Page[0]
- Page[1] Digital inputs and outputs
- Page[2] Analogue inputs and outputs
- Page[3] Axes
- Page[4]

Assignment of servo drive to MAC number 3)
Each GinLink participant has a universal MAC number. In the configuration, the assignment must be given between the name of the GinLink participant and its MAC number.

When exchanging a GinLink participant in the field, the MAC number must be adapted!

Index of axes for drives with several output stages 4) 5)
Drives with several output stages, such as AX4, MAX2/4 or SAC3x3, require this index for the assignment of the names of the axes.

5.2 Axis configuration in IMD

5.2.1 Motor.dt2

This file contains information on the motor or axis and is required by SAM for the conversion of motor increments into the position.

TurnsPerMin	Standardisation factor for the speed on the GinLink. Must match `Ctrl.Motorconfig.GinLink.Vcmd_100%`
IncsPerTurn	Number of increments on the feedback configured in FB_Ch_0 channel. Must match the axis-specific number of increments per motor revolution from the motor configuration file.
FeedPerTurn	Number of configured “units” per motor revolution
GearRatio	Gear used. If there is no gear available, then transmission 1.0

5.2.2 PosCtrl.dt2

The Variable DeadTime must be correctly set in the PosCtrl.dt2 file. This dead time [ms] is required by Sam and must be set to the current sampling frequency of the corresponding axis.

Example:
Control frequency at 16kHz:

$$DeatTime = 2\frac{1}{Regelfrequenz} = 2\frac{1}{16\,kHz} = 0.125\,ms$$

6 Controller configuration

In the folder `Ctrl.CtrlConfig`, you can change all parameters rated to the controller.

- Adjustment data for current measurement
- Measuring ranges for current, voltage, temperature
- Maximum IGBT current
- Resolver adjustment

**All adjustments are carried out in-factory by INDEL AG. Values must never be changed without consulting Indel. If the controller parameters are overwritten (burn controller parameter file *.chf to flash PROM),
adjustment data will be irrevocably lost and the controller must be sent in for repair.**

In order to be able to load the controller configuration into the drive, you need an entry in the Windows Registry. This entry can be requested from Indel.

This is necessary in order to adjust the sampling frequency of the position controller in the drive.

7 Error message from the servo drive

7.1 Error messages

Stop	0x0000'0001
Ucc below Ucc min	0x0000'0002
Ucc larger than Ucc max	0x0000'0004
I2t exceeded > 120%	0x0000'0008
Output stage overheated (80°C)	0x0000'0010
Motor temp exceeded	0x0000'0020
Motor short-circuit	0x0000'0040
Resolver SinCos error	0x0000'0080
Maximum rotational speed exceeded	0x0000'0100
Safety relay not switched on	0x0000'0200
Auto-commutation error	0x0000'0400
Power end stop reached	0x0000'0800
Phase error	0x0000'1000
PWM Watchdog: Interrupt overrun	0x0000'2000
missing Exteral Enable	0x0000'4000
missing (motor) configuration	0x0000'8000
Fieldbus watchdog	0x0001'0000

7.2 Warnings

Ucc below Ucc ok	0x0000'0001	
Ucc is set up and OK	0x0000'0002	
Warning Iq reached	0x0000'0004	1)
Warning output stage hot (75°C)	0x0000'0010	
Warning I2t exceeded	0x0000'0020	
Motor temp exceeded	0x0000'0040	
100% modulation exceeded	0x0000'0080	
Warning unloading time exceeded	0x0000'0100	

1) This warning appears when the safety relay is not switched on and the pulse inhibitor is activated.

Further details on the error messages can be found in the handbook Hardware-Manual-SAC3.pdf.

8 Indel position controller

8.1 Move commands

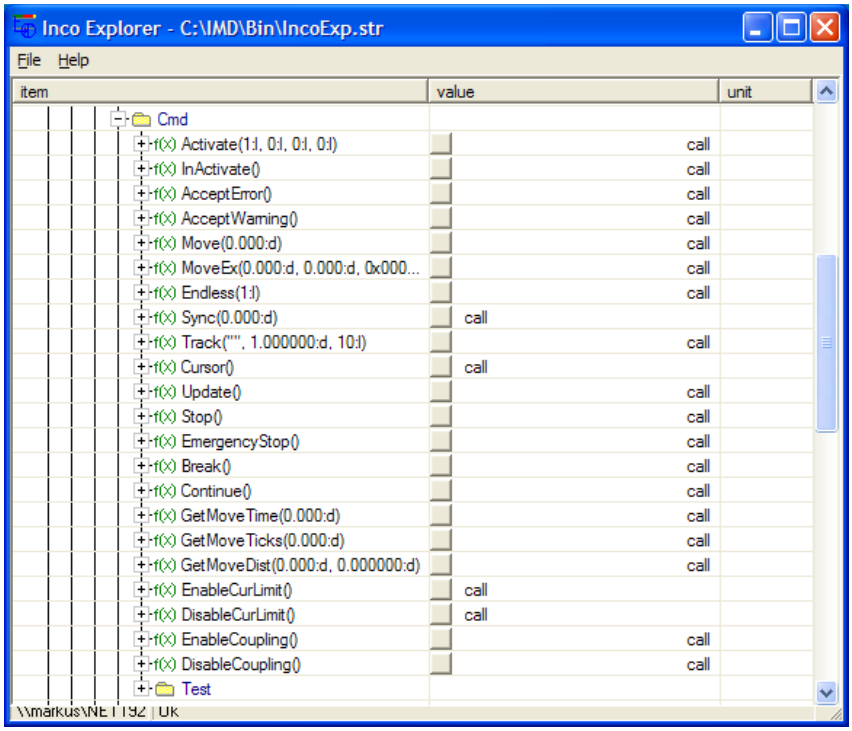


Fig. 48: Move commands

Activate() Activation of the axis

CheckPos:	The maximum permissible following error is considered; if “maxSerr” is exceeded, the control goes into “following error”
SimulatePos:	The actual value that comes from the servo controller is simulated; there is no “errS”.
SimulateOut:	The target value is simulated; the “errS” is calculated but not evaluated.
SimulateAct:	No target value is sent to the servo controller. The ramp is only calculated on the software side.
Activate(0,0,1,0)	corresponds to “Simulation Mode" in the old system

InActivate() Deactivation of the axis

Move() Move to positions in °, m, mm

Endless() Rotate endlessly in a positive or negative direction. The curve profile is configured under “Ramp/Cmd”

Break() Interrupts the current move command

Continue() Continues the last move command

Synch() Synch journey

Position: The synch mark (zero pulse) is expected within this position.
If the zero pulse is not found, the control goes into error:

SynchPosition: Synchronisation position, i.e. after the synchronisation,
the actual position of the synch point is set to “SynchPosition”.

StopAtSynch: When crossing the synch position, the unit is either stopped
or travels to the end of the ramp.

EmergencyStop()

NotStop

Stop: The axis stops with the set maximum braking ramp:
Axis.PhysicalAxis0.Ramp.Cmd.emgB = 360'000 °/s2

Inactive: The axis is switched to inactive

Stop/Inactive: The axis stops with the set maximum braking ramp:
Axis.PhysicalAxis0.Ramp.Cmd.emgB = 360'000 °/s2
The axis is switched to inactive after the delay time

Delay: Time in ms

AcceptError() Acknowledgement of errors
This function is intended for implementation of error handling.
You can acknowledge individual or all errors using this function.
An axis can only be activated if all errors have previously been acknowledged.

AcceptWarning()

Acknowledgement of warnings

8.2 Error messages from the position controller in the fieldbus master

Following error	Axis.PhysicalAxis0.Error	= 0x0000'4000	"Control Error Pending"
	Axis.PhysicalAxis0.Control.Error	= 0x0000'0001	
	Axis.PhysicalAxis0.Ramp.Error	= 0x0000'0000	
Synch error	Axis.PhysicalAxis0.Error	= 0x0000'0002	"Synch Error"
	Axis.PhysicalAxis0.Control.Error	= 0x0000'0000	
	Axis.PhysicalAxis0.Ramp.Error	= 0x0000'0000	
Controller error: Ucc < Uccmin			
	Axis.PhysicalAxis0.Error	= 0x0000'4000	"Control Error Pending"
	Axis.PhysicalAxis0.Control.Error	= 0x0000'4000	"External Controller Error"
	Axis.PhysicalAxis0.Ramp.Error	= 0x0000'0000	
	\\..\PhysicalAxis0\Ctrl.Actual.Errors	= 0x01010001	"Warn: Ucc < Ucc_OK" "Warning: Power-OFF DisCharge failure”
	\\..\PhysicalAxis0\Ctrl.Actual.Errors	= 0x0000'0003	"Ucc < Ucc_Min"

8.2.1 Moving axes

"Delete error, start axis" routine

- InActive() Deactivate position controller and servo controller
- AcceptError() Acknowledge error
- Active() Activate position controller and servo controller

- Sleep(3000) At the first switch-on with a SinCos encoder, auto-commutation is carried out during this time. After successful auto-commutation, the controller becomes active (also see configuration). The sleep time is dependent on the configuration of the auto-commutation!

- Error handling

"Synch journey" routine

- "Delete error, start axis" routine
- Is the axis at zero? If yes, move away
- Sync(Pos, SyncPos, StopAtSync)
- Error handling

Routine "Move"

- Move (pos, synchronous/asynchronous)
- Error handling

9 Trapezoid controller

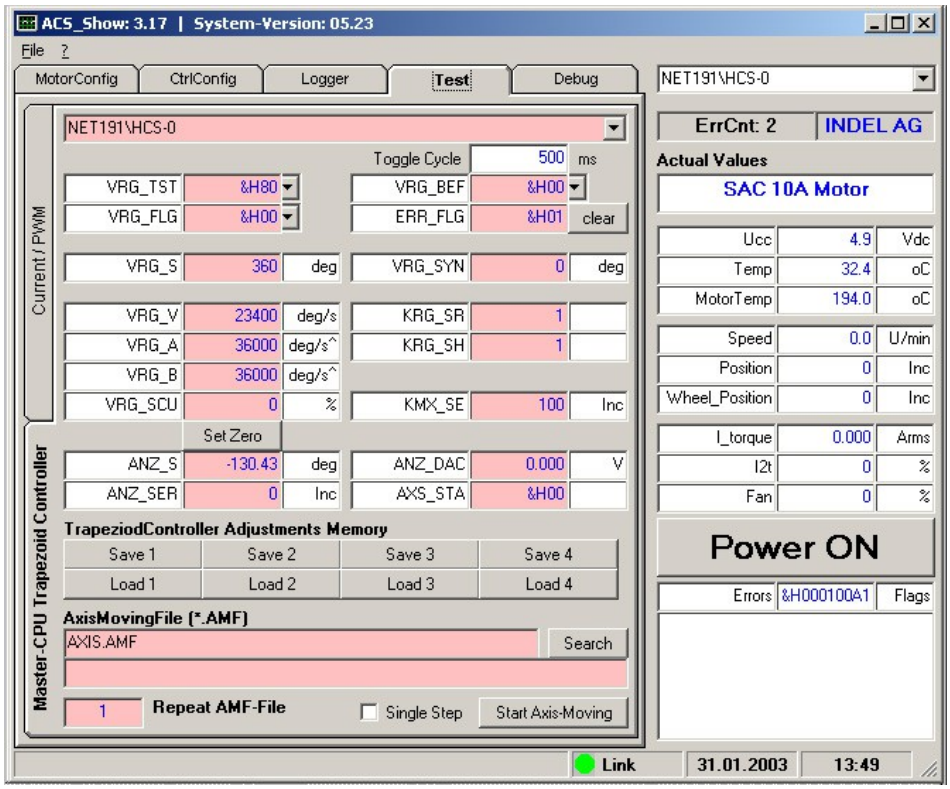


Fig. 49: ACS-Show

9.1.1 ACS-Show

With ACS-Show, all Indel axis cards: servo controller, stepper motor indexer and output stage, DC motor output stage, etc. can be put into operation.

9.1.2 Specifications for trapezoid and S profile

These values can still be changed during operation.

vrg_s	route to be travelled.	[degrees, m, mm]
vrg_v	speed to be travelled at	[degrees, m, mm/sec]
vrg_a	Beschleunigung	[Grad, m, mm/sec^2]
vrg_b	delay	[degrees, m, mm/sec^2]
vrg_scu	S curve proportion in %	[%]
vrg_syn	synchronisation position, e.g. after synchronisation, the actual position of the synch point is set to vrg_syn.	

9.1.3 Control constants

The control weights can be given separately for S (route) and V (speed) as well as for RUN (axis running) and HALT (axis at standstill). The value 1.0 means that 1 INC error causes 1 DAC bit correction. These values can only be definitively set during commissioning.

krg_vr — Control weight for the speed when the axis is running.
If Indel servo controllers are being used, this value must be set to zero!

krg_vh — Control weight for the speed when the axis is at a standstill.
If Indel servo controllers are being used, this value must be set to zero!

krg_sr — Control weight for the route when the axis is running.
Typically krg_sr = 1 for the servo controller.

krg_sh — Control weight for the route when the axis is at a standstill.
Typically krg_sh = 1 or 2 for the servo controller.

kmx_se — This is the maximum permissible path error in INC (difference between actual and target position). If this value is exceeded during travel, the controller will automatically go into emergency stop.

9.1.4 Actual values

The current position is displayed in the unit of measurement (degrees, metres or millimetres) set by you during configuration.

S_ERR — Current position deviation in INC (difference between target and actual position). If, during travel, this figure is greater than

kmx_se — the control will go into emergency stop and bit 0 will be set in the error flag `"err_flag"`.

V_ERR — (only in simulation mode) Current speed deviation in INC/grade (difference between target and actual speeds).

DAC — Current DAC value in volts.

9.1.5 Axis status

Axis status
axs_sta — Bit 0: Synch input

9.1.6 Move commands, VRG_BEF

This action switch can be used to trigger various move commands.

Start	0x81h	Taking into account **vrg_v**, **vrg_a** and **vrg_b**, move to position **vrg_s** (this function can be directly selected using **F7**).
New V/A	0x82h	Accelerate / retard to new speed vrg_v
Synch	0x83h	Taking into account **vrg_v**, **vrg_a** and **vrg_b**, move to position **vrg_s**. Check the occurrence of a synch pulse during travel. If such occurs, set the actual position to **vrg_syn** and brake as normal; otherwise, set the error flag to synch error.
Start Pull	0x84h	Follow the default position **vrg_s**. As the control always immediately attempts to reach **vrg.s** (without travelling a ramp), you must ensure that you

do not select steps that are too large. (Travelling error if the path error is too big!). The max. acceleration of the motor should also be taken into account. This mode can be used to travel curves, for example.

Stop	0x88h	Slow the moving axis down to zero, taking into account **vrg_b** (this function can be directly selected using **F8**).
Not Stop	0x89h	Slow the moving axis down to zero with immediate effect (without braking ramp).
POS=0	0x8Ah	Standardise angle to 0 ... 359°, as per the section on standardisation
POS=0	0x8Bh	set actual position to 0.
POS=VRG_SYN	0x8Ch	set actual position to VRG_SYN
Toggle		Taking into account **vrg_v**, **vrg_a** and **vrg_b**, move to position **vrg_s**, wait 'Delay' ms, set vrg_s = -vrg_s and start again from the beginning. vorn. Repeat this until it is cancelled.
PToggle		Taking into account **vrg_v**, **vrg_a** and **vrg_b**, move to position **vrg_s**, wait 'Delay' ms and travel the same route again. Repeat this until it is cancelled (this function can be directly selected using **F9**).
Delay		With this point, you can set the waiting time in **ms** between two move commands for ‘Toggle’ and ‘PToggle’.

9.1.7 Standardisation, VRG_FLG

The master always standardises the actual position automatically as soon as HALT is reached. The standardisation does not cause any increments to be lost (no cumulative errors, even if the unit is standardised to 0.0 after each journey).

None	0x00h	The position remains as it is after each journey.
Angle	0x01h	The angle is standardised at 0..359°.
Zero	0x02h	The actual position is set to 0.0 after each journey.
Endlos	0x03h	From them start command, the unit travels to the next stop at VRG_V, regardless of how big vrg_s is. (Note: vrg_s must be sufficient for the acceleration ramp, otherwise endless travel will not become active.)
Round	0x04h	Round angle to 365°

9.1.8 Operating mode, VRG_TST

With the operating mode, you decide how the control will behave towards the outside world.

Inactive	0x00h	The control is switched off for this axis.
Active	0x80h	The full scope of the control (including error monitoring, etc.) is activated.
A. o. F.	0x81h	The control is active, but does not revert to emergency stop in the event of a travelling error.
Simulation	0x82h	The motor is simulated. This is a purely target value output.

DAC/INC	0x83h	The control for this axis is switched off. The DAC or the incremental encoder input can now be used for their own purposes.

9.1.9 Standard factors

knf_dac — This factor serves the purpose of internal conversion of target speed into DAC bits. It is calculated by means of the configuration.

knf_s — This factor serves the purpose of converting your specifications V, S, A, B, into Inc. It is also calculated by means of the configuration.

knf_e

knf_tim — Sampling rate:
0 = 4 ms
1 = 2 ms
2 = 1 ms
3 = 0.5 ms

9.1.10 Error messages

Axis errors. As long as this byte is not deleted, no new commands will be accepted.

err_flg — 00h ok

Bit 0=1 → Following error. **anz_ser** has become greater than **kmx_se** -> OFF
Bit 1=1 → Synchronisation could not be carried out
Bit 2=1 → The axis card is not registering correctly
Bit 3=1 → External controller error

10 Step-by-step commissioning

The following section shows how to commission a motor or axis, step by step.
Before beginning commissioning, the entire commissioning manual and hardware manual must be read.

For the first commissioning, the motor should be run without a load. The operator must have visual contact with the motor shaft.

Work through each step precisely. If individual steps are missed out, the motor may not function correctly or may behave in a manner that is difficult to explain.

10.1 Protecting the motor against overload

In order to protect the motor against overload during commissioning, the following points must be considered:

- From the motor data sheet: Maximum time for which the maximum current may flow
- Connect and commission the temperature sensor for the motor winding
  If no temperature sensor is available, the I2t control is the only protection against motor overload. See section: 3.9.3 I2t control
- Adjust I2t control
- Do not exceed the maximum speed and acceleration ramp

10.2 Enter motor parameters

Enter the values from the motor data sheet. The details relate to PM synchronous motors in star formation. More information can be found in section 3.14 .

Switch off motor supply: L1, L2, L3 off!

1. Inco path for motor configuration Ctrl.MotorConfig.Motor

2. Rs — Is entered as R phase-phase: Rs = 2 * Rstr

3. Ls — Is entered as L phase-phase: Ls = 2 * Lstr
   (Siemens specification Ld = rotational field inductance -> L phase-phase)

4. Ke — Is entered as Ke phase-phase -> then they will all be correct in V_{RMS}/1000U/min

   Should there be any **uncertainty** regarding the correctness of Rs, Ls, Ke or if their units are unclear, the values should be verified in accordance with section 10.21 Fine-tuning of Ke, Rs and Ls .

5. Inom — Is given in Arms
   Imax — Is given in Arms
   Ired — Is given in ArmsIred = Inom (default)

6. Speed_Max — Maximum mechanical speed of rotation

7. BurnMotorCfg, File -> Save

10.3 Temperature switch

1. Connect motor and encoder (resolver, incremental encoder or SinCos) to controller.
2. Switch on 24V for controller supply; do **not** switch on motor supply (3 x 400V)!

10.3.1 Temperature sensor in resolver/SinCos cable

If there is a temperature sensor (resistor) in the resolver/SinCos cable, the motor temperature under `Ctrl.Actual.Motor` must display a logical value between 20 ... 30°.

Configuration of the temperature sensor
Inco path for motor configuration: Ctrl.MotorConfig.Motor

3. Temp_Warn — Above this motor temperature, a warning is displayed Temp_Warn = 100°C
4. Temp_End — Above this motor temperature, an error is displayed and the controller is switched to inactive. Temp_End = 120°C
5. Flag

No_Ke_adaption	= 0	Temperature-dependent Ke compensation ON
No_RsKe_TempComp	= 0	Temperature-dependent Rs compensation ON
No_TempSwitch	= 1	No bi-metal switch in the motor cable

10.3.2 Temperature limit switch in resolver/SinCos cable

If there is a limit switch for motor overheating in the resolver-SinCos cable, the Actual Value for the value `Ctrl.Actual.Motor` will display the temperature -90° if the switch is closed and +190°C if the switch is open.

6. Temp_Warn — Temperature warning = 100°C
7. Temp_End — Temperatur Error = 100°C
8. Flag

No_Ke_adaption	= 1	Temperature-dependent Ke compensation OFF

9.

No_RsKe_TempComp	= 1	Temperature-dependent Rs compensation OFF
No_TempSwitch	= 1	No bi-metal switch in the motor cable

10.3.3 Limit switch in the motor cables

If there is a limit switch for motor overheating in the motor cable, the Actual Value for the value `Ctrl.Actual.Motor` will display the temperature +190°.

9. Temp_Warn — Temperature warning = 300°C
10. Temp_End — Temperatur Error = 300°C
11. Flag

No_Ke_adaption	= 1	Temperature-dependent Ke compensation OFF
No_RsKe_TempComp	= 1	Temperature-dependent Rs compensation OFF
No_TempSwitch	= 0	Bi-metal switch in the motor cable

10.3.4 Temperature sensor in the motor cables

Temperature sensors placed in the motor cable must not be wired to signal pins!

Note insulation class! At the sensor pins, 50V is the maximum measuring voltage.

10.4 Configuring feedback

Information on the individual feedback systems can be found in sections 3.3, 3.4, 3.5 and 3.6 .

Several sets of feedback can be used for an axis. The typical usage involves an encoder that sits directly on the motor shaft and an additional encoder, e.g. on a gauge, that is placed after a gearbox or spindle.

In the case of applications with several feedback systems, the encoder sitting on the motor shaft is used for field control.
The second feedback system is used as feedback for the position controller in the drive.
Both sets of position feedback can be sent to the master via the fieldbus.

If there is only one set of feedback available on the motor shaft, this is used for the field control and for the position control.

`Ctrl.MotorConfig.FB_MotorField`	Feedback for field control
`Ctrl.MotorConfig.FB_PositionCtrl`	Feedback for position control

Example 1
One feedback system, incremental encoder:

Field control	Ctrl.MotorConfig.FB_MotorField	PM Encoder
Position control	Ctrl.MotorConfig.FB_PositionCtrl	Encoder
GinLink Feedback	Ctrl.MotorConfig.GinLink.FB_Ch_0	Encoder 32Bit
GinLink target value	Ctrl.MotorConfig.GinLink.Cmd_Ch_0	Speed

Example 2
Resolver on the motor shaft, SinCos on gauge, GinLink feedback channel 0
(32-bit wide position value)

Field control	Ctrl.MotorConfig.FB_MotorField	PM Resolver
Position control	Ctrl.MotorConfig.FB_PositionCtrl	Resolver
GinLink Feedback	Ctrl.MotorConfig.GinLink.FB_Ch_0	SinCos *1024 32Bit
GinLink target value	Ctrl.MotorConfig.GinLink.Cmd_Ch_0	Speed

Example 3
Resolver on the motor shaft, SinCos on gauge, GinLink feedback channel 1
(32-bit wide position value)

Field control	Ctrl.MotorConfig.FB_MotorField	PM Resolver
Position control	Ctrl.MotorConfig.FB_PositionCtrl	SinCos
GinLink Feedback	Ctrl.MotorConfig.GinLink.FB_Ch_0	Resolver *65536 32Bit
GinLink Feedback	Ctrl.MotorConfig.GinLink.FB_Ch_1	SinCos *1024 32Bit
GinLink target value	Ctrl.MotorConfig.GinLink.Cmd_Ch_1	Speed

Resolution of analogue feedback systems
The resolution of resolvers is 16 bit, 65'536 values;
the resolution of SinCos encoders is 10 bit, 1'024 values.

10.5 Configuring fieldbus communication on the controller

Also see section 3.13 for more information.
Inco path for GinLink configuration Ctrl.MotorConfig.GinLink

1. Vcmd_100% Maximum speed of rotation occurring in the axis. This is a standardisation factor
   Must match the value in the IMD project configuration!

2. LinkSamplingRate Sampling rate of the superordinate position control of the fieldbus master.
   Must match the value in the IMD project configuration!

3. Cmd_Ch_0 Standard: Speed 32Bit

4. Cmd_Ch_1 Standard: MasPosErr 32Bit

5. Cmd_Ch_2 Standard: +Iq 32Bit

6. FB_Ch_0 Select corresponding feedback via the GinLink.
   This selection determines the value for IncsPerTurn in the motor configuration in the IMP project configuration

7. FB_Ch_1 Standard: Additional feedback, select corresponding feedback.

8. FB_Ch_2 Standard: Active current of the controller Iq 32 bit.

10.5.1 Configuration example

SinCos encoder	512 strokes
Speed of rotation	3000 rpm
Sampling rate	2 ms
FB channel	Channel 0

Number of increments per revolution: 512 strokes * 1024 = 524'288 Inc/T

$$\text{Number of increments per sampling rate:} \quad \frac{524'288\ \text{Inc/s} * 3000 * 2\text{ms}}{60\text{s}} = 52428.8\ \text{Inc/2ms}$$

Number of bits to be transmitted by +- 52428.8 increments: = 17 bits (32 bit max.)

FB_MotorField	PM SinCos
FB_PositionCtrl	SinCos
FB_Ch_0	SinCos*1024 32 Bit
Cmd_Ch_0	Speed

IncPerTurn 524'288 512 strokes * 1024 (enter in IMD project configuration)

The value should not drop below the minimum resolution of 4096 Inc per revolution.

Example 3

SinCos encoder	40u separation	Gauge after spindle
Resolver	16-bit resolution	on motor shaft
Speed of rotation	4500 rpm	
Spindel	5mm/U	
Sampling rate	0.5 ms	
FB channel	Channel 0	SinCos
FB channel	Channel 1	Resolver

$$\text{Number of increments per revolution (of SinCos):} \quad \frac{5\text{mm} * 1024}{40\text{um}} = 128'000 \text{ Inc/MotorTurn}$$

$$\text{Number of increments per sampling rate:} \quad \frac{128'000 \text{ Inc/T} * 4500 * 0.5\text{ms}}{60} = 4'800 \text{ Inc/0.5ms}$$

Number of bits to be transmitted by +- 4800 increments: = 14 bits (32 bits max.)

FB_MotorField	PM Resolver	
FB_PositionCtrl	SinCos	
FB_Ch_0	SinCos*1024 32 Bit	
FB_Ch_1	Resolver*65536 32 Bit	
Cmd_Ch_0	Speed	
IncPerTurn	128'000	125 * 1024 (enter in IMD project configuration)

10.6 Configuring the fieldbus communication in the software

Each axis also requires corresponding configuration in the software. This is necessary in order for the SAM to be able to correctly convert and interpret the data that it receives from the controller via the fieldbus. Certain standardisations are also required for trouble-free functioning of the axis on both sides (SAM and controller).
More information can be found in section 5.

10.7 Commissioning the feedback system

During the configuration of the feedback system (incremental encoder, resolver or SinCos), the number of increments per revolution must be entered.

For configuration of feedback see section 10.4.

10.7.1 Checking the direction of rotation

1. Move motor shaft or linear motor in a positive direction; the position must count upwards.

2. The encoder value must count upwards.
   Incremental encoder Ctrl.Actual.Encoder
   Resolver Ctrl.Actual.Resolver
   SinCos Ctrl.Actual.SinCos

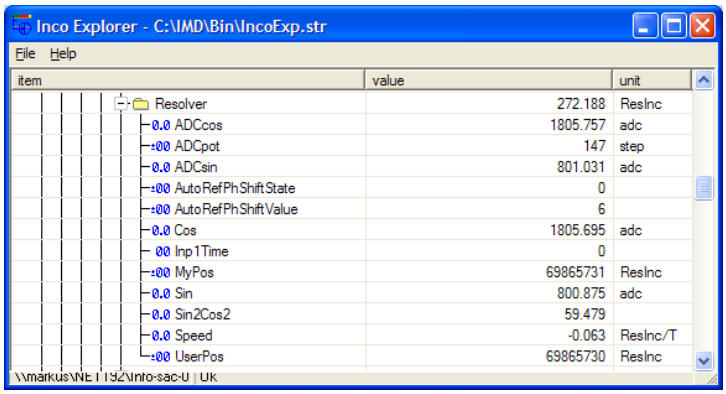


Fig. 50: Actual values, resolver

For the resolver and SinCos encoder, make an additional check to see whether the value for Sin2Cos2 is within the limits Sin2Cos2_Max. Sin2Cos2_Min . If the Sin2Cos2 is too low, check the wiring and the assembly of the encoder.

3. If the direction of rotation is incorrect, invert the flag for the direction of rotation:
   ...\\Ctrl.MotorConfig.Encoder.Flag.direction
   ...\\Ctrl.MotorConfig.Resolver.Flag.direction
   ...\\Ctrl.MotorConfig.SinCos.Flag.direction

Once the flag for the direction of rotation has been changed, the axis needs to be re-commutated! To do this, burn the parameters to the flash prom and disconnect the power to the drive. Or repeat the commutation by hand using test mode.

10.7.2 Standard direction of rotation

Direction of rotation forwards = Look at the motor from behind; the motor must be turning clockwise.

10.7.3 Checking the resolution of the encoder

1. Set the User Pos of the corresponding encoder to 0:
   Ctrl.Actual.Encoder.UserPos = 0
   Ctrl.Actual.Resolver.UserPos = 0
   Ctrl.Actual.SinCos.UserPos = 0

2. Turn the motor shaft 360° in a positive direction as precisely as possible; move the linear motor
   one magnet spacing in a positive direction as precisely as possible.

3. The User Pos should now display the value (positive) that has been configured.
   The turning or shifting of the axis should be carried out in such a way
   that it is possible to recognise the difference between encoders with 1024 and 1000 strokes.

10.8 Checking the actual position in the fieldbus master

Once the configuration has been entered into the IMD project and the servo controller, it can be checked:

Once the software configuration has been adapted, reload to target (trans)!

1. To do this, select the corresponding axis under “Axis” in the fieldbus master:

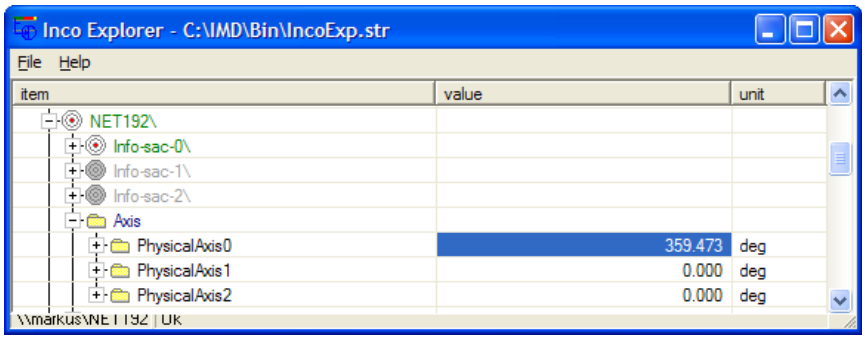


Fig. 51: Position of the axis

2. Set the position to zero by hand

3. Turn by exactly 360° at the motor shaft
   Shift the linear motor by exactly one magnet spacing

4. The sign for the position and the value for the position must correspond to the movement.

10.9 External controller release

1. The external controller release ExtEn must be wired to 24V.
2. The two safety 24V inputs must be switched on. (Stecker X100)
3. The green LED on the front plate of the controller must light up.
4. The configuration under Ctrl.MotorConfig.Enable must be set in accordance with the requirements of the application.

More information on the configuration of the external enabler can be found in section 3.10

10.10 PWM

The sampling frequency of the output stage can be reconfigured in the motor configuration file under `Ctrl.Motorconfig.PWM.PWM` . This allows the user to individually adapt to match the requirements without requiring changes to the controller file (see section 6). In the case of a multiple controller (or SAC3x3), this configuration is only possible on axis 0 and then applies to all other axes of the controller.
Information on the PWM settings can be found in section 3.15.

10.11 Power

The power supply unit of the controller is configured under `Ctrl.MotorConfig.Power`. In the case of a multiple controller (or SAC3x3), configuration of the power supply unit is only possible on axis 0 and then applies to the entire controller.
Information on the power supply unit settings can be found in section 3.17.

10.12 Position controller

The type of control of the axis is specified under `Ctrl.MotorConfig.PositionCtrl`. In the case of axes without position feedback or open-loop stepper motors, virtual feedback of 4096 IncPerMotTurn is generated.
More information on the position controller can be found in section 3.16.

10.13 Finding and verifying the number of pole pairs

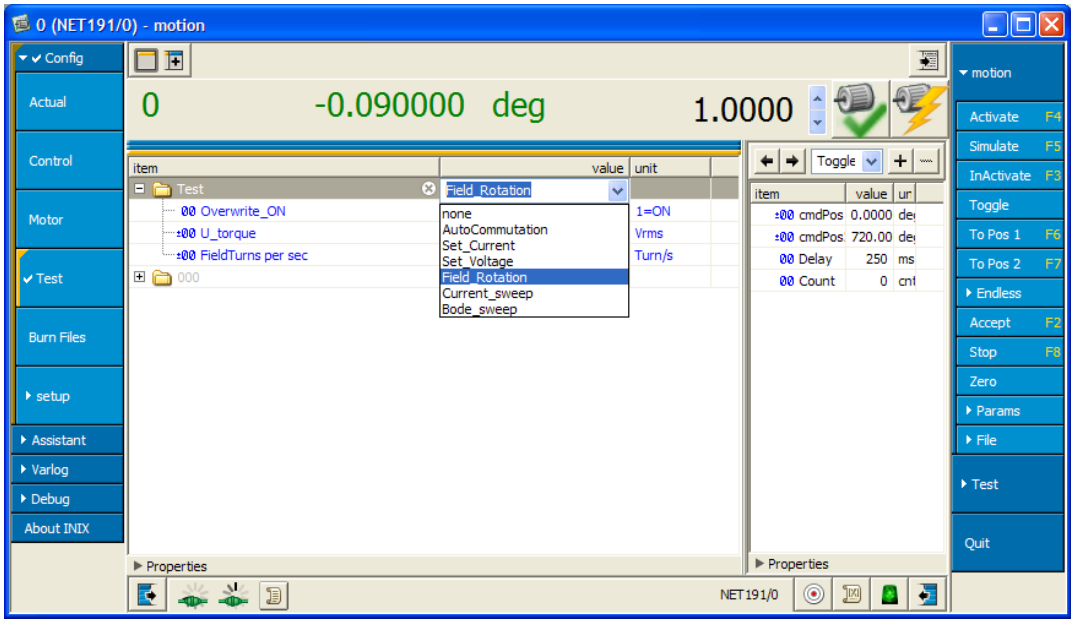


Fig. 52: Feld-Mode

U_torque Motor voltage
FieldTurns per sec Number of field revolutions per second

1. Run the motor without a load! Under Test select Field_Rotation mode

2. Select voltage U_torque , start with 1V, slowly increase the voltage value.
   Select number of field revolutions FieldTurns_Per_sec default value = 1
   Electrical rotational field turns 360° per second

3. Switch controller to simulation mode. To do this, use F3, F5.
   If the motor is still not turning at 10V, you must check the wiring.

Number of pole pairs	U/s
1	1
2	1/2
3	1/3
4	1/4

For precise determination of the number of pole pairs, it is recommended that you keep a log.

10.14 Verifying the direction of rotation (before commutation)

This test can also be carried out without commutation.

1. Run the motor without a load! Under Test select Field_Rotation mode

2. Select voltage U_torque , start with 1V, slowly increase the voltage value.
   Select number of field revolutions FieldTurns_Per_sec default value = 1
   → Electrical rotational field turns 360° per

3. Switch controller to simulation mode. To do this, use F3, F5.
   The motor must now be turning forwards!

If the direction of rotation is incorrect, invert the flag for the direction of rotation in the motor configuration: Ctrl.MotorConfig.Motor.Flag.direction

Once the flag for the direction of rotation has been changed, the axis needs to be re-commutated! To do this, burn the parameters to the flash prom and disconnect the power to the drive. Or carry out the commutation by hand using test mode.

10.15 Adjusting the current controller

The different current controller versions and parameters are described in section 3.9.

The parameters for the current controller can be automatically calculated by the SAC drive.

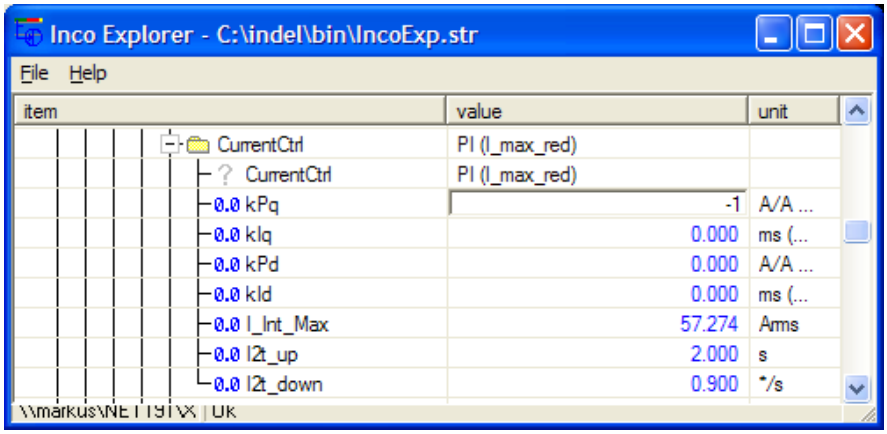


Fig. 53: Calculating the current controller

At parameter Ctrl.MotorConfig.CurrentCtrl.kPq
enter (p component of active current controller) -1 .

After this, the parameters are automatically calculated for the active and idle current controllers. The calculated kP is the maximum possible value for stable current control! Normally, the P component must be reduced slightly in order to reduce whistling noises.

In the case of linear motors, the calculated P values are normally very high (approx. 50). The P value must be reduced depending on the noise build-up when activating the axis (F3, F2, F5).

In the case of axes with distinctive resonance points, the calculated kPq, kPd may be so high that the axis begins to whistle loudly as soon as it is activated in simulation mode. In this case, the kP values should be reduced in such a way that the whistling disappears.

In the case of iron-free linear motors and high sampling rates (16kHz), the integrator time constant can be very small: < 200us

If the kP is increased by the current controller in an axis that has already been set, the kP may need to be reduced by the position controller. The current controller is beneath the position controller, meaning that there is a dependency between the two controllers.

Idle current controller

The P component of the idle current control does not necessarily need to have the same value as the P of the active current controller. If the idle current integrator does not increase considerably, the idle current P can be reduced considerably. Particularly in linear motors, the control is very good, meaning that kPd can be reduced to half kPq.

Essentially, in cases of large P values (30 ... 50) the active current P can be reduced until the noises are bearable. Reduce the idle current P until the integrator begins to increase, typically kPd = 1 ... 5.

Calculation principles

The automatic adjustment of the current controller is based on the resistance and inductance of the motor; furthermore, the sampling frequency of the drive has a role to play.

I2t control

I2tup, I2tdown	See section 3.9 Current controller: Current Control
I_Int_Max	3 times I_{MAX}

10.16 Commutation

The commutation leads to the field offset (angle) between the electrical field in the stator and the magnets of the rotor being determined.

In **motors with a resolver** the field offset only needs to be set once. All motor manufacturers usually supply their motors with the same field offset. This means that the resolver is always assembled with the same alignment to the rotor. The field offset of resolvers is stored in the motor configuration file.

In **motors with an absolute feedback system** the field offset generally needs to be set each time the encoder is mounted on the motor. This also applies in the case of servicing, if a motor or an encoder needs to be replaced. The field offset of absolute feedback systems is stored in the motor configuration file.

Motors with an incremental encoder or sine-cosine encoder need to be commutated each time the drive is switched on. Various auto-commutation methods are available for this.

If the field offset is not right, the motor cannot be run properly. This manifests itself in the fact that, for example, the maximum performance or maximum speed of rotation cannot be reached. In the worst case scenario, the motor will turn backwards!

An optimally set field offset will increase the efficiency of the drive.

10.16.1 Auto-commutation with sine-cosine and incremental encoders

Incremental encoders, sine-cosine encoders and resolvers can be commutated automatically.

1. Under Ctrl.MotorConfig.AutoCommutation select the commutation procedure and parameterise. (See 3.8 Auto-commutation)

2. Select the flags in the auto-commutation:

   ON_If_Ok=0 If the commutation was successful, the axis remains active.

   Again=1 This means that the operating mode always remains auto-Commutation.

3. Under Ctrl.Test select AutoCommutaion mode

4. Activate the axis in simulation mode using F3, F5

5. Under Ctrl.Actual.AutoCommutation.Ok check whether the commutation was successful.

6. Repeat the commutation several times at various positions.

   The field offset Ctrl.Actual.AutoCommutation should not deviate by more than +-10° across the entire travelling area.

7. Reset the flags in the auto-commutation; test mode to none.

   ON_If_Ok=1
   Again=0

Never activate the axis (either in simulation mode or active mode) if the commutation is not working perfectly!

10.16.2 Auto-commutation with absolute encoders

1. Configure absolute encoder; set auto-commutation method to Absolut Encoder
.

2. Determine the field offset by hand. (Section 10.16.3 Adjusting the resolver offset by hand)

3. Under Ctrl.Test select AutoCommutaion mode

4. Carry out auto-commutation

5. Burn the motor parameters to the flash and save in a file.

10.16.2.1 Auto-commutation with Hiperface

The right field offset must be found and set when the Hiperface is commissioned for the first time. The following steps must be carried out the first time the motor is commissioned.

1. Configure Hiperface.

2. Check the direction of rotation of the analogue (SinCos) and digital feedback of the Hiperface. The counting direction of both sets of feedback must match (counting direction of the increments).

3. Set auto-commutation to absolute encoder.
The following flag must not be set!

   ON_If_Ok=0

4. Carry out commutation with Hiperface in test mode. Now the position of the absolute encoder is synched.

5. The field offset then needs to be found. There are two options available.
To do this, the motor must be able to move freely.

   1) Adjust the field offset using auto-commutation mode (see 3.8).
   2) Adjust the field offset by hand (see 10.16.3)

   The field offset has now been found and is entered under `MotorConfig.FB_MotorField`.

6. Now set auto-commutation back to `Absolute Encoder`. Test the commutation using test mode. In addition, the drive should also be switched off and back on once. To do this, you must first burn the motor configuration file.

7. If the commutation was successful, the following flag can be set in auto- commutation. The axis is then automatically switched to active following successful commutation.

   ON_If_Ok=1

10.16.3 Adjusting the resolver offset by hand

In order to get the best possible result for the field offset, the field offset should be adjusted by hand.

One of the auto-commutation methods can essentially be used in order to gain an initial value for the field offset of the resolver. Wherever possible you should use auto-commutation with 360° field rotation (see section 3.8.5). Once the auto-commutation has been carried out, the field offset is automatically stored under `Ctrl.MotorConfig.FB_MotorField.Field_Offset`.
Then switch the auto-commutation back off and carry out the following steps for the exact field offset.

1. Phases and resolvers must be connected correctly.
   The motor must be able to turn freely, without a load

2. Under Ctrl.Test select Set_Current mode
   → Idle and active current can be entered by hand.

3. Switch on external release; green LED "Ext.En" on the servo controller must light up
   Safety inputs must also be switched on.

4. ca. 1/10 I_{NENN} Set idle current at "`I_Reactive`" and switch the controller to simulation mode. To do this, use F3, F2, F5. If the motor is turning, the correct resolver offset has not yet been found. Use F3 to switch the motor off again.

5. Resolver Offset Ctrl.MotorConfig.FieldOffset change until the motor is at a standstill.
   (try by hand to see which direction it turns more easily in)

6. When the motor is at a standstill, continually increase the idle current to I_{MAX}. Only briefly, approx. ½ second PowerOn/PowerOff or repeat F3, F5, F3 until the motor is at a standstill at `I_Reactive` = I_{MAX} .

7. "`I_Reactive`" = 0V

8. Check whether the right resolver offset has been found:

9. Under Ctrl.Test select Set_Voltage mode
   → Idle and active voltage can be entered by hand.

10. Set U_torque to 1 ... 10V. Switch controller to simulation mode.
    To do this, use F3, F5. The motor must be turning forwards! Direction of rotation forwards = Look at the motor from behind; the motor must be turning clockwise.

    If the motor is not turning clockwise, an incorrect resolver offset has been found.
    (There are several).

11. BurnToFlashProm, SaveToFile, U_torque = 0V, Flag = 0

10.17 Verifying the direction of rotation (after commutation)

For this test, the commutation must have been carried out successfully.

1. Run the motor without a load! Under Test select Set_Voltage mode

2. Select voltage U_torque , start with 1V, slowly increase the voltage value.
   Warning: The motor turns more quickly at 1V than in field rotation mode!

3. Switch controller to simulation mode. To do this, use F3, F2, F5.
   The motor must now be turning and counting forwards!

10.18 Gain offset correction for resolvers and SinCos

Gain and offset errors as well as gain asymmetry in resolvers or SinCos encoder systems can be evaluated and corrected in the software.

The “SinCos” assistant can be used to do this. The assistant is a component of the software directly in the drive.

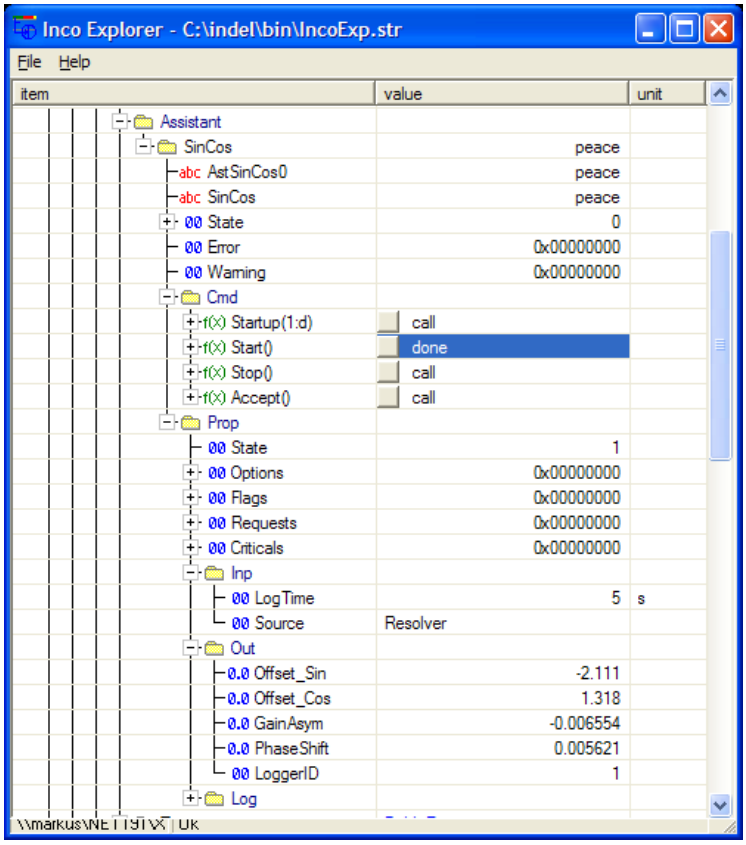


Figure 10.1: SinCos adjustment

10.18.1 Resolver adjustment

Resolver adjustment

1. Allow the motor to run at a constant 100 rpm (corresponds to 600°/s)
2. Start SinCos assistant Start()
3. Wait for calculation of gain, offset, etc.
4. Accept adjustment values Accept()
5. Burn and save parameters

Ctrl.Assistant.SinCos.Cmd.Start() can be started. The logger time can be left at 5s (default value).

The status of the procedure is displayed in Inco Explorer:

- Waiting for logger, please move axis
- Logging
- Calculating Correction Values
- Peace

Once the test has been carried out, the results of the adjustment are displayed under Ctrl.Assistant.SinCos.Prop.Out .

These must now be accepted using Ctrl.Assistant.SinCos.Cmd.Accept() . Finally, the adjusted values must be burned to the flash prom and saved.

10.18.2 SinCos encoder adjustment

Wherever possible, move the axis along the entire area of the feedback system. In the case of rotational feedback systems, at least one motor revolution is sufficient; in the case of linear feedback systems, the entire gauge must be travelled.

The speed must be selected in such a way that at least 40 measurement values of a full sine oscillation can be recorded per sampling period.

Example
Rotational encoder with 1024 sine periods
Sampling rate 16kHz

Speed of rotation for SinCos adjustment

$$\frac{16000\ \text{kHz}}{1024 * 40} = \mathbf{0.39\ U/s\ (140°/s)}$$

Under Ctrl.Assistant.SinCos.Prop.Inp.LogTime you can enter the time for the log. This is where the time for the entire movement is entered.

SinCos adjustment

1. Allow the motor to run along the entire length of the feedback system
2. Enter the time for the movement LogTime
3. Start SinCos assistant Start()
4. Wait for calculation of gain, offset, etc.
5. Accept adjustment values Accept()
6. Burn and save parameters

10.19 Adjusting PID parameters

The PID parameters for the motor must essentially be adjusted with a load. When adjusting the PID parameters for the first time, use a motor without a load for reasons of safety! If the motor vibrates too heavily, this can damage the mechanics.

The PID parameters can be determined using different approaches:

- impact response, step response; optimisation according to Chien, Hrones and Reswick
- Optimisation procedure according to Ziegler and Nichols
- “Axis Turner” tool from Indel, see section 11 Bode-Sweep – PID-Wizard

10.19.1 Optimisation procedure according to Ziegler-Nichols

The control route is first run with a simple P controller. The P component is increased until the control route displays oscillations of a constant amplitude. The controller can be set optimally from the set critical P component (kP crit) and duration (Tk) of the resulting oscillation.

Controller type	I component	D component	P component
P	-	-	0.5 * kPKRIT
PD	-	0.125 * TKRIT	0.8 * kPKRIT
PI	0.85 * TKRIT	-	0.45 * kPKRIT
PID	0.5 * TKRIT	0.125 * TKRIT	0.6 * kPKRIT

10.19.2 Procedure for adjusting the PID parameters

Log the following parameters at 1kHz

Intermediate circuit voltage	U_{CC}	Ctrl.Actual.Power
Target speed	V_{CMD}	Ctrl.Actual.PositionCtrl.cmd_V
Actual speed	V_{ACT}	Ctrl.Actual.PositionCtrl.act_V
Following error	S_{ERR}	Ctrl.Actual.PositionCtrl.err_S
Integrator	S_{INT}	Ctrl.Actual.PositionCtrl.err_S_int
Active current	I_Q	Ctrl.Actual.CurrentCtrl.act_Iq

1. PID parameters in Inco-Tree: `Ctrl.MotorConfig.PositionCtrl`

2. Set all filters to none : Ctrl.MotorConfig.Filter.Filter_0,1,2,3
   Also set speed filter to none : Ctrl.MotorConfig.SpeedFilter

3. Default parameters:

   `HoldToStandbyTime` = 100ms

`PID stand-by`	=	0	set all parameters to 0
`PID forward`	=	0	set all parameters to 0
`PID backward`	=	0	set all parameters to 0
`phvSpeed, phvAcc`	=	0	set all parameters to 0

4. As a starting value, select `kP` = I_{NOM} / 50
   Ctrl.MotorConfig.PositionCtrl.forward.kP

   Example Motor with I_{NENN} = 2.4 A -> kP = 0.05

5. Change to the fieldbus master. Specify a small ramp:

Axis.PhysicalAxis0.Cmd.Test.cmdPos1	=	360°
Axis.PhysicalAxis0.Ramp.Cmd.cmdV	=	600 °/S (100 U/min)
Axis.PhysicalAxis0.Ramp.Cmd.cmdA	=	5000 °/S2
Axis.PhysicalAxis0.Ramp.Cmd.cmdB	=	5000 °/S2
Axis.PhysicalAxis0.Ramp.Cmd.cmdJ	=	0%

   You can also select a steeper ramp, e.g. 50'000 °/s2. This has the effect that natural resonance is generated.

   Consider the characteristics of the motor and load!

   Travel this ramp in **simulation mode F3, F5**. Start ramp travel using F7. If the motor is vibrating heavily, immediately switch it back off using F3 (Inactive) or F8 (Stop) and reduce the value of kP.

   This information applies to a gear of 1:1, or refers to the motor shaft!

6. Increase "`kP PID forward`" until the vibration in the horizontal section of the travelled trapezoid does not increase or decrease. -> Use logger.

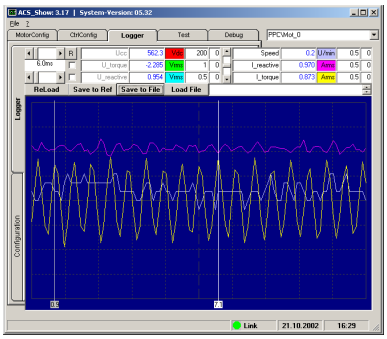


Fig. 54: Critical amplification kP crit

From fig. 7.9.1: this is the critical amplification "kP crit"; the period of vibration measured with the logger is the critical period of time "Tk".

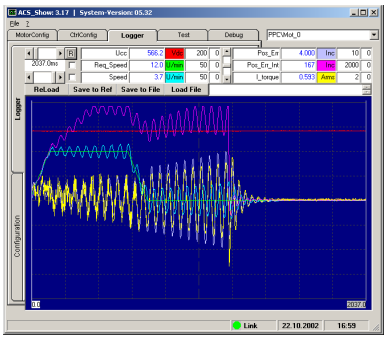


Fig. 55: Critical amplification kP crit too high

7. Apply the following formulae for kP, kI and kD:

	kP	=	0.6 * kPkrit
	kI	=	0.5 * Tk
	kD	=	0.12* Tk
From example:	kP	=	0.6 * 0.13 = 0.78
	kI	=	0.5 * 54 = 27 ms
	kD	=	0.12 * 54 = 6.48 ms

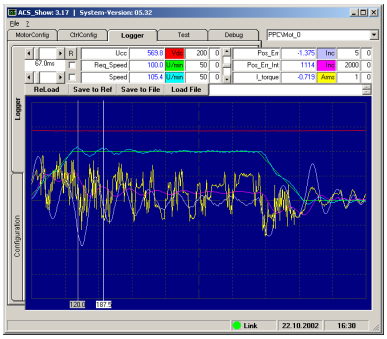


Fig. 56: First version of the PID parameters

8. Continually increase the value for kP. This means that the following error is reduced and the curve becomes smoother. At the same time, the ramp should also be changed:

Axis.PhysicalAxis0.Cmd.Test.cmdPos1	=	3'600°
Axis.PhysicalAxis0.Ramp.Cmd.cmdV	=	6'000 °/S (100 U/min)
Axis.PhysicalAxis0.Ramp.Cmd.cmdA	=	5'000 °/S2
Axis.PhysicalAxis0.Ramp.Cmd.cmdB	=	5'000 °/S2
Axis.PhysicalAxis0.Ramp.Cmd.cmdJ	=	0%

9. Repeat steps 6 and 7 above until you reach a curve form with the following criteria:
   - `Req_Speed` and `Speed` are congruent
   - After travelling the ramp, the parameter `ANZ_S` matches the specification `VRG_S` at +- 1%.
   - No extreme vibration behaviour in the event of current and following errors.

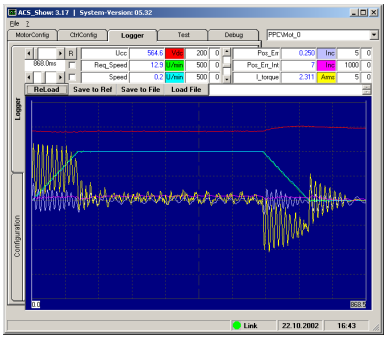


Fig. 57: kP was increased until the current and following error vibrate again

10. From example: kP was increased to 0.6 -> at this setting, the current and following error begin to vibrate again.

    Reduce kP again to approx. 80 ... 90% of the critical value: kP = 0.5

11. The PID parameters for `"PID stand-by"` arise from the “average” of `forward` and `backward`. For `kP stand-by`, enter approx. half of the average.

12. The motor should essentially run with these parameters. The fine-tuning that is still required or the adjustment with a load require a high degree of experience, and sometimes also patience. We will be happy to help you in setting the control weights.

13. Until now, the motor has always run in simulation mode. For the fine-tuning, the motor is run in `"Active"` (function button F4) operating mode.
    This switches on the superordinate position control.

14. The found parameters must be loaded into the flash prom of the controller using `"Burn Values to Target"`
    otherwise the values will be lost when you power the unit off. Finally, the values should be saved in a file using `"Save Values to File"`.

15. Apply the same procedure for `PID backward`.

10.20 Adjusting lead values

phvSpeed

The lead value for speed is a target current specification. It removes speed-related losses.

1. Measure the current during the highest possible constant travel

2. Speed-related lead value:

   phvSpeed = const current / const v

   Example:
   Constant travel at 3000 rpm with 6A
   phvSpeed = 6A / 3(1000)U/min = 6 / 3 = 2 A@Spd

3. The integrator (Pos_Err_int) should now remain constant during continual travel.
   Optimal: Pos_Err_int < 1000 Inc

phvAcc

Is an acceleration or retardation lead value that adds the current required for the acceleration / retardation to the target value.

PhvAcc is defined as current consumption in A at acceleration / retardation of 0 at 1000 rpm in one sec.

1. Measure the current during the longest possible retardation journey
   -> gives, for example, 12A for retardation in 250ms from 0 to 3000 rpm

2. **phvAcc = 12Arms / 3 (1000)U/min * 025sec = 12 / 3 * 0.25 = 1 A@acc**

3. The integrator (Pos_Err_int) should now be symmetrical during the acceleration and braking ramp

10.21 Fine-tuning of Ke, Rs and Ls

This adjustment is not compulsory. If the data from the data sheet is entered with the correct standardisation for strand values (phase-phase), the motor should also work without any problems!

Fine-tuning can also be used to verify the values from the data sheet.
This allows errors originating from any old data sheets or incorrect details to be eliminated early. If the Ke is incorrect, this will have a direct impact on the dynamics of the control.

This additional fine-tuning is also worthwhile in the case of applications with large loads or extremely fast ramps.

Changes to Ke, Rs, Ls will only be accepted if the axis is switched to inactive and then back to active.

Uq	U Torque	Active voltage
Ud	U Reactive	Idle voltage
Iq	I Torque	Active current
Id	I Reactive	Idle current
Err_Iq_Int	I Torque Integrator	Integrator active current (current controller)
Err_Id_Int	I reactive integrator	Integrator idle current (current controller)

The sequence must be adhered to!

1. Set field offset (resolver offset) when the unit is at a standstill using Imax

2. Set Ke in such a way that Err_Iq_Int is 0, wherever possible, at full rotational speed (without load)

   Uq = Iq*Rs + wKe = Iq=0 = 0 + wKe remains just –error

3. Set Ls in such a way that Err_Id_Int is 0 (symmetrical), wherever possible, during up/down ramp

   If Err_Id_Int is neg at ramp up: Ls too small
   If Err_Id_Int is pos at ramp up: Ls too large

4. Set Rs in such a way that Err_Iq_Int is constant (and near 0), wherever possible, during ramp.
   If Err_Iq_Int is neg at ramp up: Rs too small
   If Err_Iq_Int is pos at ramp up: Rs too large

10.22 Removing resonance

Methods for reducing high-frequency resonance, caused by gear or belt tolerance:

Speed-Filter: kT_Speed
Resolver filter, time constant for “resolver speed" (1ms).
Ctrl.MotorConfig.PositionCtrl.kT_Speed

- increase kT_Speed in small steps.
- kT_Speed can be selected as approx. 5 ... 10 times smaller than kD.

kd
Ctrl.MotorConfig.PositionCtrl.forward.kd

- kd works best with SinCos encoders (high-resolution encoders)
- can also be used with resolvers, possibly with reduced effect
- a kd that is too great will create whistling noises

kD
Ctrl.MotorConfig.PositionCtrl.forward.kD

- kD as small as possible
- in the case of a smaller kD, kP can be increased

Current controller
Ctrl.MotorConfig.CurrentCtrl.kPq
Ctrl.MotorConfig.CurrentCtrl.kIq
Ctrl.MotorConfig.CurrentCtrl.kPd
Ctrl.MotorConfig.CurrentCtrl.kId

- reduce kPq, kPd

Dead time compensation
Ctrl.MotorConfig.PWM.DeadTime_correction

- switch off dead time compensation (none)

Methods for reducing low-frequency resonance, caused by large masses:

Position integrator
Ctrl.MotorConfig.PositionCtrl.Pos_Int_Max

- reduce integrator limitation (any)
  (Standard: 7600)

kD
Ctrl.MotorConfig.PositionCtrl.kD

- increase D component in the control

Frequency filter 0 ... 3
Adjust filter with bode sweep.

11 Bode-Sweep – PID-Wizard

The existing PID parameters can be optimised using a bode sweep. In addition, you can configure up to 4 bi-quad filters in order to eliminate disruptive resonance / dissonance.

11.1 Motion tool settings

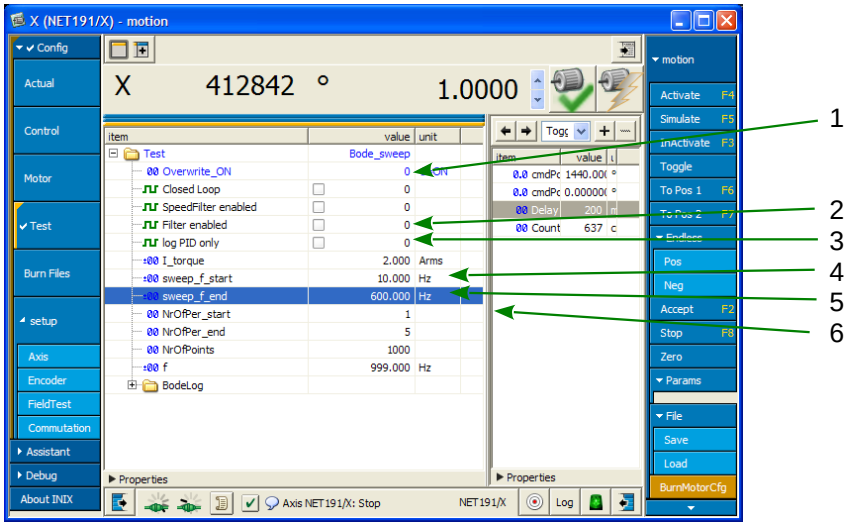


Figure 11.1: Configuring a sweep

1) Test-Mode: Bode sweep (drop-down menu)

2) Activate speed filter for bode sweep
If a speed filter is configured, this flag must be switched on during the bode sweep. Also see section Fehler: Referenz nicht gefunden Fehler: Referenz nicht gefunden

3) Activate current filter for bode sweep
The current filters should only be switched on for the bode sweep if the resonance is so strong that a decisive bode sweep is impossible.

4) Active current for bode sweep

5) Start frequency

6) End frequency

11.2 PID Wizard settings

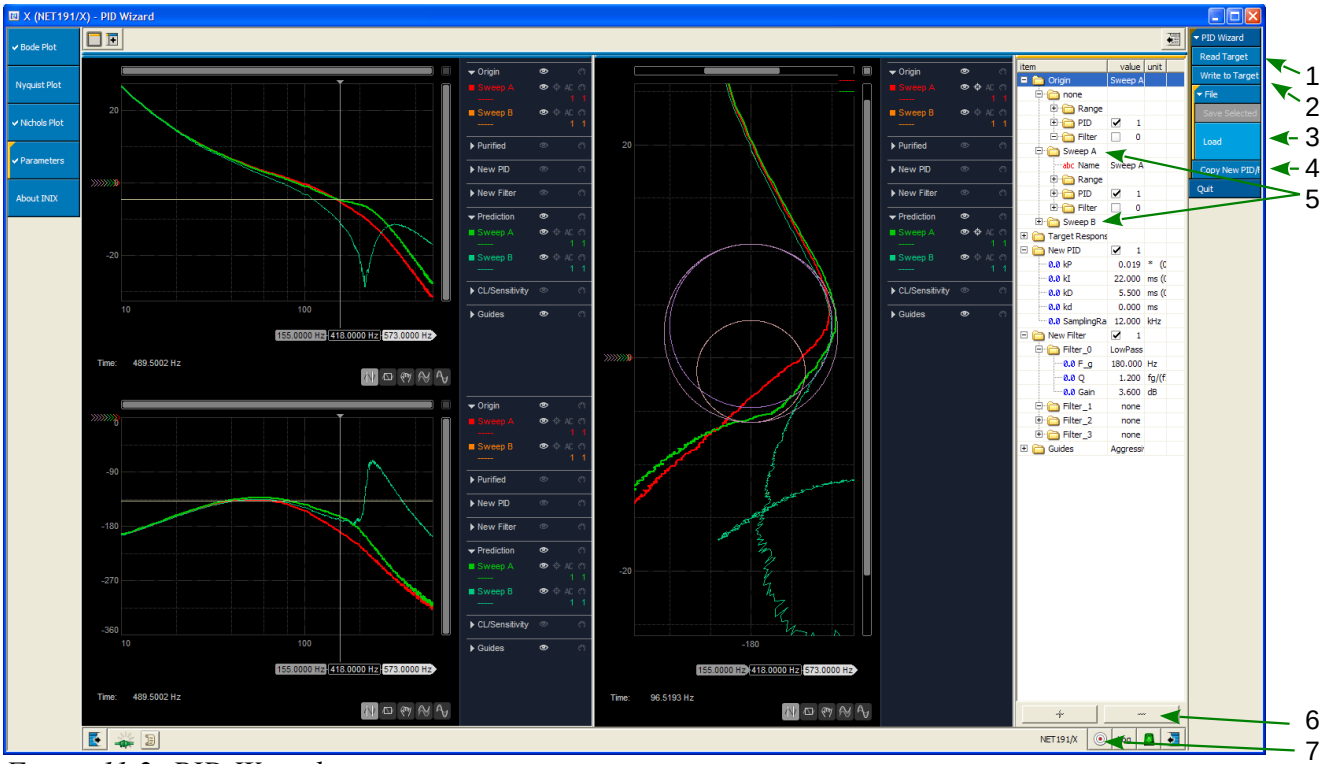


Figure 11.2: PID Wizard

1) Read Target: Load sweep data from controller

2) Write to Target: Copy filter values and PID parameters from the folders New Filter and New PID into the drive

3) Save, load sweeps
Before a sweep is saved, it must be selected in the variables tree.
Also see point 5).

4) Copy PID parameters and filter settings from the drive into the PID Wizard (folders New Filter and New PID)

5) Select a sweep

6) Delete a filter or an entire sweep.
Select the sweep/filter in the variables tree, then delete

7) Target selection: first select the target, then select the drive within the target

11.3 Recording a bode sweep

The first sweep is essentially carried out with roughly set PID parameters and without any filter.

Before starting, the current controller and the SinCos (resolver) must be adjusted. See:

10.18 Gain offset correction for resolvers and SinCos
10.15 Adjusting the current controller

1. Select the test mode Bode_Sweep in the motion tool
2. Delete flags for SpeedFilter, Filter enabled
3. At the first sweep, set I_Torque to approx. 1/4 ... 1/3 of INENN;
   it is best to start with a low current in order to not overload the motor or the mechanics.

   The I2t control should also be correctly set in order to avoid motor overload.
   (See section 3.9.3 I2t control)

   The sweep current should eventually match the current that actually flows during operation.
4. Activate the drive in simulation mode (F3, F2, F5)
5. → The frequency sweep is carried out
6. In the PID Wizard, click on Read Target
7. → The log is loaded and displayed
8. After the bode sweep, set the test mode back to None .

11.4 Procedure for optimising the control route

First sweep

As a first step, create a sweep with roughly set PID parameters and moderate current.

Observer-Filter

Should there already be resonance in the upper frequency range, or in the case of encoders with very low resolution, it is recommended that you configure the observer filter. See Fehler: Referenz nicht gefunden Fehler: Referenz nicht gefunden.

First only set the cut-off frequency of the filter; enter the value kP_Iq = 0.

In the case of encoders with very low resolution from 2048 increments per revolution, the cut-off frequency can be set between F_g = 180 ... 450Hz.

Too low values for the cut-off frequency negatively affect the bandwidth of the system and may lead to deterioration in the quality of control.

For high-resolution SinCos encoders with an interpolated resolution greater than 1'000'000 increments per revolution,the cut-off frequency is, for example, F_g = 600 ... 1000Hz.

When the observer filter is switched on, the flag SpeedFilter enabled must be switched on. See 11.2 PID Wizard settings.

Increase current

Then increase the sweep current and set it in such a way that it roughly matches the current required for the application.

Gain phase reserve in the lower frequency range

In order to gain more phase reserve in the lower frequency range, it has proven useful to use a low-pass filter with positive gain.
The gain can be configured at up to approx. 6dB with a quality factor (Q) of between 0.9 and 1.1.
Also see fig. 63 and fig. 64 in section 11.7.

Optimise PID parameters and filters

First define suitable PID parameters, then start the configuration of the filters.
Generally speaking, the best results are achieved with optimum PID parameters and notch filters.

11.5 Evaluating a bode sweep

Signals smaller than -35dB are no longer relevant for the control and optimisation of filters. The parameter Ctrl.Test.sweep_f_end can be adjusted accordingly.

In this example it is reduced from 1000 Hz (default) to 600 Hz.

The PID Wizard can only display sweeps with the same start/stop frequencies at the same time!

Before the start/stop frequency is changed, all sweeps should be deleted from the PID Wizard. (See Deleting a sweep, page: 105 Paragraph: 5)

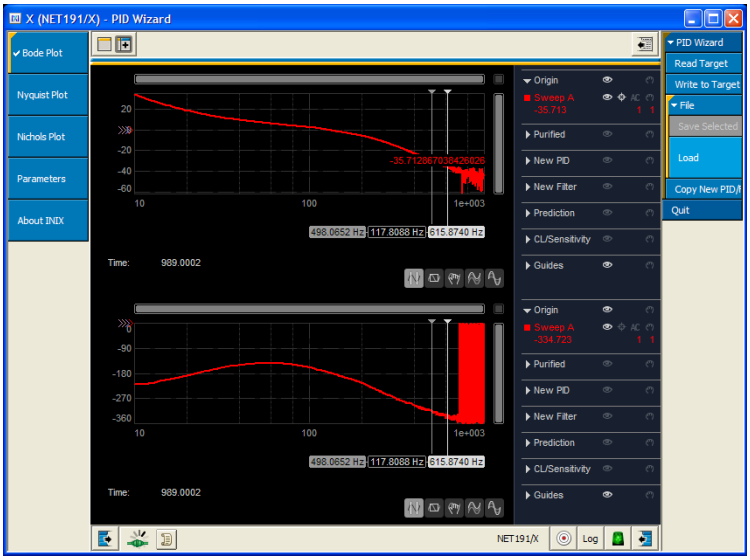


Figure 11.3: Max. Setting frequency

The rest of the parameters should not be changed (except the current).

At the start of the optimisation work, the existing PID parameters and filter values must be accepted in the PID Wizard.

To do this, click on CopyNewPID/Filter from Origin . The parameters are copied into the folders New PID and New Filter .

Then the filters and PID parameters in the folders New PID and New Filter can be adapted for the purposes of optimisation.

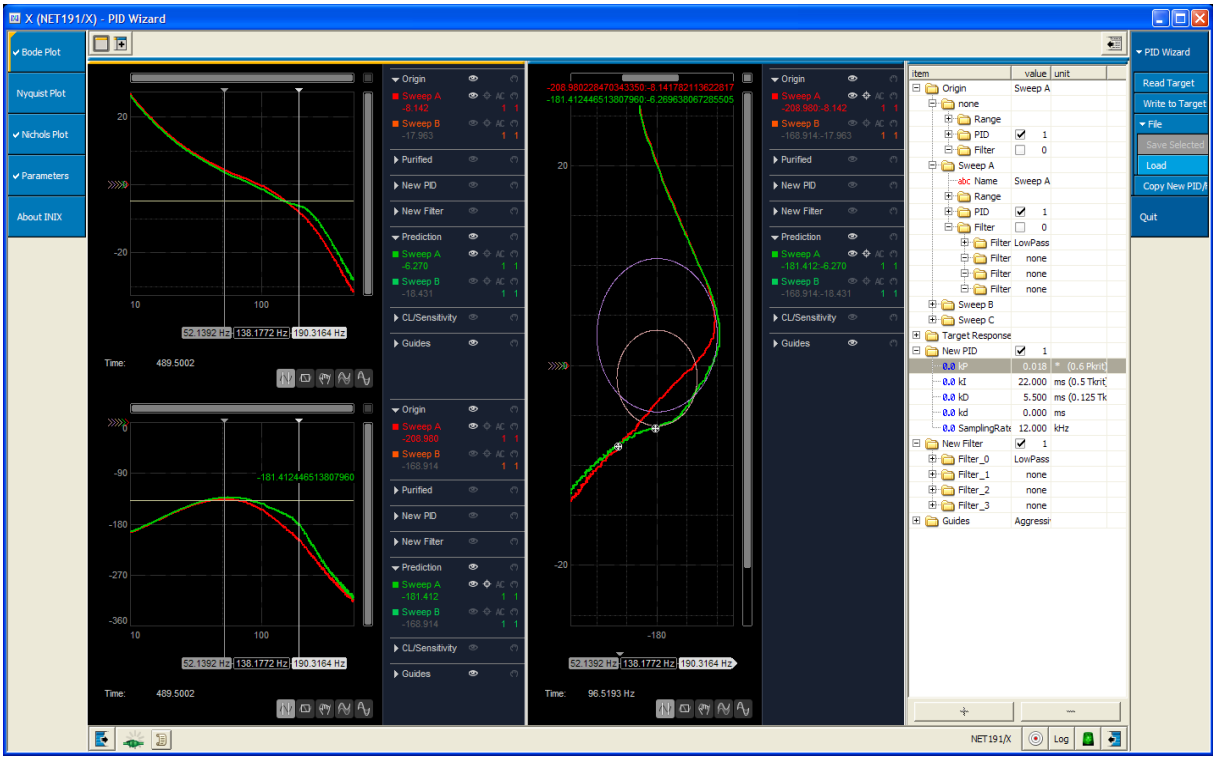


Figure 11.4: Setting filters, PID

Setting aids

Under Guides you can choose to display lines and circles designed to guide you. These lines contain various stability criteria.

There are three different “severity grades” that can be selected for the stability:

- Aggressive
- Moderate
- Conservative

Whereby the control is usually designed with Aggressive cut-off values.

AR	Amplitude margin, 4.6 dB for aggressive: At -180° average for the phase, minimum attenuation of -4.6 dB remains in the open loop
PR	Phase margin In the case of 0 continuity in the signal, a phase reserve of 41.4° remains
Mt	Stability limit, 3 dB reserve for aggressive In the closed loop, the excessive increase in the case of transient response is never greater than 3 dB
Ms	Stability limit, 6 dB reserve for aggressive

11.6 Effect of the PID parameters

kP component

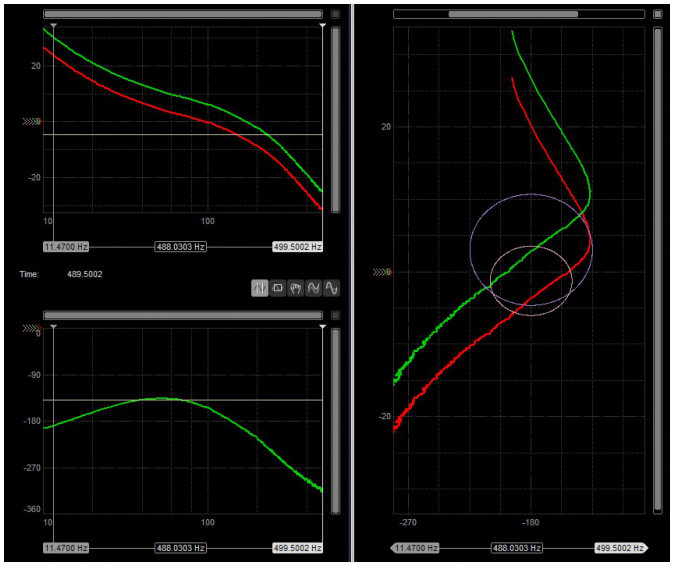


Fig. 58: Effect of the kP component

In the Bode and Nichols diagram, increasing the kP component causes the entire curve to be shifted upwards.

kI component

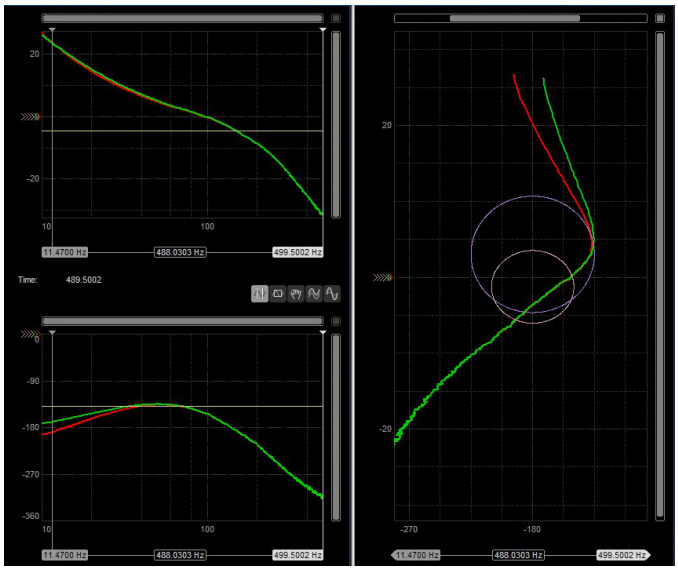


Fig. 59: Effect of the kI component

In the Nichols diagram, increasing the kI component causes the phase reserve to be raised in the low frequency range and the curve to swing to the right.

The ratio between the I component and the D component should always be 4:1 wherever possible.
I component 4 x greater than D component. This corresponds to the theory of Ziegler-Nichols.

kD component

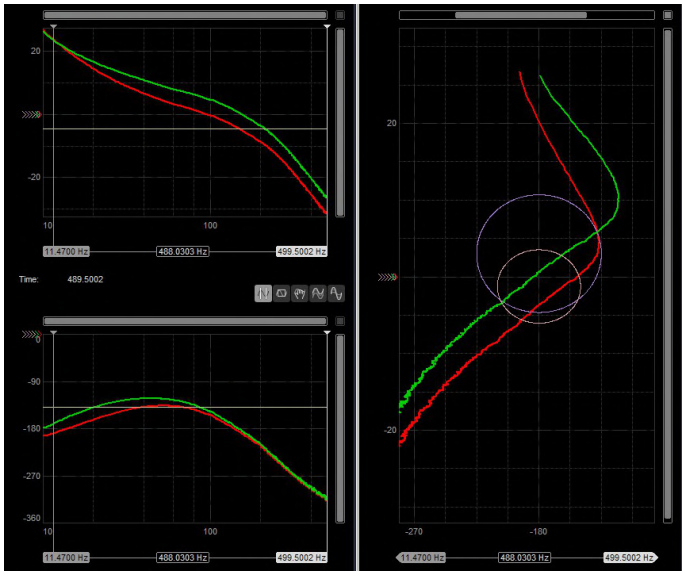


Fig. 60: Effect of the kD component

In the Nichols diagram, increasing the kD component causes the curve to be shifted to the right and upwards. Alongside the increase in the level, the phase reserve also increases.

kd component

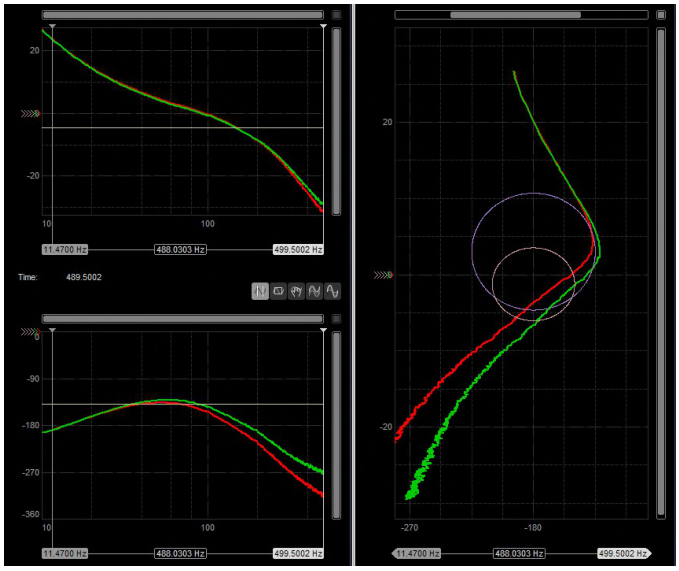


Fig. 61: Effect of the kd component

The kd component increases the phase reserve in the upper frequency range.
A prerequisite for optimum impact from this parameter is high resolution in the encoder system.

The quick changes in the high frequency range must be able to be processed with a corresponding “volume” of path information. The kd works best with high-resolution SinCos encoders.

The kd cannot be chosen at any size. The control route reacts with loud noises and current whirring.

11.7 Current filters

1... 4 current filters can be configured depending on the motion board or servo controller.

You can essentially configure as many filters as the computational power allows. The number of filters therefore depends on the sampling rate of the position controller.

In the case of AX4 or MAX4 boards with their own application, it is possible that there will be no computational power left for current filters.

There are 3 different types of filter available: low-pass, notch and two-load.

Filter-Parameter

The quality can be altered within the range of 0.5 ... 5.

The attenuation can be changed between -80 ... +6dB. Amplification of more than 1dB may not be able to be implemented in the real control route.

Low-Pass

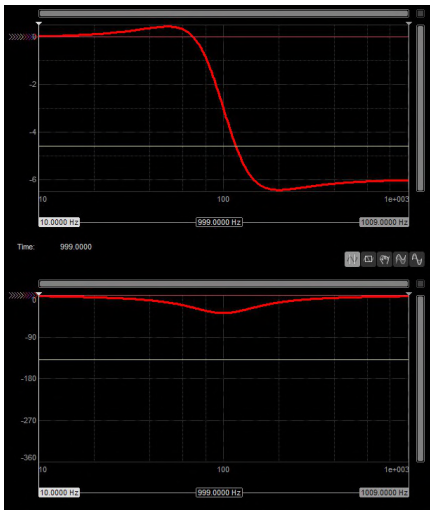


Fig. 62: Low-pass filter -6dB amplification

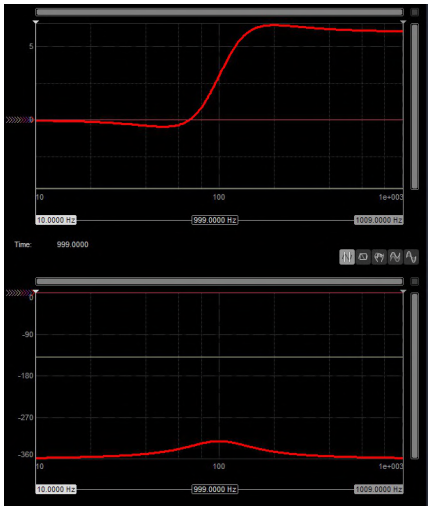


Fig. 63: Low-pass filter 6dB amplification

Low-pass filters with positive amplification can be used to increase the phase reserve.

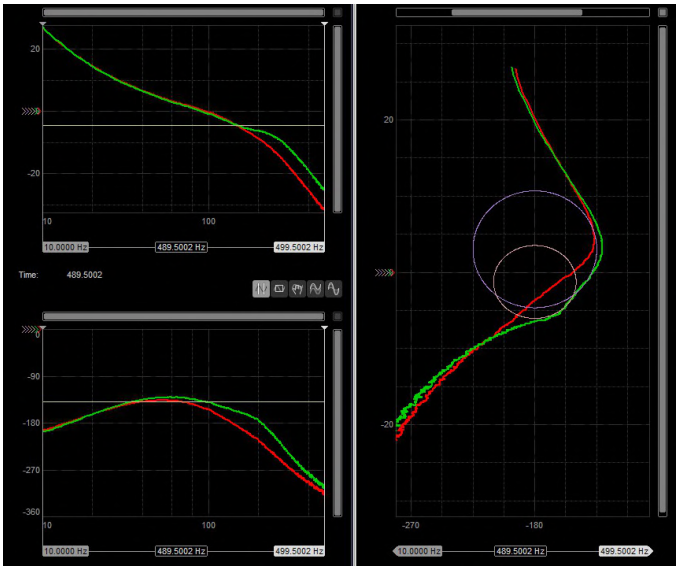


Fig. 64: Example: low-pass filter with 6dB amplification at 210Hz

Notch

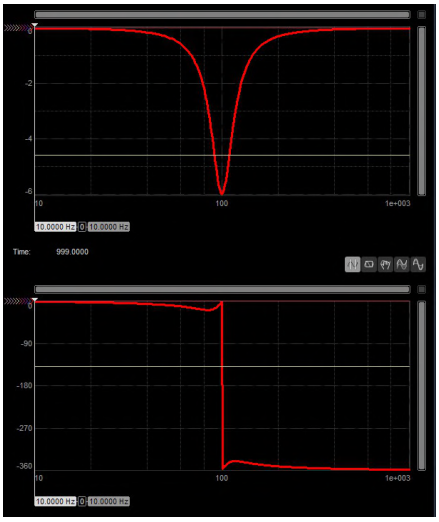


Fig. 65: Notch filter -6dB amplification

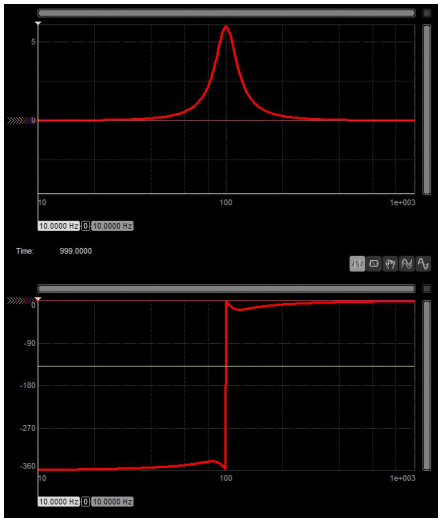


Fig. 66: Notch filter 6dB amplification

Notch filters are particularly suited to reducing excessive increases in resonance or raising the level in a targeted manner.

Two-Load

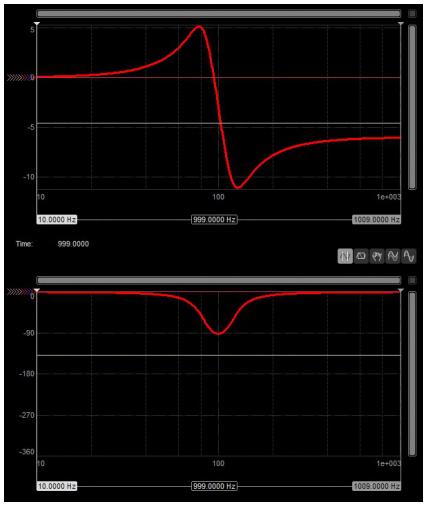


Fig. 67: Two-Load Filter Güte=3

Two-load filters with increased quality can be used to compensate for a pole zero. However, this is to be used with caution as the filter may be counter-productive in the case of low shift in the resonance.

11.8 Optimisation roles

The locus of the Ziegler-Nichols diagram must not touch the two circles
Mt Complementary Sensitivity Circle and Ms Sensitivity Circle .

The gain of the filters can be positive or negative. Speed Filter

The speed filters have a direct impact on the actual speed. If one of the speed filters is being used, the flag Ctrl.Test.SpeedFilter enabled must be switched on during sweeping

11.8.1 Observer Filter

Higher-frequency resonance and resonance caused by “elastic encoder attachment” can be largely eliminated with the help of a speed observer.

The observer filer comprises, on the one hand, a low-pass filter with cut-off frequency F_g and, on the other hand, the actual observer with the control weight kP_Iq. The attenuation of the low-pass filter is -40dB per decade.

The control weight kP_Iq can also be set to 0. This only leaves the low-pass filter.

F_g — cut-off frequency; in the case of low-vibration mechanics the cut-off frequency can be set to 600 Hz. In the case of mechanics with tolerance and belts, the cut-off frequency may need to be reduced to 200 Hz.

Too low values for the cut-off frequency negatively affect the bandwidth of the system and may lead to deterioration in the quality of control.

kP_Iq — P component for observer filters

The P component for the weighting of the observer must be determined empirically. To do this, start with a very small value: 0.0001.

Continually increase the value until the phase reserve increases. The optimum has been reached when there is still sufficient amplitude reserve when the phase reserve has been maximised.

The speed control bandwidth is increased without reducing the phase reserve.

Characteristics of the low-pass filter

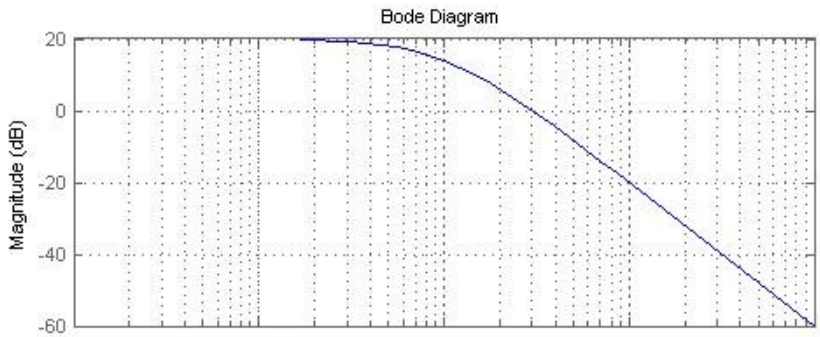


Fig. 68: Low-pass filter 2nd order

Due to its better characteristics, the observer filter is preferable to the average filter.

11.8.2 Average Filter

kT_Speed

The D component of the PID controller generates noise, which can be reduced slightly with kT_Speed. If there is a large mass, kT_Speed can be set at up to approx. 1/10 of the duration Tk. Always as low as possible and only as high as necessary.

In the case of two overlaid oscillation times:
If two oscillation times are displayed (e.g. 6ms belt oscillation time, 100ms mass vibration time), aid can be provided in the form of **kd** (acceleration error or D component of speed controller).

To do this, the kd is increased in very small steps. A compromise must be found between the smallest possible vibration behaviour and the lowest possible noise build-up in the drive.

If the kd is used, the resolver filter kT_Speed must be set to 0.1.

Characteristics of the average filter

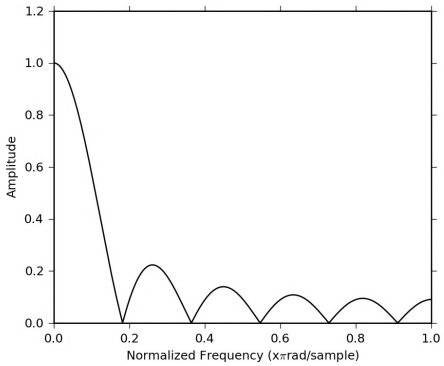


Fig. 69: Average Filter

11.9 Gantries

The following points must be considered when sweeping double Y gantries.

- Select such currents for the sweeping that the motors and mechanics are not overloaded.
- The axes must be decoupled for the recording of the bode sweep
- Place the X and Y axes in the centre and sweep
- Whilst one axis, Y1, is being swept, the other axis, Y2, must be activated and held in simulation mode.
- After the first sweep and the optimisation of the PID parameters/filters, the found parameters must be loaded into both axes, Y1,2. Then activate Y2 axis in simulation mode and hold with the optimised PID parameters/filters.
  Repeat the sweep for axis Y1.
- Several sweeps should be recorded in order to find optimum parameters that are valid for the entire working area. To do this, determine a matrix with points with spacing of 10cm. Travel to all points with the gantry and create a sweep with **both** axes, X1 and X2.

12 Commissioning a stepper motor without feedback

2-phase stepper motors are controlled with a separate current controller for each phase:
Phase U = Iq (active current controller)
Phase V = Id (idle current controller)

There are therefore 2 current controllers, phase U = Iq and phase V = Id, that must follow the sine/cosine currents. For this reason, you can only use a P controller; the kI is not used and remains at 0.

There is, of course, no position controller as there is no feedback.

The kP of PosCtrl is used to set the basic current:
Istandby = Motor.In * PosCtrl.standby.kP
Iforward = Motor.In * PosCtrl.forward.kP
Ibackward = Motor.In * PosCtrl.backward.kP

When kP = 1, the defined I_nom current of the motor is applied.

The lead values phvSpeed and phvAcc function as normal

The fieldbus feedback is made of FieldAxisPos, i.e. with 4096Inc/turn

Important

- The stepper motors have a very high Ke (24V motors approx. 30-60V; best to measure with, for example, a battery drilling machine)
- When field-controlled, they therefore only run up to 400-600 rpm. In addition, field weakening (increasing -Id) would have to be introduced, but has not yet been implemented.
- Without FB control, they run higher, but from this speed of rotation the current coincides, meaning that at 2000 rpm only 200mA flows instead of 2A, and the torque therefore decreases.
- The current, kP, must be at approx. 2-8
- Rs, Ls, Ke are still entered for 3Ph_pp -> convert
- When there is no load, resonance is to be expected; the motor is only standing still
- With a load, a smaller current is better, depending on the speed of rotation (Pos_kP=0.8, although it is normally not reached anyway)
- Encoders with, for example, only 4x400=1600Inc/T are not sufficient for field control, as only 1600/50/4 = 8 Inc remain for a 90degree motor field.
- run under 4096Inc/T sicher _woFB, encoder to SAM, small PID_P in SAM

13 Firmware update, parameter update

Alongside the firmware, the INFO-ACS controller also requires two configuration files:

Controller firmware	System.s
Motor data	File.cpf

The motor data contains the PID parameter sets, physical constants such as ohmic resistance, inductance of the motor, etc.

Firmware and parameter updates can be burned to the flash prom of the controller either from the parameterising programme INIX-Motion or using the console programme ACSUpdate.exe .

Firmware and motor data can also be carried out on remote targets via a network.

13.1.1 Updates to parameters and software

Prerequisites:

- INCO server must be in operation
- 24V supply for SAC controller
- Target must be registered
- Active link if not communicated with serial target
- ACSUpdate.exe

The programme ACSUpdate.exe is located in the folder `"..\acs\bin"`.ACSUpdate.exe is a DOS programme. To start it, open a DOS box. Structure of the command:

13.1.2 Burning firmware or motor parameters to the flash prom

The files (motor parameters and firmware) are transmitted to the programme via command lines. ACSUpdate.exe orients itself using the file endings; the sequence is not important.
Before burning, ACS-Update checks the version numbers of the existing software and the software to be loaded. With the extension `[-a]` (always), the version to be loaded is always burned; without the extension, it is only burned if it is a newer version.

```
c:\ACSUpdate TargetName [*.s][-a] [*.cpf][-a] [*.chf][-a]
```

Examples

Burn the controller software `"system.s"` and the motor parameters `"motor.cpf"` to the flash PROM as long as it is a **newer** version than the one in the flash prom. The target is axis 0, registered on a power PC master.

```
C:\ACSUpdate PPC\Axis0 system.s motor.cpf
```

Always burn the controller firmware `"system.s"` (only) to the flash PROM, even if it is an older version. The target is an ACS controller, which is addressed via the serial interface.

```
C:\ACSUpdate INFO-ACS a:\acs\update\system.s -a
```

13.1.3 Saving motor parameters in a file

Motor parameters are saved with the extension [-s] (save) in a file with the specified ending `[*.cpf]`:

c:\ACSUpdate TargetName [*.cpf][-s]

`[*.cpf]` = motor parameters

Examples
Save the motor parameters in a file. The target is an ACS controller at a serial interface and is located within a network.

C:\ACSUpdate Remote_ACS motor.cpf -s

13.1.4 Copying parameters from the RAM into the flash prom

ACSUpdate with the extension `[-h]` burns the current motor parameters from the controller's RAM to the flash prom:

c:\ACSUpdate TargetName [-p]

`[-p]` = motor parameters

Example
Burn the motor parameters from the ACS controller on a stand-alone master to the flash PROM.

C:\ACSUpdate INFO-SAM/AXIS0 -p

13.1.5 Information

The extension `[-i]` compares the firmware version or controller parameter version in the flash PROM of the controller with the version to be loaded and shows whether an update is required. The update is not carried out!

c:\ACSUpdate TargetName [*.s][-i] [*.cpf][-i]

13.1.6 Automating flash PROM updates

In order to be able to burn flash PROM updates in a logical manner for a machine or system with several motors, including those distributed within a network, a configuration file can be transmitted to the programme ACSUpdate.exe. The configuration file contains all of the commands needed to burn several flash PROMs. The file ending must be `[*.CFG]`!
The extension `[-a]` can also be transferred to the configuration file.

```
C:\ACSUpdate [*.cfg] [-a]
```

Content of the configuration file

```
; local PowerPC master, axes 0,1,2,3
PPC/AXIS0 system.s axis0.cpf
PPC/AXIS1 system.s axis1.cpf
PPC/AXIS2 system.s axis2.cpf
PPC/AXIS3 system.s axis3.cpf

; PPC master in network, axes 0,1
Remote_PPC/AXIS0 system.s axis4.cpf
Remote_PPC/AXIS1 system.s axis5.cpf

; ACS controller at serial interface in network
Remote_ACS system.s axis6.cpf
```

Example
Implement the above configuration file:

```
C:\ACSUpdate ACSConfig.cfg
```

13.1.7 Version and assistance

`C:\ACSUpdate -V`	displays the current version of the controller software.
`C:\ACSUpdate -?`	displays a brief help screen.

13.1.8 Updates with a laptop

Prerequisites:

- INCO server must be in operation
- 24V supply for INFO-ACSr
- Serial target must be registered
- Important Notes
- ACS-Show or ACSUpdate.exe

To burn flash PROM updates to the controller with a laptop, please see sections:

```
"3.3  Laptop installation"
"4.2  Starting ACS-Show"
"4.8.1      Loading software and parameters"
```

or use the console programme “ACSUpdate.exe”. To carry out an update, all the controller INFO-ACSr needs is a 24V supply and the serial connection to the laptop. An active link (i.e. `"Trans.exe"` implemented) is not required for a serial connection.

Important notes

The following sequence must be followed when connecting a laptop computer to the controller’s serial interface:

1. Remove the mains power supply from the laptop, so that it is only being supplied with power by the battery.
2. Connect the INFO-ACSr and the laptop using the corresponding serial cable.
3. Reconnect the mains power supply.

Grund: Due to the galvanic isolation of the transformer, the laptop supply is increased to a potential of 110V (provided the laptop is supplied by a 230V network). As in the case of standard D-SUB plugs it cannot be guaranteed that the shielding will come into contact before the signal cables, there is a risk that the potential equalisation will take place via the signal ground cable. This will lead to destruction of the relevant SIO channel.

13.2 Emergency system

If an error occurs when burning the motor parameters and the flash prom is destroyed, the drive can still be started within the emergency system.

In order to be able to start the drive in the emergency system, a short-circuit plug must be connected to the serial interface (front plate). Flash PROM burning is supported in the emergency system.

Connections:

Signals Pin	SAC2/3 D-SUB 9-pol.	SAC3x3 RJ-45
RxD, TxD	2, 3	1, 2
DSR, DTR	6, 4	3, 4

Once the controller has been started, the short-circuit plug can be removed and the serial cable plugged back into the PC.

14 Trouble Shooting

14.1 INFO-link problems

Link problems must be solved first. The error counter must not count upwards. See light quantities for the measuring unit “INFO-Mess”.

Link problems must be solved first. The error counter must not count upwards.
See light quantities for the measuring unit “INFO-Mess”.

14.2 Problems with analogue encoders: SinCos, Resolver

If you have problems with analogue feedback systems, you will find more information here:

Section:
3.5 Resolver
3.6 SinCos
3.7 Checking the sine cosine / resolver level

14.3 Soiling

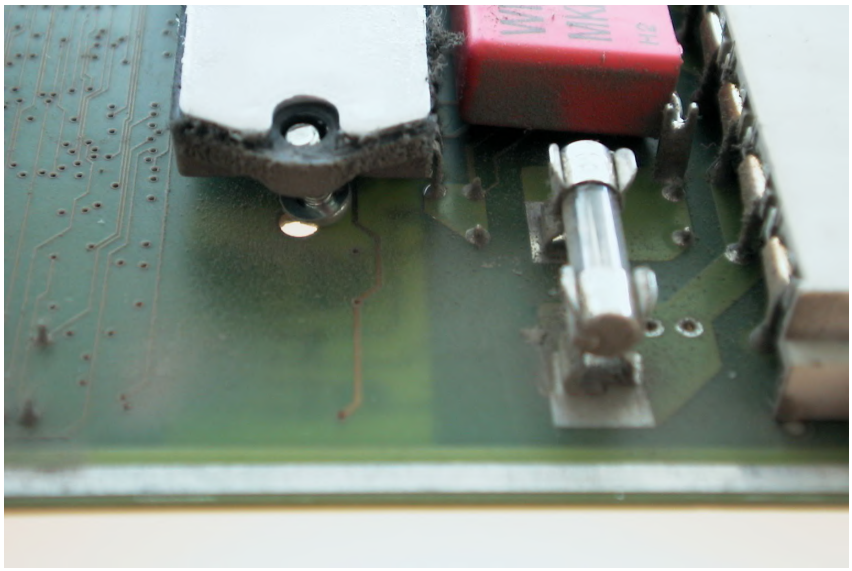


Fig. 70: Soiling

Fans must be equipped with a dust filter. Dirt and moisture can lead to short-circuiting in the card!

14.4 Supply

14.4.1 Intermediate circuit voltage

Fig. 71: Intermediate circuit

The current (yellow) decreases even though the position error is increasing. The difference between the target (green) and actual (blue) speed of rotation also increases.

Error: The intermediate circuit voltage is too small.

Causes

- Too many controllers on one mains supply
- Motor overloaded
- Supply voltage (230VAC, 400VAC) too small

14.4.2 Voltage dips

The phase recognition may be engaged if an individual phase dips too considerably. In order to remedy this, the flag No_PhaseFailure can be set to 1. This means that phase recognition is completely switched off.

Path in Inco-Tree for the resolver: Ctrl.MotorConfig.Power.Supply.Flag

14.4.3 Supply to MAX board

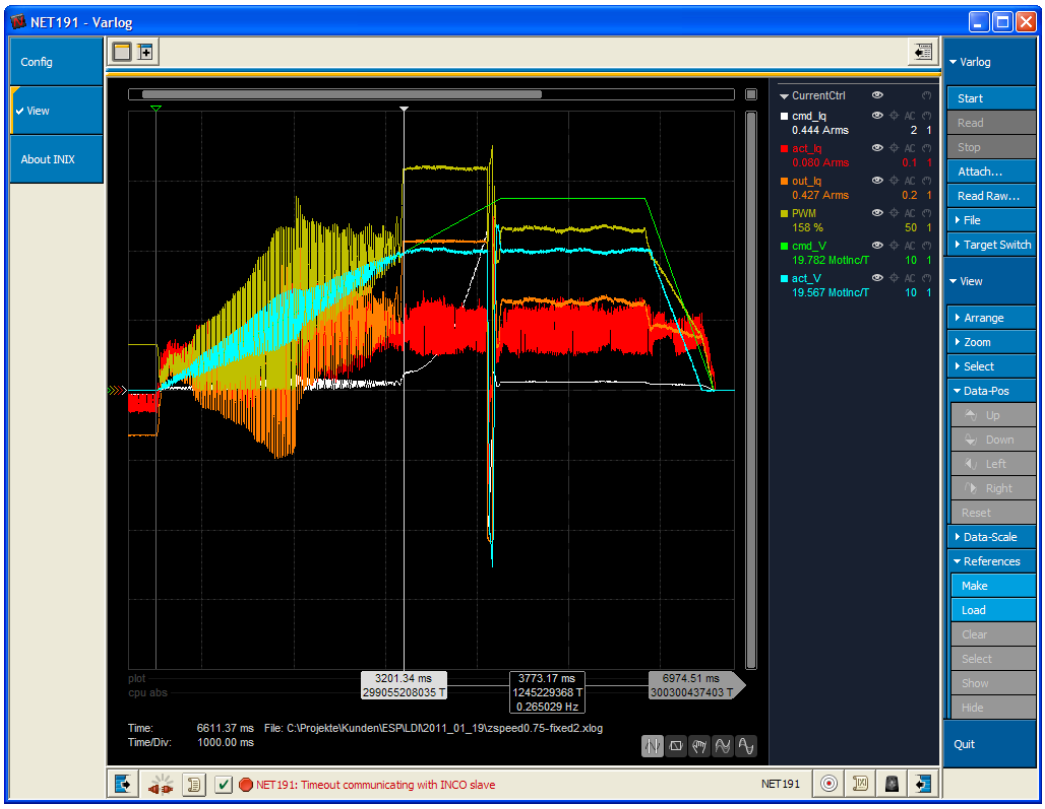


Fig. 72: Supply to MAX board

The supply to the motor (+V_MOT) of the MAX2/4 board is too low.
The minimum supply voltage is +15V; in the example above, the supply is +12V.

14.5 Last

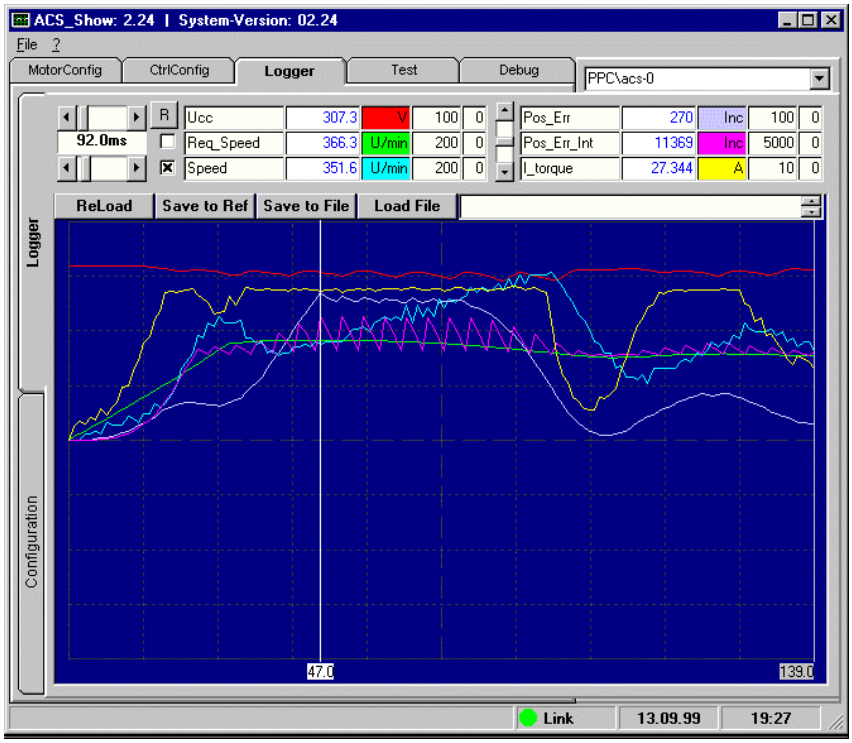


Fig. 73: Last

The maximum current has been reached. The entire path is driven at maximum current.

Error: The controller is at full capacity.

In the case of the cursor position, the positioning error is approx. 270 increments. The maximum permissible amount is 250 error increments. The actual speed deviates considerably from the target speed. The motor can no longer be controlled in this state!
The controller software limits the positioning error to 250 increments.

14.6 PID-Parameter

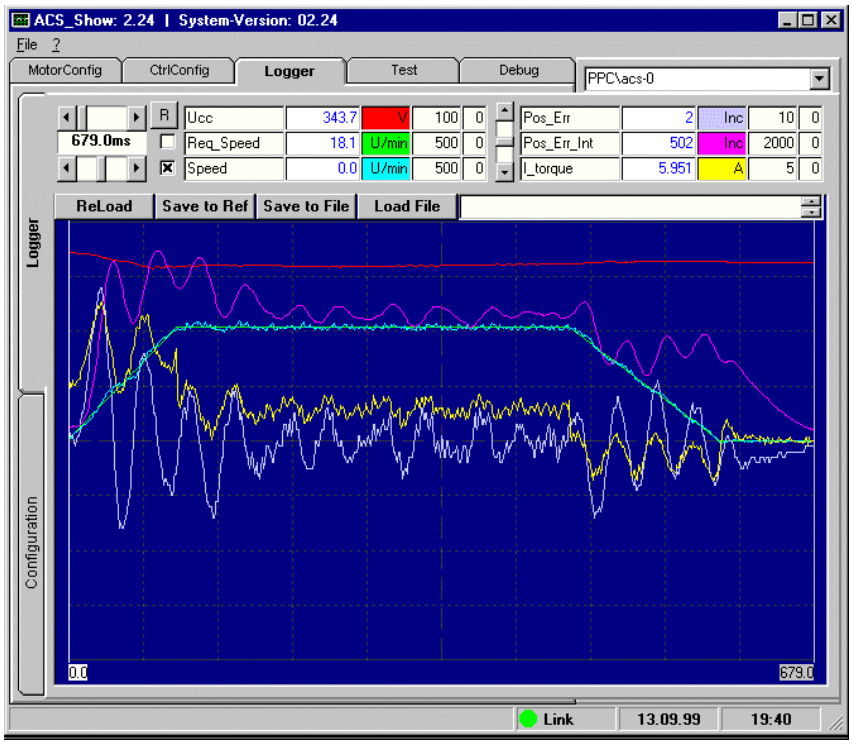


Fig. 74: PID-Parameter

The PID parameters for “forwards” are not optimally set.

14.7 Disruptions

Disruptions on the resolver cable. ±3 increments positioning errors cannot be generated by an active current (I_Torque) of less than 100mA. Please note the wiring notes in section 2.3 “Wiring” and in the wiring guidelines.

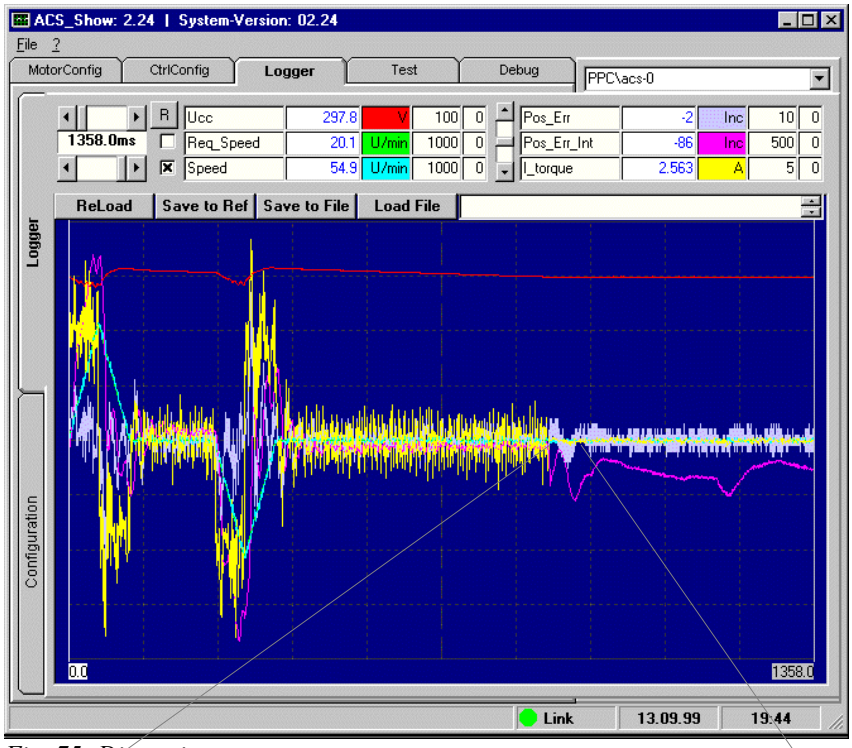


Fig. 75: Disruptions

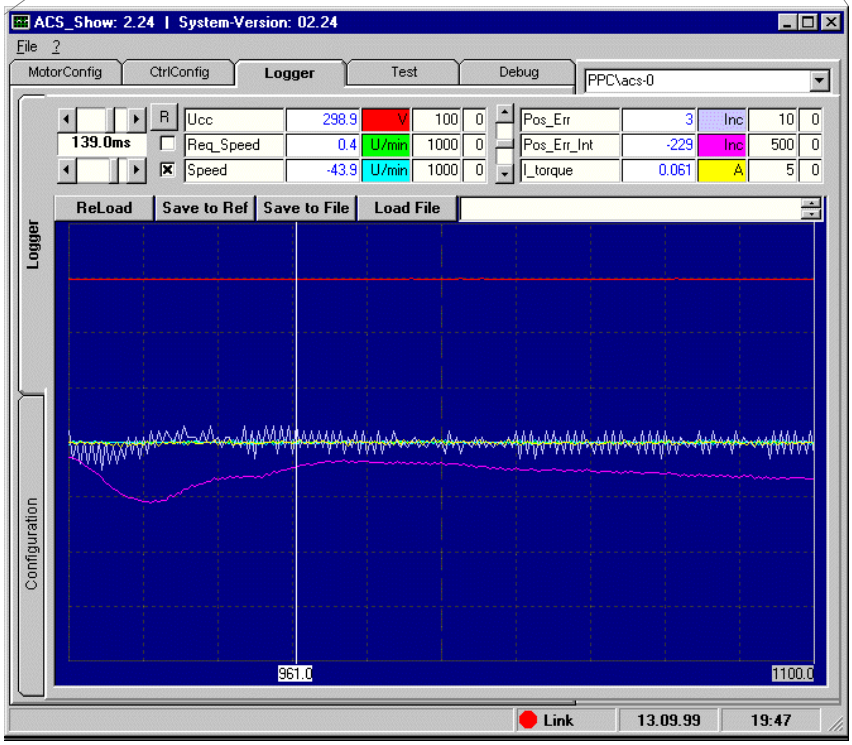


Fig. 76: Disruptions

14.8 Lead values

The lead value `phvSpeed` has been selected far too high. The actual speed is 40 rpm greater than the target speed. The controller is no longer able to compensate for the large phv. The integrator `Pos_Err_Int` (purple) and the `Pos_Err` (grey) are at their maximum.

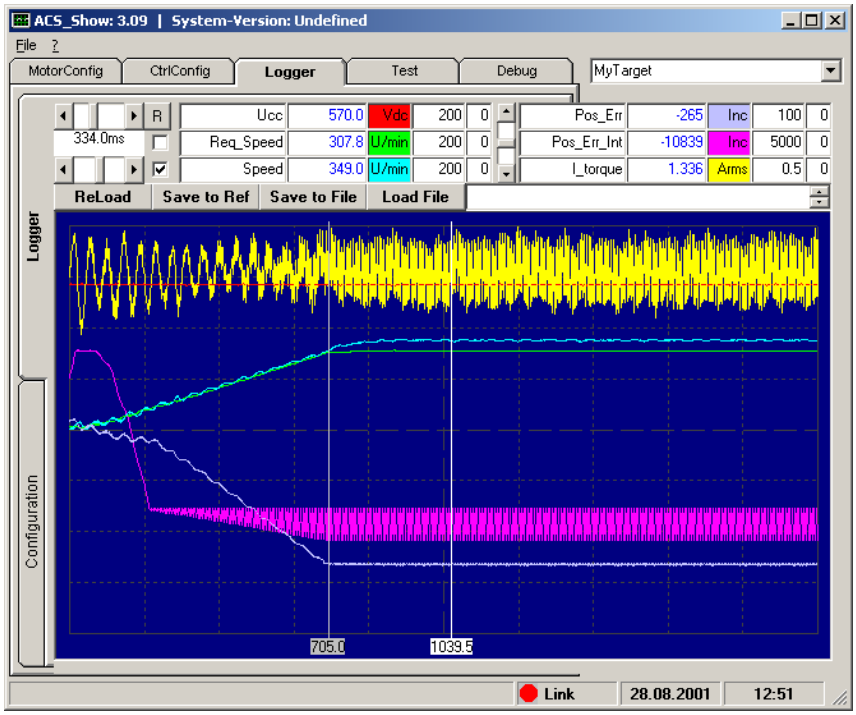


Fig. 77: Lead values

14.9 Standardisation errors

In the INFO-link configuration, the speed of rotation at 10V must be given. In the servo controller, you must also specify the speed of rotation at 10V. If this standardisation is not correct, the control behaviour will not be as it should. If there is too great a difference in the standardisation, it will not be possible for the axis to move, i.e. it will go into following error or overcurrent.

Standardisation errors can be recognised by means of the curve “target speed” or "Req_Speed" (green curve). If the fieldbus master needs to adjust the target speed during travel, the standardisation is not correct.

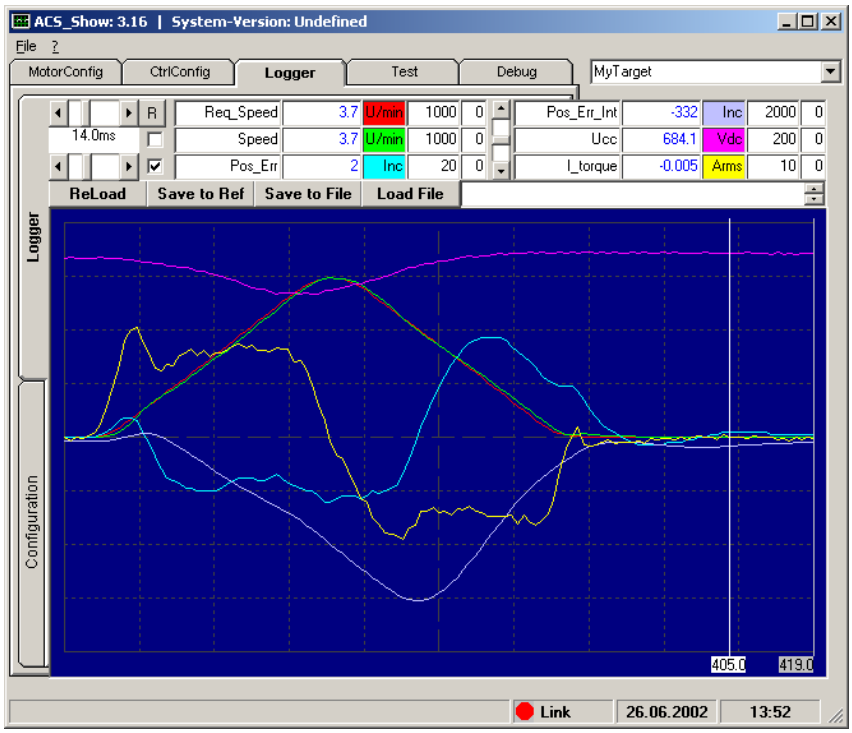


Fig. 78: Standardisation errors

14.10 Incorrect Ke

The value for the Ke is set too high. When braking, the motor therefore receives too little voltage and cannot brake correctly.

See section 7.12 “Fine-tuning of Ke, Rs and Ls”

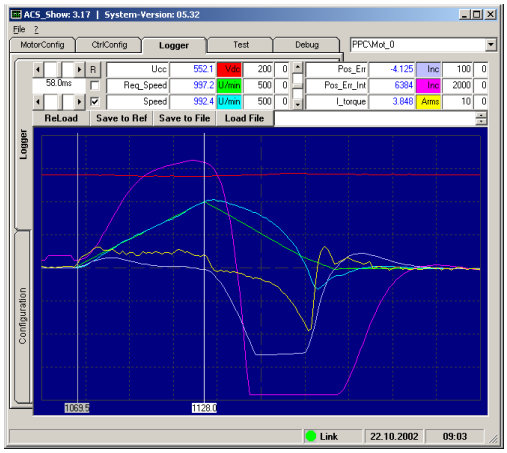


Fig. 79: Incorrect Ke; recording with speed waves

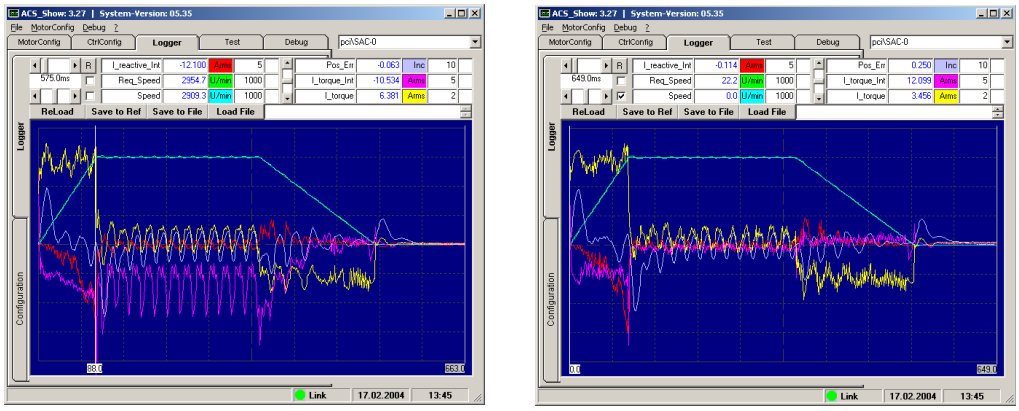


Fig. 80: Incorrect Ke; recording with active current-idle current-integral Image on left: incorrect Ke, image on right: correct Ke

14.11 Incorrect resolver offset

The resolver offset is set incorrectly. This manifests itself in the fact that the axis cannot travel at full speed. If the resolver offset is incorrect, the idle voltage is calculated incorrectly and the controller goes into error with “Imax received”.

See section 10.16.3 “Adjusting the resolver offset”

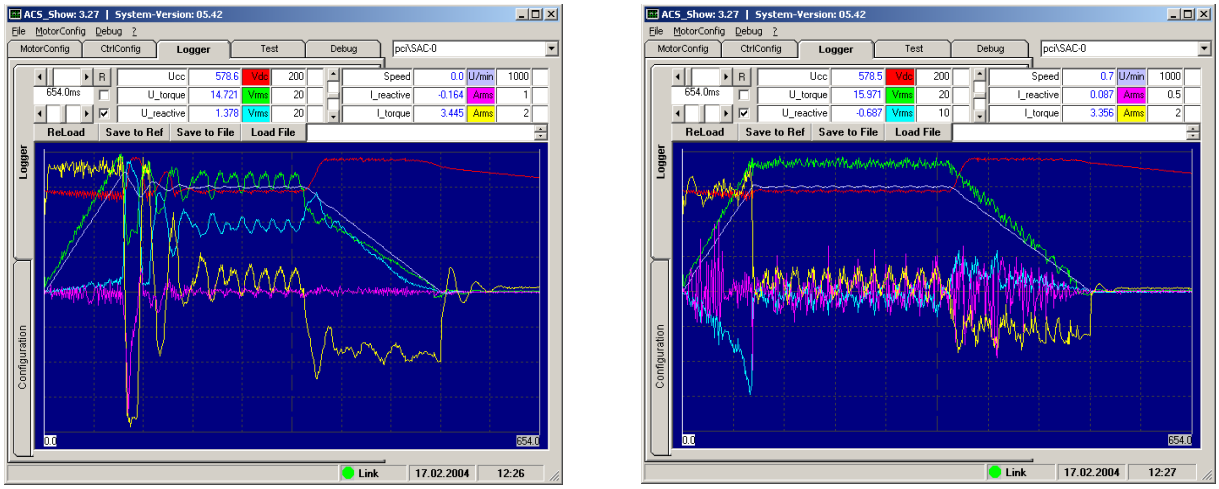


Fig. 81: Incorrect resolver offset; recording with voltage waves Image on left: incorrect, image on right: correct resolver offset

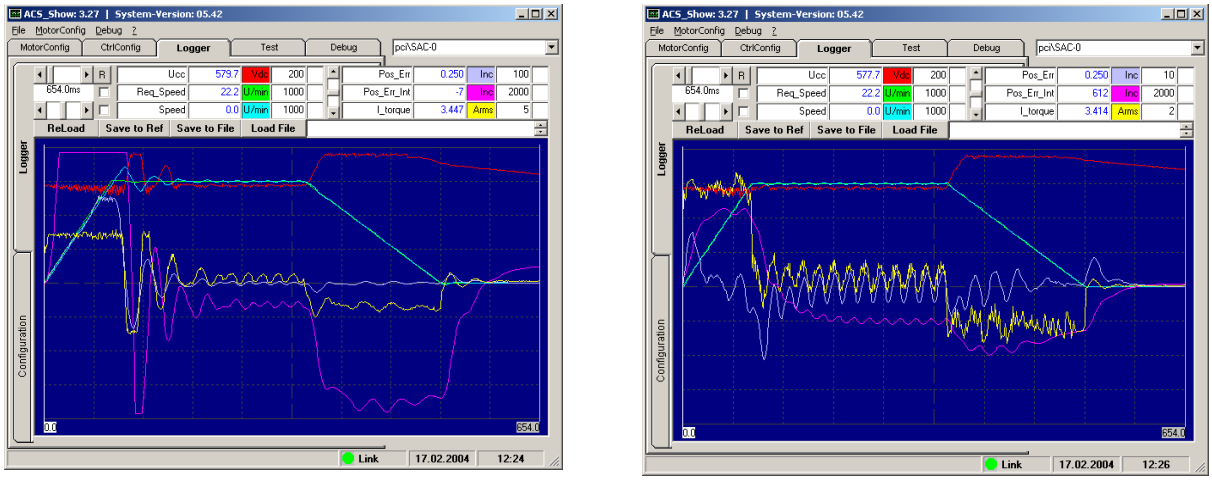


Fig. 82: Incorrect resolver offset; recording with speed waves Image on left: incorrect, image on right: correct resolver offset

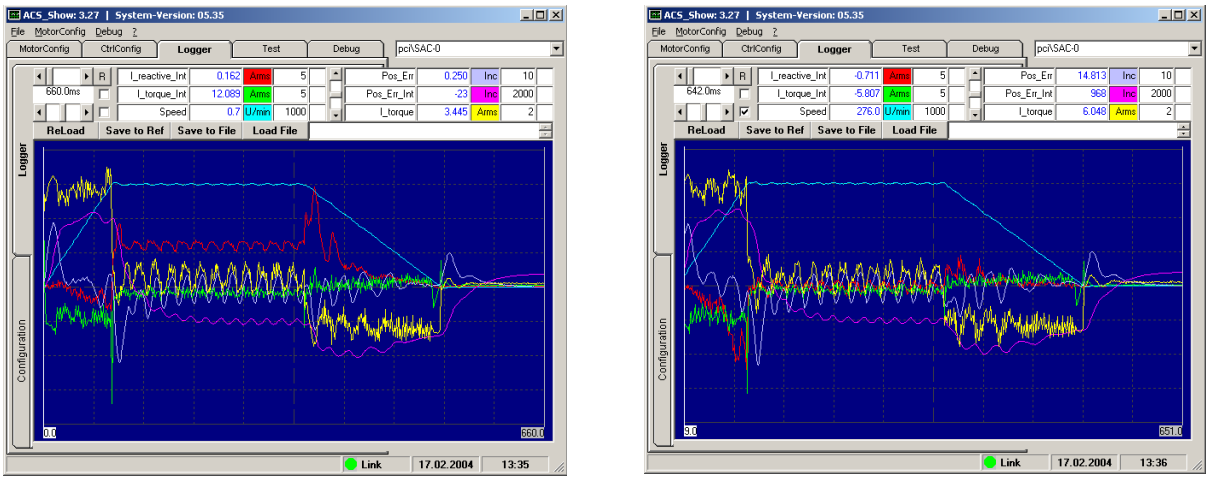


Fig. 83: Incorrect resolver offset; recording with speed waves Image on left: incorrect, image on right: correct Ke

Fig. 84: Incorrect resolver offset, speed waves

15 Further documentation

Drive-Inbetriebnahme-Manual.pdf
Indel-Safety-Manual.pdf
Hardware-Manual-Motion-Boards.pdf
Hardware-Manual-SAC3.pdf
Verdrahtungsrichtlinie.pdf
Aufbaurichtlinie.pdf

16 List of figures

17 Document status

Disclaimer

No guarantee is made for the correctness or completeness of the information provided. Subject to technical changes.

File-History

1.20	29.04.2011	Changes to the motor configuration file for GinLink axes, block commutation for Maxon motors Formula for KTY temperature sensors, Ext_En flag short-circuit
1.21	09.05.2011	Block commutation for Maxon motors expanded, I2t expanded
1.22	10.05.2011	Actual hardware values
1.23	20.05.2011	i2t-Regelung max Motor-Temp.
1.24	21.07.2011	Switch-off sequence with STO and braking ramp
1.25	10.09.2011	New images
1.26	26.09.2011	Procedure for STO
1.27	27.09.2011	Various small adaptations
1.28	11.10.2011	Switching of PID parameters for position controller
1.29	27.04.2012	Burning system software with ZIP file (GinLink), Document status section added Standard assignment of feedback channels on GinLink
1.30	27.04.2012	Configuration of incremental encoder at SinCos interface Beschreibung dIq_dt_corr Flag
1.31	27.04.2012	DeadTime configuration in IMD added, IBN stepper motor expanded
1.32	20.06.2012	Enable bits for GinLink declared
1.33	21.09.2012	Section 10.11.2.1, Auto-commutation with Hiperface, added
1.34	17.04.2013	General expansion of section 10. Now with cross-references to more detailed information in other sections. Can now be used as step-by-step instructions. Description of the asym_acc flag. Section 3.15 expanded. Controller frequency can now be configured in the motor configuration file. Section 5.2 expanded with a description of the Motor.dt2 file. New section 1.2:Overview of the document Description of I2t expanded, now split into the parameters I2t_up_run and I2t_up_halt. Section 11.4 expanded. Phase gain in the lower frequency range by means of a low-pass with pos. Gain
1.35	22.04.2013	Div. small details adjusted. Div. spelling errors corrected.
1.36	26.04.2013	Section 3.14.1Motor configuration adapted Section 3.16.1 Description of the Asym_acc flag corrected
1.36	21.11.2013	Translation Commissioning Manual from German to English