

AN032: TMC4361A closed-loop motor control for stepper motor drivers

Document Revision V1.0 · 2020-Mar-10



This application note explains functional details of closed-loop motor control of stepper motor (gate) drivers in combination with the motion control IC TMC4361A. It serves as technical background information. Please refer to the specific product documentation for specific parameter information and configuration instructions.

Contents

1	CLOSED-LOOP MOTOR CONTROL	2
1.1	CLOSED-LOOP MOTOR CONTROL IS NOT FIELD-ORIENTED CONTROL	2
1.2	CLOSED-LOOP MOTOR CONTROL PREVENTS STEP LOSS	3
1.3	CLOSED-LOOP MOTOR CONTROL SAVES ENERGY	3
1.4	HOW TO SETUP CLOSED-LOOP MOTOR CONTROL	3
2	CONNECTING TMC4361A WITH μ C, MOTOR (GATE) DRIVER AND ENCODER	4
2.1	SIGNAL DESCRIPTION OF THE REQUIRED PINS	4
2.2	CONNECTING THE PARTICULAR PINS FOR A MINIMALIST SETUP	5
2.3	GENERAL TMC4361A REGISTER SETUP HINTS	5
2.4	SAMPLE SETUPS	6
2.5	REGISTER ACCESS FROM μ C TO TMC4361A	8
2.6	SETUP SPI OUTPUT CONFIGURATION FOR TMC MOTOR (GATE) DRIVERS	9
2.7	SETUP INTR OUTPUT CONFIGURATION	9
2.8	SENDING COVER DATAGRAMS TO SETUP TMC26x/389	10
2.9	EXAMPLE CONFIGURATIONS OF APPROPRIATE CURRENT SETTINGS FOR TMC26x	11
2.10	EXAMPLE CONFIGURATION FOR TMC21x0	11
2.11	EXAMPLE CONFIGURATION OF APPROPRIATE CURRENT SETTINGS FOR TMC21x0	12
3	BASIC ENCODER SETUP FOR CLOSED-LOOP CONTROL	13
3.1	GENERAL ENCODER SELECTION	13
3.2	ABSOLUTE ENCODER CONFIGURATION	13
3.3	ENCODER RESOLUTION	14
3.4	ENCODER COMPENSATION	15
4	CLOSED-LOOP SETUP	17
4.1	LIMITING THE MAXIMUM CORRECTION VELOCITY	18
4.2	CLOSED-LOOP OPERATION DURING RAMP POSITIONING MODE	19
4.3	CLOSED-LOOP VELOCITY MODE	19
4.4	CLOSED-LOOP CALIBRATION AND ACTIVATION	20
5	CURRENT SCALING DURING CLOSED-LOOP MOTOR CONTROL	21
5.1	EXAMPLE FOR A COMBINED USAGE OF CLOSED-LOOP ANGLE TRACKING AND CURRENT SCALING	22
6	CONSIDERING THE BACK-EMF	24
6.1	VELOCITY SETUP	25
6.2	VELOCITY CALCULATION	27
7	EXAMPLES	29
8	REVISION HISTORY	39

1 Closed-Loop Motor Control

TMC4361A provides closed-loop motor control capabilities for stepper motor (gate) drivers. Typically, current values or step impulses from TMC4361A for the motor drivers are based only on internal calculations. With the closed-loop unit of the TMC4361A the output currents resp. Step/Dir outputs of the internal step generator will be modified directly in dependence of feedback data. This feedback data can be obtained from different encoder types like incremental ABN encoders or absolute SSI or SPI encoders.

1.1 Closed-loop motor control is not field-oriented Control

The classical field-oriented control typically uses a cascade of three control loops. The inner loop controls the motor current. One level beyond velocity is controlled until final position is reached. The current control loop always assigns a current, which has a commutation angle of 90° , to gain maximum torque. Feedback is obtained by encoders or by the measurement of phase currents and voltages; with additional help of mathematical models.

The 2-phase closed-loop control of TMC4361A follows a different approach than PID control cascades, which consider stepper motor driver characteristics. The ramp generator - *that assigns target and velocity* - is independent of position control (commutation angle control), which is also independent of current control. The closed-loop motor control scheme is depicted in the following picture.

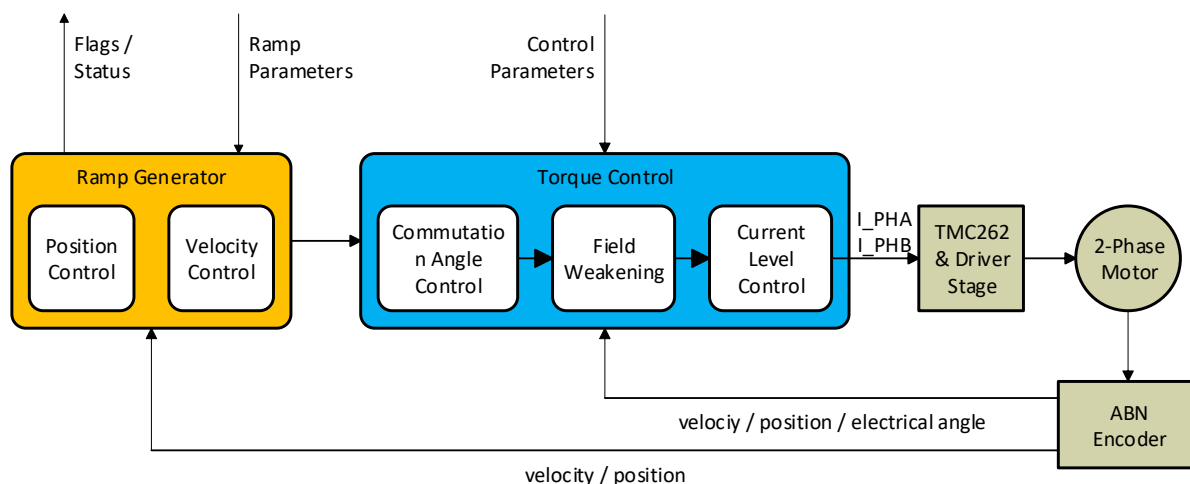


Figure 1.1 Structure of the closed-loop Motor Control

AREAS OF SPECIAL CONCERN

Use of PID Regulation Feature

The encoder should be mounted directly on the motor axis for closed-loop operation to prevent unwanted coupling effects.

Also, if the encoder is not mounted directly on the motor, the PID regulation feature must be used. For example, PID control will avoid slipping of belt drives, because they cannot be handled correctly with the closed-loop motor control.



1.2 Closed-loop motor control prevents step loss

Phase currents are assigned directly to the motor drivers, typical for stepper motor drivers. This results in current vector which should be followed by the rotor. The rotor position will be directly sampled by encoder feedback. The closed-loop motor control now monitors the resulting load angle.

The direction of the current vector tracks the rotor position in case the load angle exceeds a certain limit. The result is a load angle, which never exceeds the given limit. As a result no step loss will occur. Thus, the current vector follows an overpowered load until the load is reduced. After overcoming the overload, the current control assigns the required position or velocity.

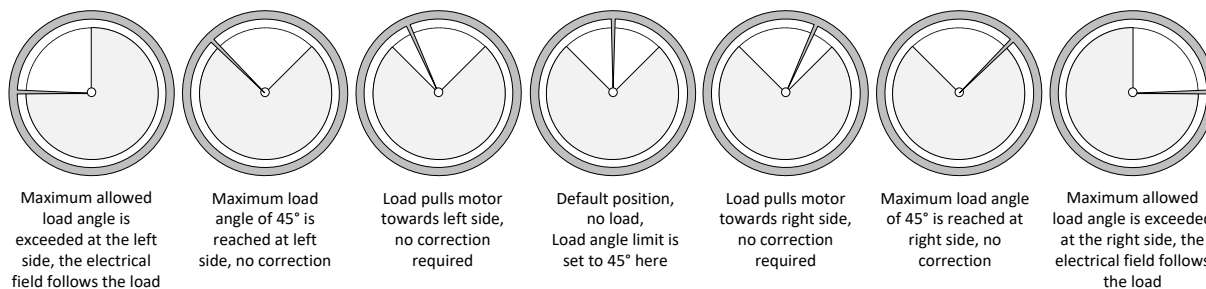


Figure 1.2 Load Angle Control and Current Synchronization

NOTE:

→ Typically, the load angle limit will be 90° which is the maximum torque which is one full step, but it is also possible to assign any limit between 0° and 180°

1.3 Closed-loop motor control saves energy

Besides the load angle tracking, it is also possible to assign different current scale values for different load situations. Thus, lower load situation can be used to save energy. This is a result of the free adaptable current scaling feature which assigns the current in dependence on the actual load. During low load phases, lower current levels will be assigned to the motor driver, whereas the maximum current will be applied if the load limit is reached.

1.4 How to setup closed-loop motor control

In the following application note the complete set of closed-loop features of TMC4361A will be explained. The general flow can be divided in three different stages whereas the latter two are optional and can be used independently on each others. Firstly, the closed-loop behavior have to be prepared and established. This general setup process is mandatory and will be clarified in chapter 3. The two optional refinements of the closed-loop motor control will elucidated in chapter 4 (current scaling capabilities) and chapter 5 (back emf considerations). In the last chapter, further settings for closed-loop operations will be explained.

But before starting with the closed-loop setup, the whole setup must be connected correctly. In the next chapter 2 the possible connections between μC , TMC4361A, the motor, the encoder and the motor (gate) drivers will be introduced.

- i. Information about correct pinning and layout considerations of the different devices can be found in the data sheets of the particular components. Information on correct SPI and encoder communication schemes for TMC4361A are available in corresponding manuals, providing detailed information.

NOTE:

→ There is one exception: If circular motion (see 0) is enabled and current velocity is too high for exact positioning during one revolution, it is possible to change from velocity to positioning mode.



2 Connecting TMC4361A with μ C, motor (gate) driver and encoder

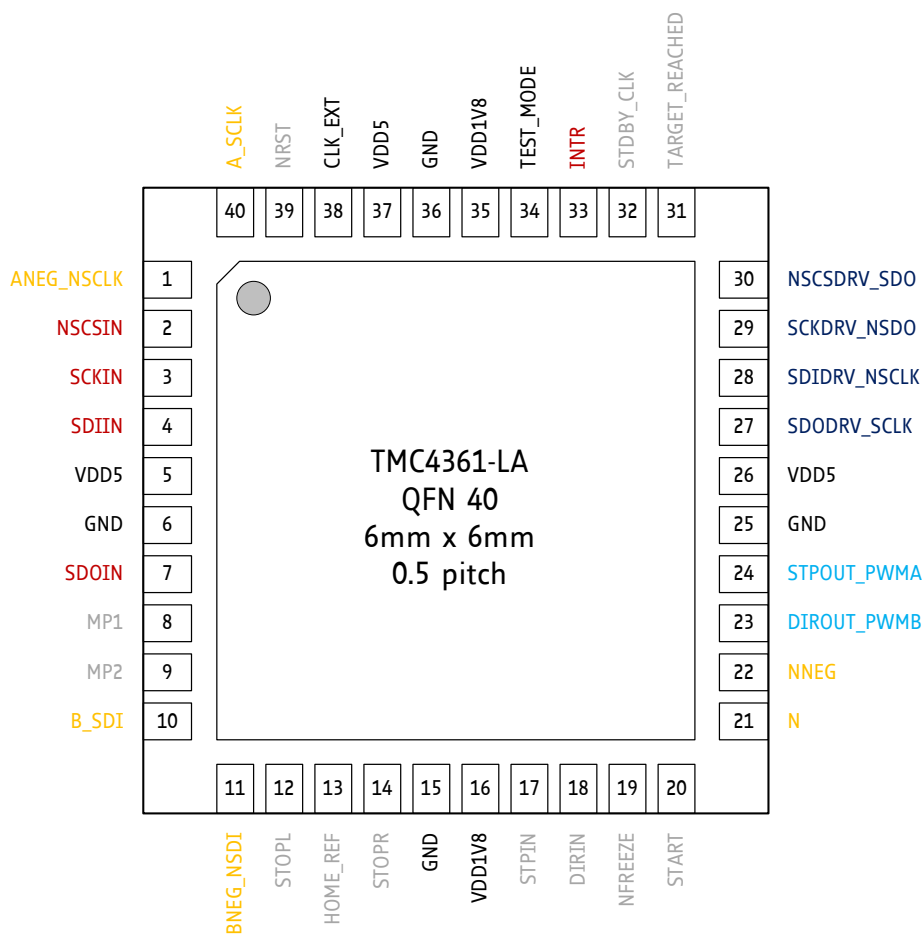


Figure 2.1 Pinning (top view)

2.1 Signal Description of the required Pins

Pins	Number	Function
GND	6, 15, 25, 36	Digital ground pin for IOs and digital circuitry
VDD5	5, 26, 37	Digital power supply for IOs and digital circuitry (3.3V... 5V)
VDD1V8	16, 35	Connection of internal generated core voltage of 1.8V
CLK_EXT	38	External Clock input to provide an clock with the frequency f_{CLK} for all internal operations.
TEST_MODE	34	Test mode input. Tie to low for normal operation.
INTR	33	Interrupt output, programmable PD/PU for wired-and/or
NSCSIN, SCKIN, SDIIN, SDOIN	2, 3, 4, 7	SPI interface to the μ C.
A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NSADI, N, NNEG	40, 1, 10, 11, 21, 22	Encoder interface for incremental ABN encoders and absolute SSI and SPI encoder types.
STPOUT_PWMA, DIROUT_PWMB	24, 23	Step Direction output for motor stepper drivers
NSCSDRV_SDO, SCKDRV_NSDO, SDIDRV_NSCLK, SDODRV_SCLK	30, 29, 28, 27	SPI interface to Trinamic Motor drivers.



2.2 Connecting the particular pins for a minimalist setup

In the following the required pin signals of TMC4361A will be identified to establish a complete closed-loop system. This is a minimalist setup. If you want to use the other pin signals with its features, please refer to the TMC4361A manual.

For a minimal closed-loop motor control setup please do the following:

- Connect the ground pins and the power supply pins properly. Put at every supply line a capacitor of 100nF to ground
- Connect the external clock signal to CLK_EXT with a frequency between 4.2 ... 32 MHz.
- Disable TEST_MODE by connecting it with ground.
- Connect the INTR output pin to the μ C. It is not necessary to evaluate this signal, but it will be very helpful if so. The setups which will be explained in the next chapters will use this pin.
- Establish an SPI connection from the μ C to TMC4361A. Thus, connect NSCSIN, SCKIN, SDIIN and SDOIN with the μ C (please refer to the TMC4361A manual to establish a correct working SPI communication scheme).
- The feedback signals should be connected to the encoder interface pins (A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NSDI, N, NNEG). Here, three types of encoder feedback signals can be evaluated: incremental ABN encoders, absolute SSI and SPI encoders. Please refer to TMC4361A manual to connect encoder feedback signals correctly.
- Connect the StepDir output pins (STPOUT and DIROUT) to any motor driver
- And/Or connect the SPI output signal lines (NSCSDRV_SDO, SCKDRV_NSDO, SDIDRV_NSCLK, SDODRV_SCLK) to the SPI input pins of a particular TMC motor driver

NOTE:

- *The encoder should be mounted directly on the motor axis for closed-loop operation to prevent unwanted coupling effects.*

2.3 General TMC4361A register setup hints

Different setups are possible. Please note that any encoder type can be used with any motor driver. The shown examples in the next section are not determined for the particular encoder type.

The differential signals of incremental ABN and absolute SSI encoder types are optional. If selected, the signals will be differentiated by its digital values. There is no differential amplifier in front of the digital inputs inside the chip. Per default, differential signals are expected. It can be easily switched off by setting the **GENERAL_CONF register 0x00** properly (bit12: *diff_enc_in_disable*).

Besides setting proper configuration for the SPI output interface via **SPIOUT_CONF register 0x04** to provide correct SPI data for the connected motor drivers, it is also necessary to set the correct encoder type in the **GENERAL_CONF register 0x00** (bit 11:10: *serial_enc_in_mode*).

For further information about the GENERAL_CONF and the SPIOUT_CONF register, please refer to the TMC4361A manual.

Further encoder settings will be available with the **ENC_IN_CONF register 0x07** which will be explained in detail in the next chapters.

NOTE:

- *All input encoder signals are digital inputs. That means that the differential pairs are evaluated digitally.*
- *Thus, it may be required to install differential transceivers between TMC4361A and encoder chip*
- *Encoder input signals can be digitally filtered using the **INPUT_FILT_CONF register 0x03**: encoder input sample rate INPUT_FILT_CONF(2:0) and encoder input filter length INPUT_FILT_CONF(6:4). The filter settings can be set freely. Please refer to the TMC4361A manual for further information.*



2.4 Sample Setups

In the following examples, dashed lines defines optional wires.

Example 1: Motor driver TMC2660/260/261 and incremental ABN encoder

The TMC26x can be configured and driven completely via the SPI output interface of the TMC4361A. The configuration values for the five registers of the TMC26x have to be addressed to the cover register 0x6C of TMC4361A. It will sent automatically after writing to this register. Thus, no connection from μC to the TMC motor driver is required because the communication between both devices can be tunneled through TMC4361A.

If SPI output will be only used for configuration the Step/Dir output pins of TMC4361A can be used to provides steps for the motor driver. The index signal N is also optional.

If no step direction signals are used, the SPIOUT_CONF register 0x04 have to be set accordingly that SPI current datagrams will be sent during motion.

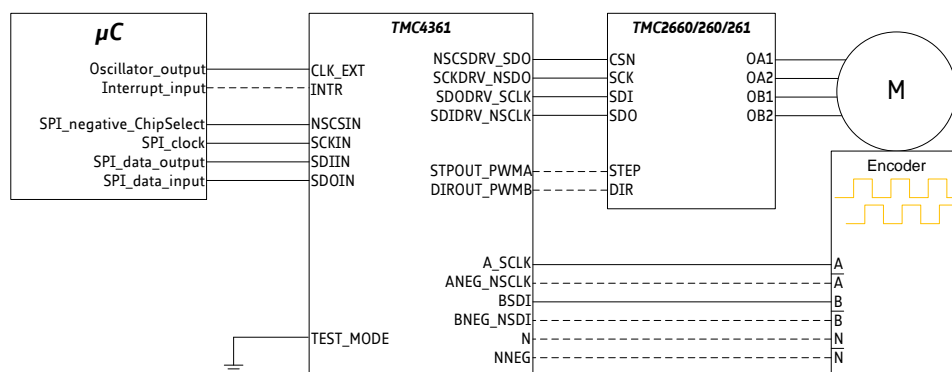


Figure 2.2 Example 1: Connections between μC , TMC4361A, TMC26x, motor M and an incremental encoder which is connected directly to the motor M

Example 2: Motor driver TMC262 and an absolute SSI encoder

This setup depicts the direct configuration of the stepper driver by the μC . Anyhow, the TMC262 can be also configured and driven completely via the SPI output interface of the TMC4361A as mentioned in the example before. Using this setup, the step direction output is necessary to provide step inputs for the motor driver. Also, the negated SSI encoder signal lines are optional.

NOTE:

- TMC4361A supports absolute SSI encoders which provides multiturn as well as singleturn data.
- In the latter case, calculation of multiturn data have to be turned on internally (TMC4361A) for the most closed-loop motor control applications.

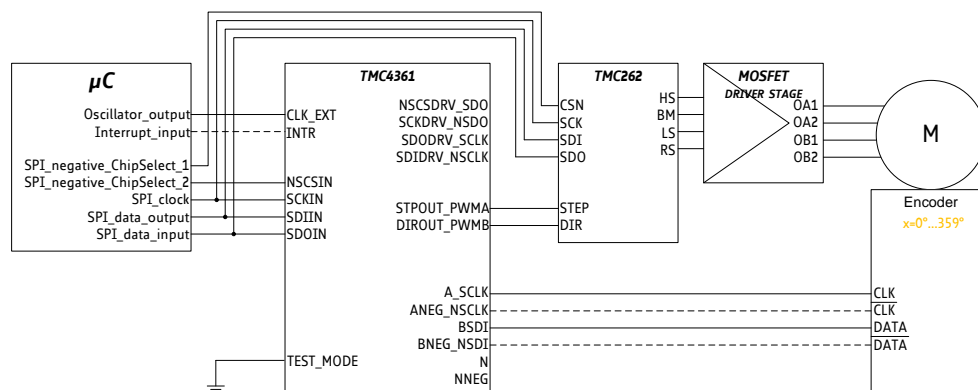


Figure 2.3 Example 2: Connections between μC , TMC4361A, TMC262, motor M and an absolute SSI encoder which is connected directly to the motor M

HINT For three phase stepper motors, TMC389 motor gate drivers can be used. The setup is similar to Example 1 with an optional StepDir connection or Example 2.

©2020 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany

Terms of delivery and rights to technical changes reserved.

Download newest version at www.trinamic.com



Example 3: Motor driver TMC24x/23x and an absolute SPI encoder

This setup covers the connection between TMC4361A and the TMC motor gate drivers TMC248/239/249. Only SPI current datagrams will be sent, no configuration is necessary. If the stepper motor drivers TMC246/236 are connected no MOSFET power stage is required. All other connections can be transferred without changes from this setup.

The SPI encoder is not differential intrinsically. Thus, differential inputs are switched off automatically if SPI encoders are selected in the GENERAL_CONF register 0x00.

NOTE:

- TMC4361A supports absolute SPI encoders which provides multiturn as well as singleturn data.
- In the latter case, calculation of multiturn data have to be turned on internally for the most closed-loop motor control applications.

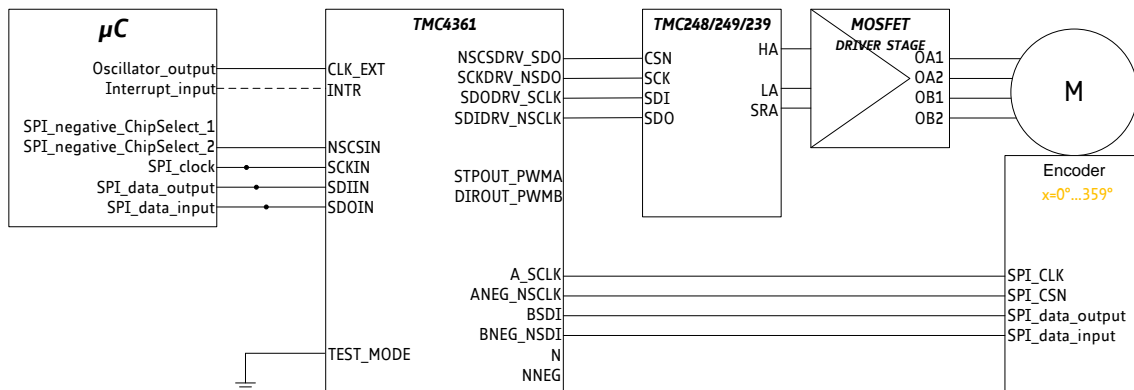


Figure 2.4 Example 3: Connections between μC , TMC4361A, TMC248/239/249, motor M and an absolute SPI encoder which is connected directly to the motor M. If TMC249/239 is used, no MOSFET driver stage is necessary

Example 4: Motor driver TMC2130 and an incremental ABN encoder

The TMC26130 can be configured and driven completely via the SPI output interface of the TMC4361A. The configuration address values for the registers of the TMC2130 have to be addressed to the cover register 0x6D of TMC4361A, whereas the data value have to be addressed to the cover register 0x6C due to the fact that 40 bits overall have to be sent to TMC2130. It will sent automatically after writing to this register 0x6C. Thus, no connection from μC to the TMC motor driver is required because the communication between both devices can be tunneled through TMC4361A.

If SPI output will be only used for configuration the StepDir output pins of TMC4361A can be used to provides steps for the motor driver. The index signal N is also optional.

If no step direction signals are used, the SPIOCONF register 0x04 have to be set accordingly that SPI current datagrams will be sent during motion.

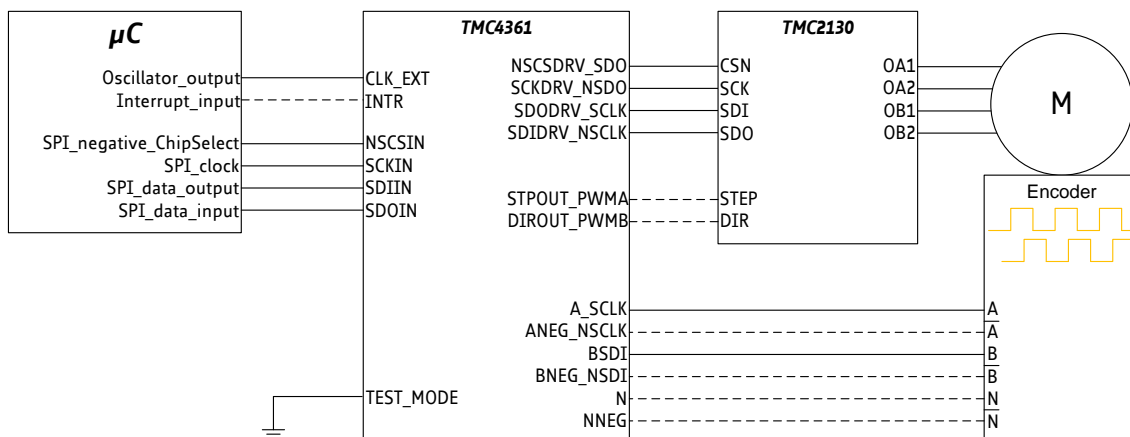


Figure 2.5 Example 4: Connections between μC , TMC4361A, TMC2130, motor M and an incremental ABN encoder which is connected directly to the motor M.



Example 5: Any motor driver and an incremental ABN encoder

Here, TMC4361A is connected with any motor driver. The configuration have to be done via the μC if the driver do not support the TMC SPI configuration scheme. However, if the SPI communication is the same as for TMC drivers, the configuration can be tunneled through TMC4361A. In most cases, step direction output is required. Only if DAC are connected directly to the TMC4361A SPI output current datagrams can be used for motor motion. Please refer to TMC4361A for further information.

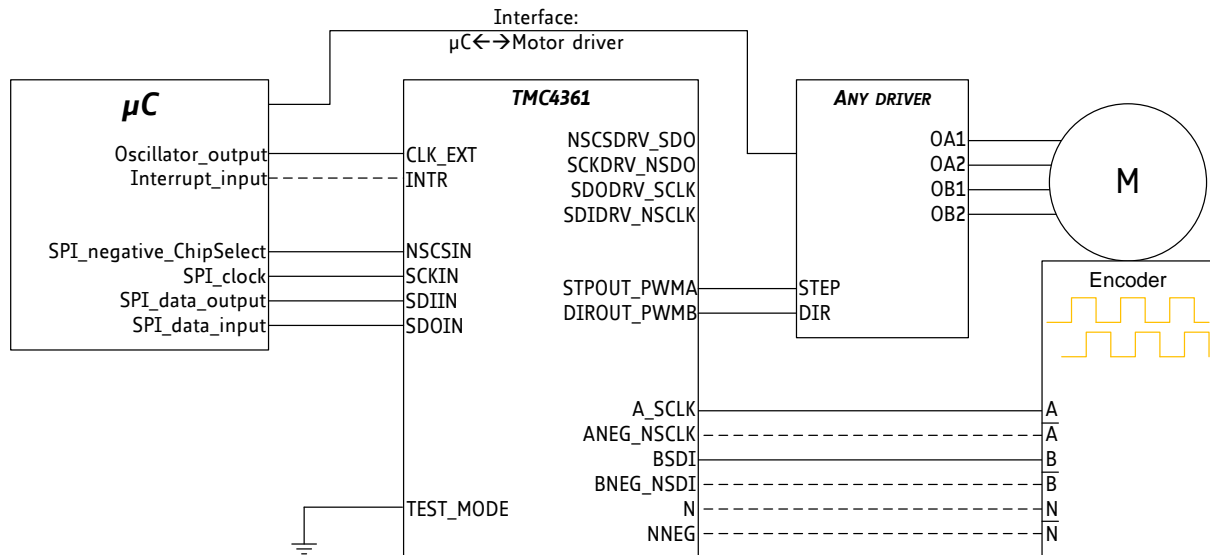


Figure 2.6 Example 5: Connections between μC , TMC4361A, any motor driver stage, motor M and an incremental ABN encoder which is connected directly to the motor M.

2.5 Register access from μC to TMC4361A

Ensuing datagrams (from the μC to TMC4361A) will be written as hexadecimal order extensions to the current values of the particular registers.

Please note the most significant bit has to be '1' for writing access to the register. Please refer to the TMC4361A manual for further information.

For example, if bit13 of a given GENERAL_CONF register 0x00 has to be set to '1' and bit18 to '0', the following sequence result in the required datagram:

```
GENERAL_CONF && 0xFFFFFBFFFF || 0x8000002000.
GENERAL_CONF (example)      =      0x0012345678
&& 0xFFFFFBFFFF             →      0x0012305678
|| 0x8000002000             →      0x8012307678
```

Now, 0x8012307678 can be sent to TMC4361A and the particular switches will be set.



2.6 Setup SPI output configuration for TMC motor (gate) drivers

If current datagrams should be sent from TMC4361A to TMC motor (gate) drivers and/or configuration datagrams from the μ C through TMC4361A to TMC motor (gate) drivers, the SPI output have to set appropriately in the **SPIOUT_CONF** register **0x04**. The setup depends on the motor (gate) driver which is connected to TMC4361A. In the following summary the timing is equally:

- SPI_OUT_BLOCK_TIME = 8 clock cycles
- SPI_OUT_LOW_TIME = 4 clock cycles
- SPI_OUT_HIGH_TIME = 4 clock cycles

This will result in an eightfold slower data rate than the base clock frequency. Faster and slower transfers are possible and depends on the board layout. The COVER_DATA_LENGTH will be set to 0 to enforce automatic assign at which the bit length for cover datagrams is adapted in dependence of the selected TMC motor (gate) driver.

Furthermore, the bits12:4 can be used to adapt the communication with the driver further. For further information about driver specific settings and timing information, please refer to TMC4361A manual.

TMC motor (gate) driver	Using only StepDir for motion control	Order μ C→TMC4361A
TMC23x	-	SPIOUT_CONF && 0x0000001FF0 0x8484400008
TMC24x	-	SPIOUT_CONF && 0x0000001FF0 0x8484400009
TMC26x	-	SPIOUT_CONF && 0x0000001FE0 0x848440000A
	✓	SPIOUT_CONF && 0x0000001FE0 0x848440000B
TMC389	-	SPIOUT_CONF && 0x0000001FF0 0x848440001A
	✓	SPIOUT_CONF && 0x0000001FF0 0x848440001B
TMC21xx	-	SPIOUT_CONF && 0x0000001FE0 0x848440000D
	✓	SPIOUT_CONF && 0x0000001FE0 0x848440000C

2.7 Setup INTR output configuration

To avoid polling for an executed cover datagram the usage of the INTR output pin is advisable. After configuring the interrupt polarity (default: low active interrupt), the COVERDONE event (**bit25** in events register **0x0E**) have to be assigned as INTR output in the **INTR_CONF** register **0x0D**.

Then, the status event register **0x0E** have to cleared. This can be done easily by reading it.

Now, every interrupt indicates a transferred cover datagram. The next cover datagram can be initiated then. After every interrupt, the status event register 0x0E have to be read to clear the interrupt.

The following sequence will be repeat in most order sequences at startup in the next chapters. It comprises the assignment of the COVER_DONE event as interrupt at INTR and the initial clear of all EVENTS bits after startup.

Sequence	Comments	Order μ C→TMC4361A
1.	Assign COVERDONE as INTR	0x8D02000000
2.	Clear events once after start up	0x0E00000000

NOTE:

→ If certain bits of the EVENTS register should be protected by utilizing **EVENT_CLEAR_CONF** register **0x0C** the bit has to be reset by a writing access to the EVENTS register 0x0E. Please refer to TMC4361A datasheet for further information.

©2020 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany

Terms of delivery and rights to technical changes reserved.

Download newest version at www.trinamic.com



2.8 Sending cover datagrams to setup TMC26x/389

TMC24x/23x are not configurable via register request. Other TMC stepper motor (gate) drivers can be configured using the cover datagram feature of TMC4361A. The mechanism will be explained with the help of TMC26x/389. Thereby, five registers have to be set to put the driver into operation.

The read response of the driver can be read out at the **COVER_DRV_LOW** register **0x6E**. Anyhow, this is not necessary as the status bits will be assigned automatically to the status flag register **0x0F** of TMC4361A.

In the following the sequence of datagrams from μC to TMC4361A will be displayed with an **example** for a TMC26x/389 setup. Important values to configure are the chopper settings (CHOP_CONF), the current scale (CS) bit which will determine the maximum current limit at all for the motor (gate) driver, the StepDir disable bit (SDOFF) and the sense resistor value (VSENSE). All register for the TMC26x/389 should be adapted to the current board setup. Please refer the TMC26x/389 manual for detailed information.

Sequence	Comments	Order $\mu\text{C}\rightarrow\text{TMC4361A}$
1.	Assign COVERDONE as INTR and clear events	See section 2.7
2.	Set the DRVCTRL register of TMC26x/389 (cover datagram): single edge steps, disable step interpolation, microstep resolution: 256	0xEC0000000
3.	Wait for an interrupt and then clear events.	...0x0E0000000
4.	Set the CHOPCONF register of TMC26x/389 (cover datagram): tbl=36, standard chopper, HDEC=16, HEND=11, HSTR=1, TOFF=5, RNDTF=off	0xEC00090585
5.	Wait for an interrupt and then clear events.	...0x0E0000000
6.	Disable the SMARTEN register of TMC26x/389 (cover datagram)	0xEC000A0000
7.	Wait for an interrupt and then clear events.	...0x0E0000000
8.	Set the SGCSCONF register of TMC26x/389 (cover datagram): SGT=0, CS=31 (maximum value!)	0xEC000C001F*
9.	Wait for an interrupt and then clear events.	...0x0E0000000
10. a)	Set the DRVCONF register of TMC26x/389 (cover datagram): SLPH=3, SLPL=3, DISS2G=off, TS2G=0-3.2us, SDOFF=on, VSENSE=0* <ul style="list-style-type: none"> - Use this configuration if TMC26x/389 should evaluate current datagrams and not StepDir input signals) - TMC4361A SPI_OUT CONF register has to be set accordingly 	0xEC000EF080*
10. b)	Set the DRVCONF register of TMC26x/389 (cover datagram): SLPH=3, SLPL=3, DISS2G=off, TS2G=0-3.2us, SDOFF=off, VSENSE=0* <ul style="list-style-type: none"> - Use this configuration if TMC26x/389 should evaluate StepDir input signals and not current datagrams - TMC4361A SPI_OUT CONF register has to be set accordingly 	0xEC000EF000*
11.	Wait for an interrupt and then clear events.	...0x0E0000000

*Current settings will be explained in the next section



2.9 Example configurations of appropriate current settings for TMC26x

The current settings are dependent on the used motor type. In the following two examples are explained, how to set correct current settings for CS (current scaling) and VSENSE of TMC26x. The current examples are configured as regards **peak current** due to the fact that these settings of TMC26x will provide the maximum current which can be reached:

Example 1: Conditions: Motor current $I_{RMS} = 0.85$ A, Sense resistor: $R_{SENSE} = 0.22$ Ω

Settings:

$$\begin{aligned} V_{SENSE} = 0 & \rightarrow \text{Sense resistor voltage } V_{FS} = 305\text{mV (FS - Full scale)} \\ & \rightarrow \text{Maximum possible current } I_{FS} = V_{FS} / R_{SENSE} = 1.39 \text{ A} \\ I_{PEAK} = I_{RMS} \cdot \sqrt{2} & = 1.2 \text{ A} \\ & \rightarrow CS = I_{PEAK} / I_{FS} \cdot 32 - 1 = 26 \end{aligned}$$

Result:

Cover datagram at sequence no. 8: **0xEC000C001A** and
 Cover datagram at sequence no. 10a): **0xEC000EF080** or
 Cover datagram at sequence no. 10b): **0xEC000EF000**

Example 2: Conditions: Motor current $I_{RMS} = 0.5$, Sense resistor: $R_{SENSE} = 0.15$ Ω

Settings:

$$\begin{aligned} V_{SENSE} = 1 & \rightarrow \text{Sense resistor voltage } V_{FS} = 165\text{mV} \\ & \rightarrow \text{Maximum possible current } I_{FS} = V_{FS} / R_{SENSE} = 1.1 \text{ A} \\ I_{PEAK} = I_{RMS} \cdot \sqrt{2} & = 0.71 \text{ A} \\ & \rightarrow CS = I_{PEAK} / I_{FS} \cdot 32 - 1 = 19 \end{aligned}$$

Result:

Cover datagram at sequence no. 8: **0xEC000C0013** and
 Cover datagram at sequence no. 10a): **0xEC000EF0C0** or
 Cover datagram at sequence no. 10b): **0xEC000EF040**

ATTENTION If no closed-loop current scaling (chapter 4) is used, current settings should match the **RMS current and not the peak current to avoid overcurrents** during motion with a low velocity!

2.10 Example configuration for TMC21x0

In the following the sequence of cover datagrams from μC to TMC4361A will be displayed with an example for a TMC21x0 setup. The start-up sequence is the same as explained for the TMC26x, except that both cover registers (**COVER_LOW 0x6C** and **COVER_HIGH 0x6D**) have to be used as 40 bits overall are sent and received to / from TMC21x0. The read response of the driver can be read out at the **COVER_DRV_LOW** register **0x6E** and **COVER_DRV_HIGH 0x6F**. Important values to configure are the chopper settings (CHOP_CONF) including VSENSE, the current scale (CS) bit which will determine the maximum current limit at all for the motor (gate) driver, the StepDir disable bit (direct_mode) of GCONF register. Any other registers for the TMC21x0 should be adapted to the actual board setup. Please refer the TMC21x0 manual for detailed information.



Sequence	Comments	Order µC→TMC4361A
1.	Assign COVERDONE as INTR and clear events	See section 2.7
2. a)	Set the GCONF 0x00 register of TMC21x0 (cover datagrams): direct_mode=0 → StepDir mode on <ul style="list-style-type: none"> - Use this configuration if TMC26x/389 should evaluate current datagrams and not StepDir input signals) - TMC4361A SPI_OUT CONF register has to be set accordingly 	0xED00000080 0xEC00000000
2. b)	Set the GCONF 0x00 register of TMC21x0 (cover datagrams): direct_mode=1 → StepDir mode off <ul style="list-style-type: none"> - Use this configuration if TMC26x/389 should evaluate current datagrams and not StepDir input signals) - TMC4361A SPI_OUT CONF register has to be set accordingly 	0xED00000080 0xEC00010000
3.	Wait for an interrupt and then clear events.	...0x0E00000000
4.	Set the CHOPCONF register 0x6C of TMC21x0 (cover datagrams): 256 microsteps, vsense=0, TBL=36 clock cycles, spread cycle chopper, HEND=1, HSTRT=2, TOFF=3	0xED000000EC 0xEC00010223**
5.	Wait for an interrupt and then clear events.	...0x0E00000000
6. a)	Set the IHOLD_IRUN register 0x10 of TMC21x0 (cover datagram): IHOLD_DELAY=5, IHOLD=31, IRUN=31 <ul style="list-style-type: none"> - SPI mode (Step Dir mode off): Set IHOLD to maximum as all incoming current datagrams are scaled by this value 	0xED00000090 0xEC00051F1F**
6. b)	Set the IHOLD_IRUN register 0x10 of TMC21x0 (cover datagram): IHOLD_DELAY=5, IHOLD=15, IRUN=31 (maximum value!)	0xED00000090 0xEC00050F1F**
7.	Wait for an interrupt and then clear events.	...0x0E00000000

**Current settings will be explained in the next section

2.11 Example configuration of appropriate current settings for TMC21x0

The current settings are dependent on the used motor type. In the following example is explained, how to set correct current settings for IRUN and/or IHOLD (current scaling) and VSENSE of TMC21x0. The current example is configured as regards **peak current** due to the fact that these settings of TMC21x0 will provide the maximum current which can be reached:

Example 1: Conditions: Motor current $I_{RMS} = 0.85 \text{ A}$, Sense resistor: $R_{SENSE} = 0.22 \Omega$

Settings:

$$\begin{aligned}
 V_{SENSE} = 0 & \rightarrow \text{Sense resistor voltage } V_{FS} = 320\text{mV (FS - Full scale)} \\
 & \rightarrow \text{Maximum possible current } I_{FS} = V_{FS} / R_{SENSE} = 1.33 \text{ A} \\
 I_{PEAK} = I_{RMS} \cdot \sqrt{2} & = 1.2 \text{ A} \\
 & \rightarrow CS = I_{PEAK} / I_{FS} \cdot 32 - 1 = 27
 \end{aligned}$$

Result:

Cover datagram at sequence no. 4: set vsense=0
 Cover datagram at sequence no. 6a): **0xEC00051B1B** or
 Cover datagram at sequence no. 6b): **0xEC0005xx1B**



3 Basic encoder setup for closed-loop control

In the following the basic encoder setup for TMC4361A closed-loop operation will be explicated.

3.1 General encoder selection

As explained, different encoder types (selection by setting **bit11:10** of GENERAL_CONF register **0x00**) will be supported. There is also an option to disable differential inputs by enabling **bit12** of GENERAL_CONF. **Per default, differential inputs are enabled, except for SPI encoders!** Following possibilities are practicable:

Encoder type	Differential output	Order $\mu\text{C} \rightarrow \text{TMC4361A}$
ABN encoder	✓	GENERAL_CONF && 0x00FFFE3FF 0x8000000000
	-	GENERAL_CONF && 0x00FFFE3FF 0x8000001000
SSI encoder	✓	GENERAL_CONF && 0x00FFFE3FF 0x8000000400
	-	GENERAL_CONF && 0x00FFFE3FF 0x8000001400
SPI encoder	-	GENERAL_CONF && 0x00FFFF3FF 0x8000000C00

The N event configuration can be adapted in the **ENC_IN_CONF** register **0x07**: bits(11:2) have to be set particularly. Please note that clear_on_n (bit1) must not be set because clearing the external position ENC_POS will definitely disrupt correct closed-loop behavior in most cases.

3.2 Absolute encoder configuration

Before absolute encoders are assigned as active (by setting GENERAL_CONF accordingly), please configure the data transfer scheme properly.

Therefore, changing the serial encoder data input register **ENC_IN_DATA 0x08** will lead in a different bit length for the data transfer and the corresponding interpretation of the data by TMC4361A. Further on, multi cycle data and gray code enabling can be set by adapting **bit17** and **bit18** of **ENC_IN_CONF** register **0x07**. **Bit19** will change the interpretation of the alignment of the data.

By setting the bit count of the data transfer, the number of clock cycles which will be generated by TMC4361A for the serial encoder are automatically assigned. The timing of the data transfer/clock can be adapted by changing the registers **0x56** (Low and High level duration), **0x57** and **0x58** which will tune required delay times for the absolute encoder data transfer. Please refer to TMC4361A manual for further information.

Finally, SPI data transfer of SPI encoders can be changed by using **bit20** and **bit21** of the **ENC_IN_CONF** register **0x07**.

Bit12 and bit13 of ENC_IN_CONF register 0x07 enable multiturn data if the encoder supports this feature. If only singleturn data is transmitted from the encoder, in closed-loop motor control applications multiturn data has to be calculated by TMC4361A. Thus, set **bit16** (calc_multi_turn_behav) of register **0x07** to '1'. This way, TMC4361A will add or remove one revolution if an overflow for the singleturn data is recognized.

AREAS OF SPECIAL CONCERN

Restrict maximum data variation

The difference between encoder values of consecutive data transfers must not exceed the half of one revolution during multiturn calculation mode.

To avoid the acceptance of encoder values that differ significantly, please use bit31 of the **ENC_IN_CONF** register 0x07 and the **SER_ENC_VARIATION** register 0x63 - bit(7:0). Enabling this **serial_enc_variation_limit** switch results in an automatic rejection of encoder data that differ more than $1/8 \cdot \text{SER_ENC_VARIATION} \cdot \text{ENC_IN_RES}$ compared to actual ENC_POS.



3.3 Encoder resolution

A correct setup of the encoder resolution is mandatory. First of all, the full step per revolution have to be setup properly. Thus, the **FS_PER_REV** value in the **STEP_CONF** register **0x0A** have to be checked (default: 200 full steps per revolution) and, if necessary, changed. Further on, the microsteps per fullsteps **MSTEP_PER_FS** must not be adapted because 256 microsteps are mandatory for closed-loop operation.

NOTE:

→ *The microstep per fullsteps **MSTEP_PER_FS MUST be set to 256** in register 0x0A in the case of closed-loop control.*

Necessarily, the encoder resolution **ENC_IN_RES** (register **0x54**) have to be set. This value is defined as **encoder steps per revolution**.

The internally calculated microstep per encoder step constant will be able to read out at 0x54(30:0) with 15 digits and 16 decimal places.

This value represents the number of microsteps which will be added to the encoder position **ENC_POS** with every detected encoder step for incremental encoders. For absolute encoder, this constant will be multiplied with the current angle which will also result in an encoder position **ENC_POS** representing microsteps. The encoder constant will be calculated as follows:

$$\text{ENC_CONST} = \text{MSTEPS_PER_REV} \cdot \text{FS_PER_REV} / \text{ENC_IN_RES}.$$

Per default, the decimal places represent binaries. But for some application, it will be better use a decimal representation of the decimal places. If so, the decimal places 0x54(15:0) are calculated as 1/10000.

If calculated automatically, TMC4361A will take the representation which will suit the best. Only, if both representation are not able to map the encoder constant to exactly 16 decimal places, bit0 of the **ENC_IN_CONF** register will define the binary (bit0='0') or decimal (bit0='1') representation. Please note that a not exact encoder constant can lead the encoder mismatches in the long run. On the next page, two examples clarifies the differences between both representations.

Example 1:

An incremental encoder with **8192 pole pairs** will generate **32768 encoder steps per revolution** due to 4 possible AB transition per pole pair. Thus, **ENC_IN_RES** have to be set to 32768. With **256 microsteps** per full step and **200 full steps** per revolution, this leads to an encoder constant of $\text{ENC_CONST} = 1.5625$. This value can be exactly represented as binary number which can be read out then at 0x54: **ENC_CONST = 0x00019000**

Example 2:

An incremental encoder with **500 pole pairs** will generate **2000 encoder steps per revolution** due to 4 possible AB transition per pole pair. Thus, **ENC_IN_RES** have to be set to 2000. With **256 microsteps** per full step and **200 full steps** per revolution, this leads to an encoder constant of $\text{ENC_CONST} = 25.6$. Six tenth can not be exactly represented with 16 binary decimal places. Therefore, the decimal representation will be used as **ENC_CONST** with 6000/10000: **ENC_CONST = 0x00191770**

If it is necessary to calculate the encoder constant manually, it can be written directly to 0x54 by setting the MSB of this register to '1'. For example, if the encoder constant should be set to 25.6 manually, write 0x80191770 to register 0x54. Because this is a decimal representation, bit0 of 0x07 have to be set to '1'. Otherwise, the encoder step position will be calculated with a binary encoder constant.

Finally, the encoder direction should match the motor direction. This can be reached by various settings. Thus, please move the motor with in an open loop setup in any direction (the encoder resolution has to be set before). If the internal position **X_ACTUAL** and the external position **ENC_POS** will not differ in its signs, the directions of the motor and the encoder match. No change are



necessary. As opposed to a match, following possibilities for a correct setup are available if motor and encoder diverge:

1. Invert the encoder direction by setting invert_enc_dir **bit29** of ENC_IN_CONF register **0x07** or
2. Interchange the motor cables or
3. Remount the encoder.

It is recommended to use the first setup change as no hardware change will be required.

AREAS OF SPECIAL CONCERN

Do not use the invert_enc_dir switch in combination with a serial encoder and a manual defined encoder constant because the value of ENC_IN_RES defines the resolution for one revolution.

This combination will result in failures due to the fact that the manual encoder constant equals not the revolution resolution in most cases.

3.4 Encoder compensation

Systematical encoder misalignments can be compensated on chip. Especially magnetic encoders have high variances on the output value even if the encoder is correctly mounted. A deficiently installed encoder can send values which do not result in a circle. Often, the deviation from the real position results in a new function which is similar to a sine function. Adding offset that follows a triangular shape can improve the encoder value evaluation significantly (refer to Figure 3.1).

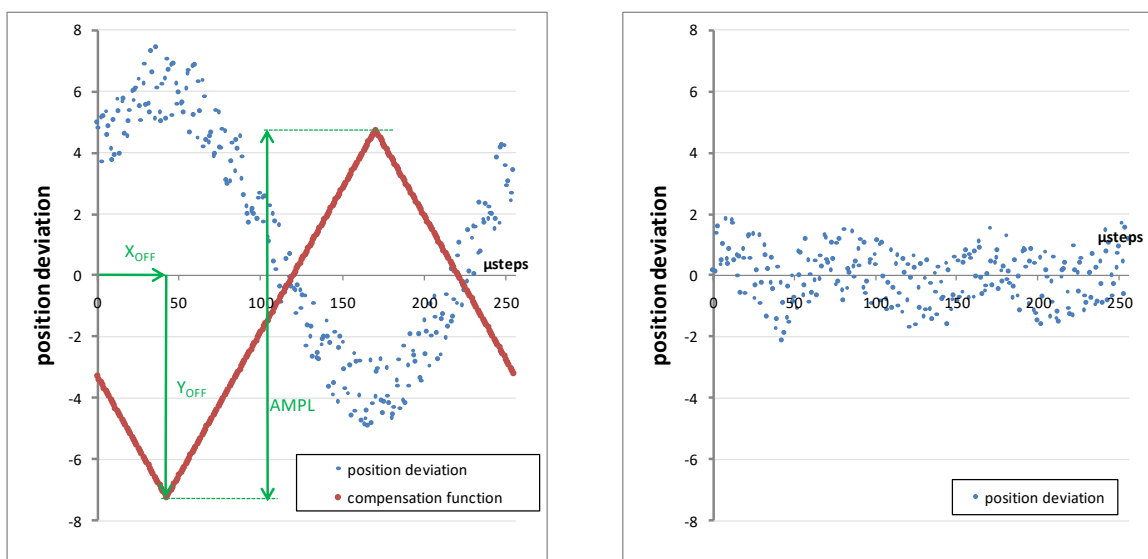


Figure 3.1 Implemented triangular function to compensate for encoder misalignments

The left graph illustrates the difference between encoder position and real position (blue dots - position deviation) as a μ step function within one encoder revolution of 256 microsteps. Blending in a proper triangular function whose function values will be added to the position deviation values will result in another graph at the right side whose position deviations are now minimized significantly compared to the uncompensated values. TMC4361A provides this simple compensation algorithm which adds automatically an microstep offset to the internal microstep counter ENC_POS if turned on. Only three parameter have to be assigned properly in register **0x7D**:

- **ENC_COMP_XOFFSET** (X_{OFF}): the 16bit unsigned abscissa value which is normalized to the number of microsteps per revolution
- **ENC_COMP_YOFFSET** (Y_{OFF}): the 8bit signed ordinate value of the minimum of the triangular function in microsteps
- **ENC_COMP_AMPL** ($AMPL$): the 8bit unsigned maximum amplitude value of the triangular function which **must not exceed 127!**



NOTES:

- *The triangular function must be opposed to the position deviation function to compensate properly which means that the maximum of the position deviation function will lead to the minimum of the triangular function and vice versa.*
- *The abscissa value of the maximum of the triangular function is defined a **half revolution distant to the minimum!***
- *The compensation algorithm have not to be turned on explicitly. It is switched on all the time, but as long as the encoder compensation registers are set to 0, all values will be compensated with 0.*

To obtain the correct value for ENC_COMP_XOFFSET the minimum of the triangular function have to be assigned. The function argument of this minimum is the X offset. As this register is normalized to the microstep count per revolution, the value have to be **divided by the number of microsteps per revolution and then multiplied by 2¹⁶**.

Example 1:

A motor with 200 fullsteps and 256 microsteps per fullstep has 51200 microsteps per revolution. The minimum for the triangular function is located at (10000 microsteps; -12 microsteps) with a maximum deviation of 65 microsteps

- ENC_COMP_XOFFSET = $10000 / 51200 * 2^{16} = 12800$
- ENC_COMP_YOFFSET = -12
- ENC_COMP_AMPL = $65 - (-12) = 77$

The maximum of the triangular function will be then at (35600;65)

Example 2:

A motor with 72 fullsteps and 256 microsteps per fullstep has 18432 microsteps per revolution. The minimum for the triangular function is located at (11000 microsteps; -54 microsteps) with a maximum deviation of 8 microsteps

- ENC_COMP_XOFFSET = $11000 / 18432 * 2^{16} = 39111$
- ENC_COMP_YOFFSET = -54
- ENC_COMP_AMPL = $8 - (-54) = 62$

The maximum of the triangular function will be then at (1784;65)

Automatic Linearization of encoder misalignments

Following sequence suggest how to obtain compensation data for a given setup:

1. Setup the motor driver and the encoder properly (**no load!, maximum current**)
2. Move motor to ENC_POS = 0.
3. Set X_ACTUAL = 0.
4. Move in one direction for several revolutions and save the ENC_POS_DEV values at every fullstep position (MSCNT = 128/384/640/896)
5. Move backwards and save the same data and generate the average over both direction and the several revolutions
6. Calculate the minimum value and the maximum amplitude of the triangular function by comparing the minimum and maximum of the stored position deviation values
7. Set the compensation value register 0x7D properly
8. Move again in both direction and check if the compensation is working properly. If not, repeat from 2.
9. After later executed power-ups, it has to be verified that the positions XACTUAL and ENC_POS do not differ to the settings of the initialization process (2.-7.)

NOTE:

- *Due to the fact that the maximum of the triangular function is located a half revolution away from the minimum, it can be advisable to try different settings for the compensation register around the minimum to get the best compensation result if the real position deviations do not fit exactly to sine function!*



4 Closed-loop setup

After setting the TMC4361A encoder properties properly, all closed-loop properties should be setup accordingly. **Please be aware that before calibrate and switch on closed-loop operation mode that is explained in detail at the end of this chapter, every configuration setup step has to be finished, also the optional features described in the following chapters 5 and 6.**

For a basic closed-loop motor control operation, only a few values have to be set:

1. Dependent on the actual mismatch between encoder position ENC_POS and internal position XACTUAL, the external field which will be initiated by TMC4361A will be altered by an angle offset to overcome the mismatch. This maximum commutation angle **CL_BETA 0x1C(8:0)** can be set in a range between 0 and 511 microsteps whereas 255 microsteps are equal to 90° when 256 microsteps per full step are set. As long as the position mismatch exceed CL_BETA microsteps the maximum commutation angle will be used to resolve the position mismatch. Because an offset of 90° to the actual motor angle are equivalent to the largest force which can be applied to the motor, CL_BETA = 255 (default value) would fit for best performance. Higher angle values will result in a field weakening operation which can lead to higher velocities but lesser force.

NOTE:

→ If the maximum difference CL_BETA is reached, regulation will be continued with a maximum commutation angle of CL_BETA! An event **CL_MAX** will be generated in this case.

2. The closed-loop proportional term **CL_DELTA_P 0x5C** defines the response to a position mismatch as long as the maximum commutation angle is not reached. The higher this value the stronger the response to position mismatches, but also the higher the possibility for oscillations. Higher values means also that the maximum commutation angle CL_BETA will be reached with a smaller position mismatch (see Figure 4.1). The representation of CL_DELTA_P consists of 8 digits and 16 decimal places, e.g. CL_DELTA_P = 0x018000 = 1.5 will result in a commutation angle which is 1.5 multiplied to the current position mismatch. In the following, three different setups are illustrated in a table to clarify the differences of various settings (microsteps per fullstep = 256):

Setup No:	1	2	3
CL_BETA =	255 (0x0FF)	200 (0x0C8)	275 (0x113)
CL_DELTA_P =	1.5 (0x018000)	0.9375 (0x00F000)	2.75 (0x02C000)
Position mismatch (microsteps/angle)	Commutation angle (microsteps/angle) (red entry - maximum angle reached!)		
36 / 12.7°	54 / 19.0°	34 / 12.0°	99 / 34.8°
96 / 33.8°	144 / 50.6°	90 / 31.6°	264 / 92.8°
148 / 52.0°	222 / 78.0°	139 / 48.9°	275 / 96.7°
210 / 73.8°	255 / 90°	197 / 69.3°	275 / 96.7°
266 / 93.5°	255 / 90°	200 / 70.3°	275 / 96.7°

3. Finally, setting **CL_TOLERANCE 0x5F** (microsteps) properly will result in a mismatch range between -CL_TOLERANCE and +CL_TOLERANCE where CL_DELTA_P is automatically equal to 1.0. It is recommended to set CL_TOLERANCE at least slightly higher than the ENC_CONST to avoid too fast responses (if CL_DELTA_P > 1.0) on encoder transitions due to encoder noise.

NOTE:

→ If the difference between internal and external position is inside of the range created by the CL_TOLERANCE value, a **flag CL_FIT_F** will be released. If the ENC_POS_DEV have been greater than the tolerance before, an **event CL_FIT** will be also generated to indicate a concordance of internal and external position after a mismatch.



4.2 Closed-loop operation during ramp positioning mode

If closed-loop operation is turned on during internal positioning mode, TARGET_REACHED status will not be activated as long as the absolute value of the position mismatch ENC_POS_DEV will exceed the assigned value in the **CL_TR_TOLERANCE** register **0x52**. This ensures that even when the internal ramp has been finished, no target reached flag/event is released as long as the external position will not match the internal position within the given tolerance range.

4.3 Closed-loop velocity mode

It is also possible to use turn on the closed-loop velocity mode by switching on **bit28** of the **ENC_IN_CONF** register **0x07**:

```
ENC_IN_CONF && 0x00FE7FFFFFFF || 0x8710400000.
```

During this modus, XACTUAL will be manipulated if the position difference ENC_POS_DEV exceeds 768 microsteps. If so, XACTUAL will be changed for 256 microsteps towards ENC_POS. Thus, the position mismatch should not exceed 768 microsteps. That way, a velocity mode can be achieved where TMC4361A will always try to match the given velocity VMAX.

Further on, if the maximum velocity VMAX should not be exceeded, it is recommended to set PID_DV_CLIP = 0 and to enable the velocity limit. This setting will allow a catch-up as the position mismatch is relatively small. Anyhow, if the encoder is perfectly mounted and have a high resolution, it is possible that catch-up will be very slow or will not be achieved at all. If so, set PID_DV_CLIP to a low value (e.g. between 1 and 1000) according to your maximum value VMAX.

NOTE:

- *Basically, this closed-loop operation mode fits to the velocity ramp mode. Anyhow, it is not forbidden to use positioning mode with the velocity closed-loop operation mode. Due to the feasible permanent adaptations of the internal position XACTUAL, it is possible that the internal ramp can miss the end point of the internal positioning ramp.*



4.4 Closed-loop calibration and activation

Now, the basic setup is ready to drive in closed-loop motor control modus. **Please be aware that before calibrate and switch on closed-loop operation mode, every configuration setup step has to be finished, also the optional features described in the following chapters 5 and 6.** Turning on closed-loop operation by setting the **regulation_modus** switch = b'01 of **ENC_IN_CONF** register **0x07** should be sufficient as calibration of closed-loop operation will be done automatically when closed-loop modus is switched on. It is only required that XACTUAL register 0x21 equals MSCNT register 0x79.

Anyhow, for best performance of **a compensated encoder** the following calibration sequence should be followed:

1. Disable any current scaling and calibrate with maximum current.
2. **Move to any full step position** in open loop mode which means that the absolute current values of the sine and cosine waveform lookup table should be equal: $|CURRENTA| = |CURRENTB|$ (CURRENTA register 0x7A(8:0); CURRENTB register 0x7A(24:16)). Using the TMC4361A default waveform and 256 microsteps per full step, full step position is reached if $XACTUAL \bmod 256 = 128$, e.g. $XACTUAL = 128$ or 384 or -640 or 14976 , ... Here, the most stable position should be reached as the current through all inductors amount the same absolute value which leads to an equal force from each side affecting the rotor.
If the motor (gate) driver uses its own sine wave (e.g. using TMC26x with StepDir input), move to a full step position of the motor driver!
3. Ideally, the encoder position ENC_POS should not switch if possible. If the encoder position jiggles, try the next full step position.
4. Then, turn on closed-loop operation (**bit23:22 of ENC_IN_CONF 0x07 = b'01**) and calibrate immediately (**bit24 of ENC_IN_CONF 0x07 = '1'**):

```
ENC_IN_CONF && 0x00FE3FFFFFF || 0x8701400000
```
5. After waiting for at least 10us, turn off closed-loop calibration:

```
ENC_IN_CONF && 0x00FE3FFFFFF || 0x8700400000
```

Now, closed-loop operation is active and TMC4361A will always try to match the external ENC_POS with the internal position XACTUAL. The current difference between both can be read out at the **ENC_POS_DEV** register **0x52**. During closed-loop operation, the value of ENC_POS_DEV considers the closed-loop offset **CL_OFFSET** which can be read out at register **0x59**. This value is the position difference during the calibration process. It is readable, but also writeable.

NOTE:

→ *The calibration phase (step 4) can be skipped in case XACTUAL equals MSCNT. Set it up after step 3 in case XACTUAL can be adapted before closed-loop operation.*

AREAS OF SPECIAL CONCERN

To avoid erroneous closed-loop behavior, **do not write to closed-loop offset CL_OFFSET register 0x59 during closed-loop operation** except for calibration purpose:

To avoid recalibration at restart after switching off closed-loop operation, turn on closed-loop and then write back the stored CL_OFFSET value to its register. The application should move in the same manner as before.

Using CL_OFFSET can also **avoid the recalibration** of the application after a shutdown. Especially systems with absolute encoders would benefit of this recalibration option. Additionally, if XACTUAL and the number of revolutions would written back to ENC_POS (if absolute encoders are turned on) before starting closed-loop operation, the whole application would find its last stable closed-loop position, even if the motor have been moved during off-time.

But beware, do not use this option for recalibration if an incremental encoder is used when the N signal is not involved to get the correct absolute position in one revolution!



5 Current scaling during closed-loop motor control

The TMC4361A provides a scaling unit which can also be used during closed-loop operation. Thereby, energy can be saved as long as no mismatch occur. Basically, up to four parameters can be set by assigning the **SCALE_VALUES** register **0x06** properly. The particular real resulting scale value of one of the parameters x (which will be introduced in the following) is calculated by the following equation:

$$\text{real_scale_value} = (x + 1) / 256.$$

AREAS OF SPECIAL CONCERN

If TMC26x / TMC21xx motor (gate) driver are used in StepDir mode and the current values should be transferred via cover datagrams, only the 5 least significant bits of the current values are transmitted.

Thus, following equation holds true for this setup:

$$\text{real_scale_value} = (x + 1) / 32$$

NOTE:

→ *The subsequently introduced values are based on the current scaling with a maximum value of 255. If the StepDir mode with scale value datagrams are used, please adapt the introduced values accordingly.*

In register **0x06**, the following values can be set:

- **Closed-loop maximum current scale value CL_IMAX = SCALE_VALUES(15:8):**
By setting this register, the maximum current will be set. This current scale value is reached as soon as the position mismatch ENC_POS_DEV exceeds the CL_BETA limit. If during TMC26x configuration (see section 2.8) the RMS current is assigned as CS scale value, CL_IMAX can be set to 255. Otherwise, if the peak current I_{PEAK} is set, CL_IMAX should set to 180 due to the fact that $\sqrt{2} \cdot 256 = 181$. Thus, the maximum current will not exceed I_{RMS} . This is especially crucial if the motion speed is low. Anyhow, larger values are possible (e.g. to overcome heavy loads), **but beware of overcurrent over a long run!**
- **Closed-loop minimum current scale value CL_IMIN = SCALE_VALUES(7:0):**
The minimum scale value can be set freely. It will be assigned as long as the position mismatch do not exceed a certain limit. This leads to current saving behavior as long as the position mismatch is small. As good starting point, hold scale current settings for open loop behavior can be assigned. For instance, with $CL_IMIN = \frac{1}{4} \cdot CL_IMAX = 180 / 4 = 45$, up to 1/16 of the maximum energy can be saved.
- The **position mismatch limit in microsteps at which current scale values will be increased** is defined by **CL_START_UP = SCALE_VALUES(23:16)**. The current scale value will be calculated according to a linear function between CL_START_UP and CL_BETA. It is recommended to set CL_START_UP higher than CL_TOLERANCE.
- CL_START_DN defines the position mismatch value at which the ramp down of the current values will start. It is recommended, to set this parameter to 0 where this limit is set to CL_BETA. Hence, the downward ramp is the same as the upward ramp. Other settings lead to a hysteresis.

In the figure on the next page, the introduced parameters can be found. If the downward ramp is different compared to the upward ramp ($CL_START_DN > 0$ $CL_START_DN \neq CL_START_UP$), the linear function will be the same, resulting in a smaller position mismatch where the minimum scale value is reached. If CL_START_UP is set to 0, closed-loop minimum will be reached almost never as the position mismatch is almost never equal to 0.



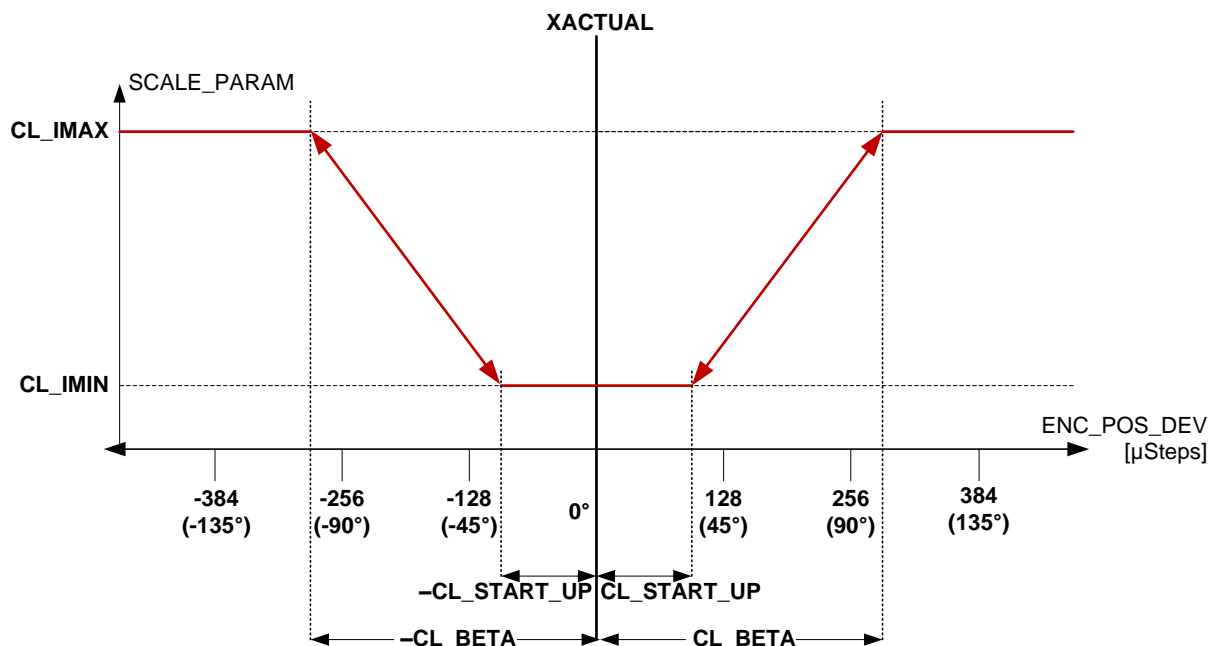


Figure 5.1 Current scaling value dependent on position mismatch during closed-loop scaling modulus in addition to closed-loop operation

For evaluating different upscaling and/or downscaling ramps the delay parameters `CL_UPSCALE_DELAY` (register 0x18) and `CL_DNSCALE_DELAY` (register 0x19) can be assigned, too. These values define the period of time in clock cycles in which the current scale value will be altered for one step towards `CL_IMAX` resp. `CL_IMIN`. Figure 5.2 depicts the current scaling timing behavior as a function of `CL_UPSCALE_DELAY` and `CL_DNSCALE_DELAY`.

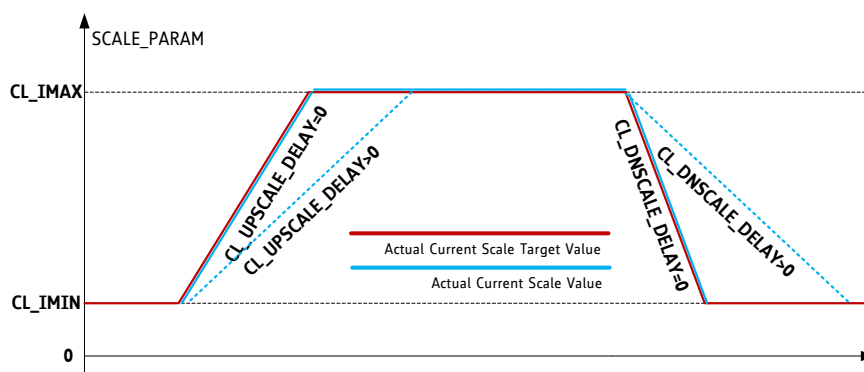


Figure 5.2 Current scaling timing behavior

After setting the closed-loop scale values properly, closed-loop scaling can be enabled:

0x8500000080.

NOTE:

→ Note that, even other scaling modes are enabled, any scale mode becomes disabled if closed-loop scaling is enabled.

5.1 Example for a combined usage of closed-loop tracking and current scaling

In the following a sequence is depicted which shows a rising deviation between the target position (internal position `XACTUAL` = red vector) and the real motor position (external position `ENC_POS` = blue vector). Due to the settings of `CL_BETA` = 255 (90°) the green dotted commutation angle will not change until the deviation exceed 90° here if `CL_DELTA_P` = 1.0 (0x010000). Thus, the green

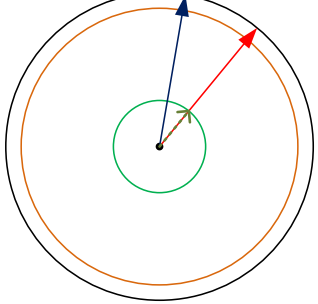
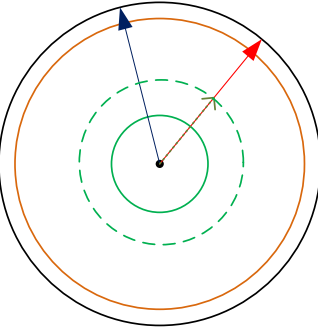
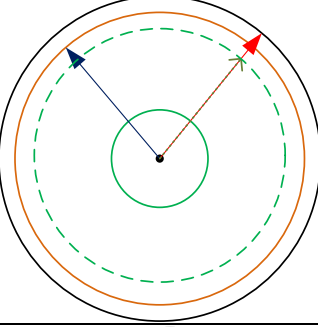
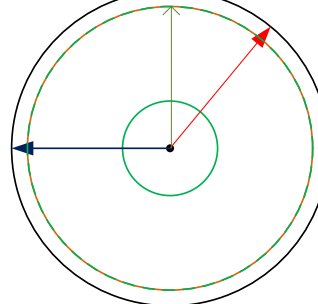
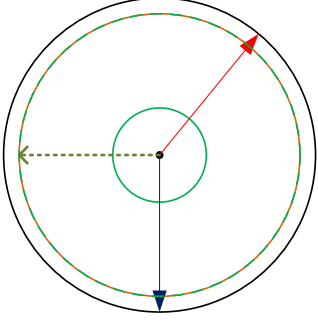
©2020 TRINAMIC Motion Control GmbH & Co. KG, Hamburg, Germany

Terms of delivery and rights to technical changes reserved.

Download newest version at www.trinamic.com



commutation angle leads the motor to the target position inside of the 90° deviation, but the green current vector will grow with the rising of the deviation to encounter the mismatch. It reaches its maximum if the limit of 90° is reached. Then, the current vector will follow the external position with full current because no steps should be lost.

Setup: CL_IMIN = 0.3 · I _{MAX} (green circle), CL_IMAX = 0.9 · I _{MAX} (orange circle), CL_STARTUP = 31.6° (90 microsteps)	Red = X _{ACTUAL} Blue = ENC_POS Green = Commutation angle
1. Small deviation= 29.7°, minimum current = 0.3 · I _{MAX} , position will be hold	
2. Moderate deviation= 53.7°, medium current = 0.53 · I _{MAX} , position will be hold	
3. Large deviation= 79.9°, large current = 0.8 · I _{MAX} , position will be hold	
4. Very large deviation > 90°, maximum current = 0.9 · I _{MAX} , „Motor follows encoder/load“	
5. The overload situation holds on, maximum current = 0.9 · I _{MAX} , „Motor follows encoder/load“	



6 Considering the back-EMF

When higher velocities are reached, a phase shift between current and voltage occurs at the motor coils. The current control will be transferred in a voltage control. This motor and setup dependent effect have to be compensated as currents will be still assigned for the motor control. This can be done by the γ -correction where a velocity dependent angle in motion direction will be added to the current commutation angle.

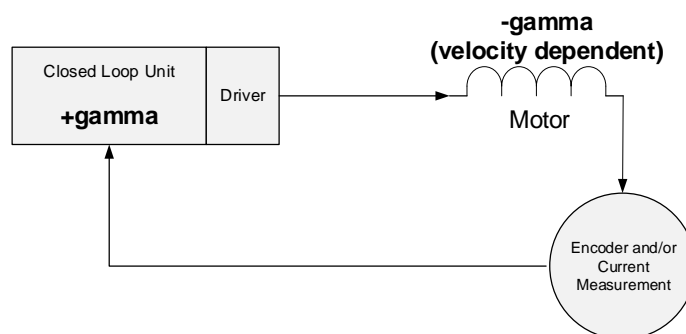


Figure 6.1 Back-EMF consideration with γ -correction unit based on measured encoder velocity

The **maximum gamma angle** can be set to almost 90° (= 255 microsteps) at **CL_GAMMA 0x1C(23:16)**. Due to the fact that this angle is added to the commutation angle, the complete angle can reach 180° if CL_BETA is also set 90° (= 255 microsteps).

AREAS OF SPECIAL CONCERN

If the γ -correction is turn on, the maximum possible commutation will be (CL_BETA + CL_GAMMA). This value must not exceed 180° (512 microsteps at 256 microsteps per fullstep).

Angles of 180° and more will result in **unwanted motion direction changes!**

Switching on γ -correction, set **cl_emf_en to '1'** in the ENC_IN_CONF register 0x07. Furthermore, the velocity limits for the γ -correction have to be assigned:

```
ENC_IN_CONF && 0x00FEFFFFFF || 0x870200000
```

- **CL_VMIN_EMF 0x60: Below** this velocity value [pps] the γ -correction angle is set to **0**.
- **CL_VADD_EMF 0x61:** This velocity value [pps] will be added to CL_VMIN_EMF to assign the upper limit for maximum γ -correction angle.
- **Beyond (CL_VMIN_EMF + CL_VADD_EMF)** the current γ -correction angle GAMMA will be set to the **maximum** value of CL_GAMMA
- Between CL_VMIN_EMF and (CL_VMIN_EMF + CL_VADD_EMF) the current γ -correction angle GAMMA will be calculated by a linear function which is depicted in the following figure

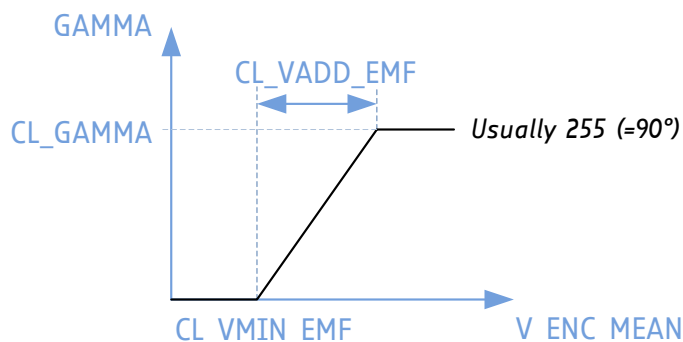


Figure 6.2 Calculation of the current back emf angle GAMMA



6.1 Velocity Setup

Setting `CL_VMIN_EMF` and `CL_VADD_EMF` properly is motor dependent. Thus, connect a current probe to one phase of the motor cables and a voltage probe on the same signal line. To set appropriate velocity limits please observe the distortion of the motor current curve. Figure 6.3 depicts a oscilloscope shot where no distortion of the motor current is observed which should be the normal case during low velocities.

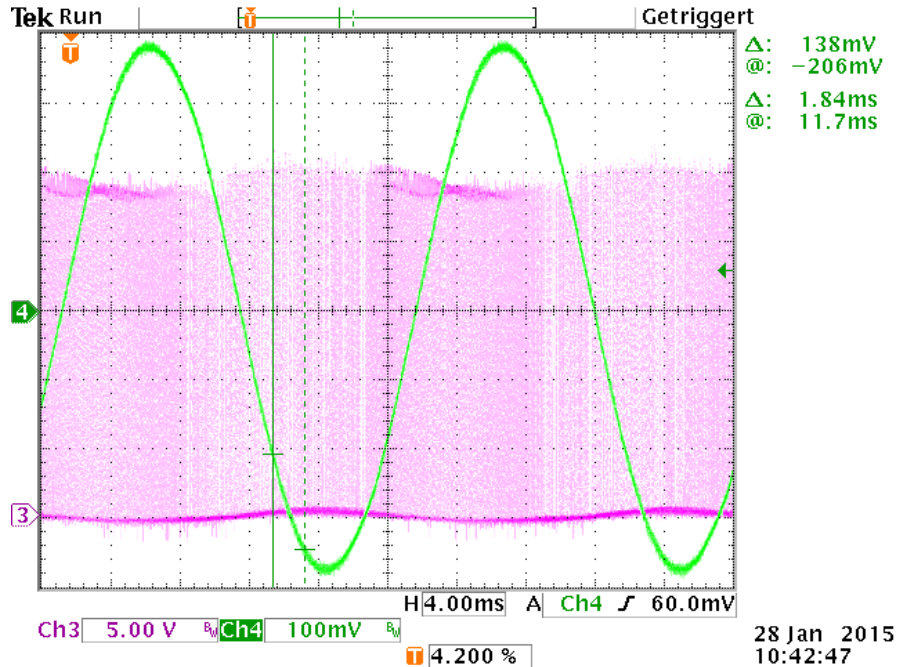


Figure 6.3 Oscilloscope shot of a motor signal line (green= motor current, magenta= motor output driver sense resistor voltage) when the current curve is not be distorted

A good starting value for `CL_VMIN_EMF` is the velocity when the back emf voltage reaches the motor supply voltage. At this point first slight distortions of the motor currents should be observed at the oscilloscope as shown in Figure 6.4.

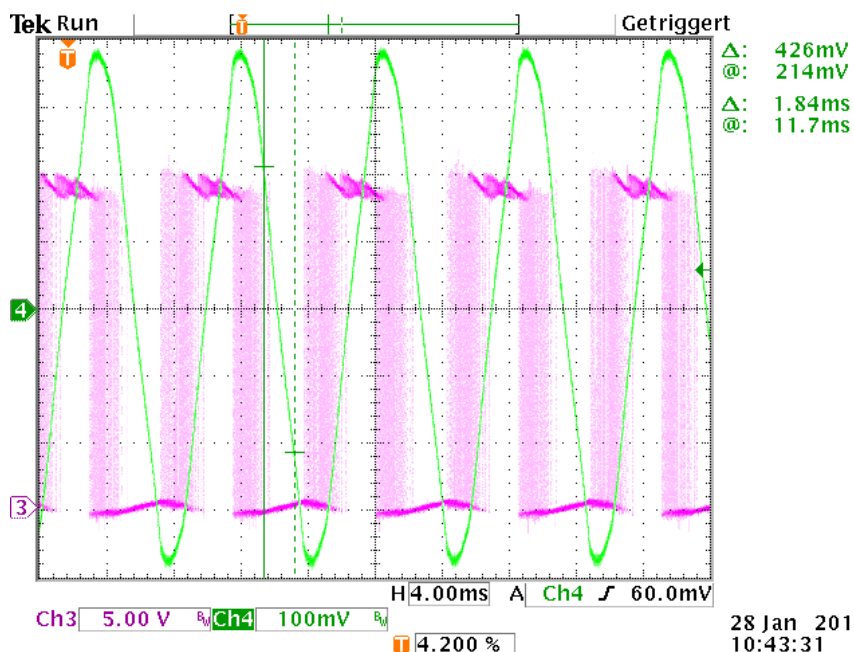


Figure 6.4 Oscilloscope shot of a motor signal line (green= motor current, magenta= motor output driver sense resistor voltage) when the current curve will begin to be distorted due to an increased back emf voltage



The velocity at which the motor currents are completely distorted firstly is a good starting value for the velocity when GAMMA will be reach its maximum of CL_GAMMA (mostly 90°). Thus, **CL_VADD_EMF** will be this velocity subtracted by CL_VMIN_EMF. Another indicator for reaching the maximum velocity for γ -correction is the vanishing of chopper cycles in the voltage curve of the sense resistor as it can be also seen in Figure 6.5 if the chopper control is turned on.

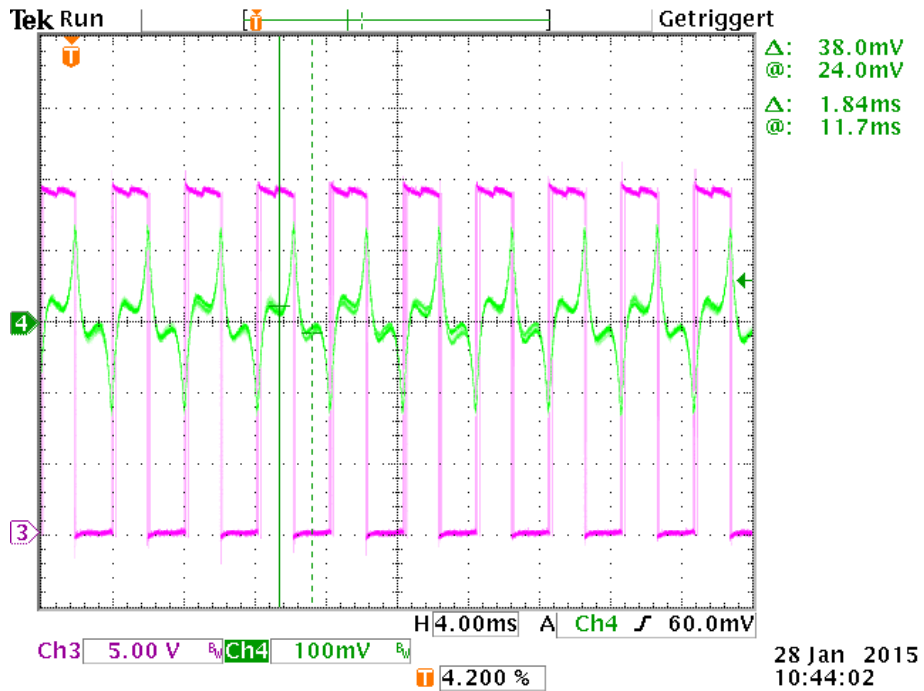


Figure 6.5 Oscilloscope shot of a motor signal line (green= motor current, magenta= motor output driver sense resistor voltage) when the current curve is completely distorted due to the back emf voltage at high velocities. Further on, no chopper cycles can be identified in the magenta voltage curve.

It is also possible to plot the maximum motor current (maximum amplitude of the motor current) vs. the ramp velocity. At the breaking points, both velocity limits can be identified.

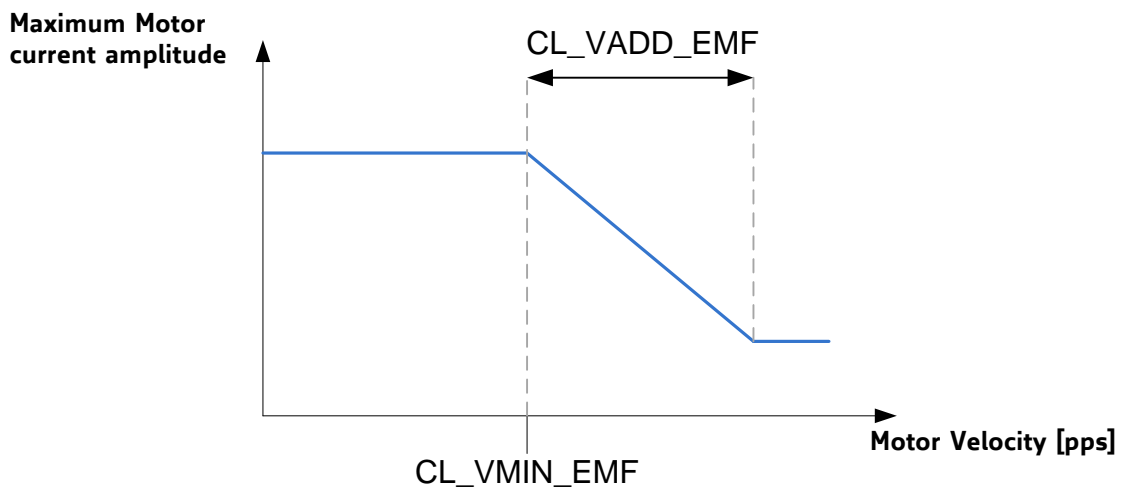


Figure 6.6 Plotting maximum amplitude of the motor current over the motor velocity will also lead to the identification of the velocity limits for the γ -correction

NOTE:

- To get most sufficient result for motor behavior with γ -correction usage, all motor lines should be analyzed for the velocity limits.
- Further on, tweaking the first guess limits slightly can also result in better motor motion behavior.



6.2 Velocity calculation

As it can be seen from Figure 6.2, the trimming is based on the **real motor velocity**. This velocity ENC_VEL [pps] is calculated internally by TMC4361A and will be released at register 0x65. For a correct correction algorithm the encoder velocity have to be filtered (V_ENC_MEAN 0x66 [pps]). The filtering is done by the following equation:

$$V_{ENC_MEAN} = V_{ENC_MEAN} - \frac{V_{ENC_MEAN}}{2^{ENC_VMEAN_FILTER}} + \frac{V_{ENC}}{2^{ENC_VMEAN_FILTER}}$$

Besides the filter exponent, further parameters can be set a register 0x63 for different encoder types to adapt the closed control with back-EMF consideration:

Parameter	Incremental ABN encoder	Absolute SPI/SSI encoder
ENC_VMEAN_WAIT	Delay period [#clock cycles] between two subsequent transfers from V_ENC for the V_ENC_MEAN calculation procedure → sample rate for mean velocity calculation = $1 / ENC_VMEAN_WAIT$	
	ENC_VMEAN_WAIT > 32 8bit at 0x63(7:0)	Automatically set to the assigned encoder update rate SER_PTIME 0x58
ENC_VMEAN_FILTER	Filer exponent for encoder mean velocity calculation 4bit at 0x63(11:8)	
ENC_VMEAN_INT	Maximum update delay period [#clock cycles] for the V_ENC calculation	
	16bit at 0x63(31:16) If $ENC_VMEAN_INT < 256$, it will be set automatically to 256!	Automatically set to the assigned encoder update rate SER_PTIME 0x58
CL_CYCLE	Delay period [#clock cycles] for the closed-loop control → closed-loop control update rate = $1 / CL_CYCLE$	
	Fixed for ABN encoders for a minimum possible delay (mostly 5 clock cycles)	16bit at 0x63(31:16) Set to at least SER_PTIME 0x58
$SER_ENC_VARIATION$	---	8bit at 0x63(7:0) Adaptation of maximum accepted deviation between two consecutive encoder values If set to 0: maximum accepted deviation = $1/8 \cdot ENC_IN_RES$ Else: maximum accepted deviation = $1/8 \cdot SER_ENC_VARIATION/256 \cdot ENC_IN_RES$ serial_enc_variation_limit has to be set to '1' (ENC_IN_RES register 0x07)



NOTE:

- A good starting value for **ENC_VMEAN_WAIT** is 128 and for **ENC_VMEAN_FILTER** is 7. Both values should be adapted in conjunction if ABN encoders are used. Further on, the lower both values are the faster the **V_ENC_MEAN** is adapted to the current velocity. But this also results in higher gradients of the mean velocity which can lead to regulation jumps if the γ correction is enabled.
- To prevent this, check the **V_ENC_MEAN** velocity values during motion transferring the velocity limits **CL_VMIN_EMF** and $(CL_VMIN_EMF + CL_VADD_EMF)$. If the mean encoder velocity is adapted smoothly during motion γ -correction will be also executed properly.

Further on, **ENC_VELO 0x62** assigns the delay in number of clock cycles which sets the encoder velocity **V_ENC** and **V_ENC_MEAN** to 0! If the signals of the AB lines will not switch during this time period or the absolute encoder values will not change, the encoder velocity will be set to 0.

**AREAS OF
SPECIAL
CONCERN**

If the absolute encoder switches its direction the first new encoder velocity will be 0! Thus, if the encoder values are toggling the encoder positions in such a way that direction changes are produced during motion, this can lead to $V_ENC = V_ENC_MEAN = 0$. This will result in an incorrect γ -correction if **CL_VMIN_EMF has been already exceeded.**



7 Examples

In the following complete examples are illustrated by assigning the different registers. **It is advised to check all four examples, even if the setup do not correspond to given application, because slight differences for closed-loop setup are illustrated among the four examples.**

Example 1: Motor driver TMC26x (SPI mode) is used with an incremental ABN encoder; back-emf is not considered; no calibration process

In the following the closed-loop calibration process is omitted after one movement in open loop to the next fullstep position. By setting XACTUAL equal to MSCNT, turning on closed-loop operation is sufficient for calibration purpose if a change of XACTUAL before closed-loop operation is appropriate (see TMC4361A manual section 16.3.2 option 1B). Though, the simplified motion to the next fullstep can lead to torque loss during closed-loop operation if the calibration point is not selected well.

Preliminary considerations:

- Non-differential ABN encoder
- No back-emf consideration
- Catch-up velocity limit = ± 50 kpps
- 400 full steps per revolution
- $I_{RMS} = 1.1$ A, Sense resistor: $R_{SENSE} = 0.15 \Omega$
- $V_{SENSE} = 0 \rightarrow I_{FS} = 2.0$ A
- $I_{PEAK} = I_{RMS} \cdot \sqrt{2} = 1.56$ A $\rightarrow CS = 24$
- Encoder direction is inverted to motor direction

Sequence of register access:

Sequence	Comments	Order $\mu C \rightarrow TMC4361A$
1.	Differential encoder off	0x8000007020
2.	256 microsteps per full step, 400 full steps per revolution	0x8A00001900
3.	Filtering of encoder input signals (Sample rate = $f_{CLK} / 8$, filter length = 4)	0x8300003300
4.	SPI output configuration: Current datagrams for TMC26x (SPI_OUT_BLOCK_TIME / SPI_OUT_HIGH_TIME / SPI_OUT_LOW_TIME = 8/4/4 clock cycles)	0x848440000A
5.	Assign COVERDONE as INTR	0x8D02000000
6.	Clear events	0x0E00000000
7.	Set the DRVCTRL register of TMC26x/389 (cover datagram): single edge steps, disable step interpolation, microstep resolution: 256	0xEC00000000
8.	Wait for an interrupt and then clear events.	...0x0E00000000
9.	Set the CHOPCONF register of TMC26x/389 (cover datagram): tbl=36, standard chopper, HDEC=16, HEND=11, HSTR=1, TOFF=5, RNDTF=off	0xEC00090585
10.	Wait for an interrupt and then clear events.	...0x0E00000000
11.	Disable the SMARTEN register of TMC26x/389 (cover datagram)	0xEC000A0000
12.	Wait for an interrupt and then clear events.	...0x0E00000000
13.	Set the SGCSCONF register of TMC26x/389 (cover datagram): SGT=0, CS=24	0xEC000C0018
14.	Wait for an interrupt and then clear events.	...0x0E00000000



15.	Set the DRVCONF register of TMC26x/389 (cover datagram): PH=3, SLPL=3, DISS2G=off, TS2G=0-3.2us, SDOFF=on, VSENSE=0	0xEC000EF080
16.	Wait for an interrupt and then clear events.	...0x0E00000000
17.	Encoder resolution: 1000 pole pairs → ENC_IN_RES = 4000 (→ ENC_CONST = 25.6)	0xD40000FA0
18.	Invert encoder direction	0x8720000000
19.	CL setup: CL_BETA = 255, CL_GAMMA = 0	0x9C000000FF
20.	CL_DELTA_P = 1.25	0xDC00014000
21.	CL_TOLERANCE = 32 (slightly above ENC_CONST)	0xDF00000020
22.	Proportional term for velocity limitation = 1000	0xDA000003e8
23.	Integral term for velocity limitation = 50	0xDB00000032
24.	Clipping value for catch-up velocity = 50 kpps	0xDE0000C350
25.	Clipping value for integral term = 1000	0xDD000003e8
26.	Disable current scaling	0x8500000000
27.	Hold and positioning mode for full step calibration	0xA000000004
28.	Any velocity (here: 10 kpps) (VMAX has 8 decimal places!)	0xA400271000
29.	Read out MSCNT	0x7900000000
30.	Read out XACTUAL	0x2100000000
31.	Set XTARGET = XACTUAL + 384 - (MSCNT mod 256)	0xB7xxxxxxxx
32.	Wait for until TARGET_REACHED is set again (can also be set as interrupt!)	...
33.	Set VMAX = 0 to prevent unwanted ramp starting	0xA400000000
34.	Read out MSCNT	0x7900000000
35.	Set XACTUAL = MSCNT	0xA1xxxxxxxx
36.	Turn on closed-loop operation	0x8720400000
37.	Turn on velocity limit for closed-loop operation	0x8728400000
38.	Current scale limits: CL_IMAX = 255, CL_IMIN = 100, CL_START_UP = 100	0x860064FF64
39.	CL_UPSCALE = 1000 clock cycles	0x98000003E8
40.	CL_DNSCALE = 100000 clock cycles	0x99000186A0
41.	Enable closed-loop scaling	0x8500000080
42.	CL_TR_TOLERANCE = 60 (if absolute position mismatch is smaller than 3 encoder transitions, TARGET_REACHED can be set)	0xD20000003C
	Set ramp conditions according to the required setup...	



Example 2: Motor driver TMC26x (SPI mode) is used with an incremental differential ABN encoder; advanced calibration process

Preliminary considerations:

- Differential ABN encoder
- Back-emf consideration
- Catch-up velocity limit = ± 50 kpps
- 200 full steps per revolution
- $I_{RMS} = 1.1$ A, Sense resistor: $R_{SENSE} = 0.15 \Omega$
- $V_{SENSE} = 0 \rightarrow I_{FS} = 2.0$ A
- $I_{PEAK} = I_{RMS} \cdot \sqrt{2} = 1.56$ A $\rightarrow CS = 24$

Sequence of register access:

Sequence	Comments	Order $\mu C \rightarrow TMC4361A$
1.	Differential encoder on	0x8000006020
2.	256 microsteps per full step, 200 full steps per revolution	0x8A00000C80
3.	Filtering of encoder input signals (Sample rate = $f_{CLK} / 8$, filter length = 4)	0x8300003300
4.	SPI output configuration: Current datagrams for TMC26x (SPI_OUT_BLOCK_TIME / SPI_OUT_HIGH_TIME / SPI_OUT_LOW_TIME = 8/4/4 clock cycles)	0x848440000A
5.	Assign COVERDONE as INTR	0x8D02000000
6.	Clear events	0x0E00000000
7.	Set the DRVCTRL register of TMC26x/389 (cover datagram): single edge steps, disable step interpolation, microstep resolution: 256	0xEC00000000
8.	Wait for an interrupt and then clear events.	...0x0E00000000
9.	Set the CHOPCONF register of TMC26x/389 (cover datagram): tbl=36, standard chopper, HDEC=16, HEND=11, HSTR=1, TOFF=5, RNDTF=off	0xEC00090585
10.	Wait for an interrupt and then clear events.	...0x0E00000000
11.	Disable the SMARTEN register of TMC26x/389 (cover datagram)	0xEC000A0000
12.	Wait for an interrupt and then clear events.	...0x0E00000000
13.	Set the SGCSCONF register of TMC26x/389 (cover datagram): SGT=0, CS=24	0xEC000C0018
14.	Wait for an interrupt and then clear events.	...0x0E00000000
15.	Set the DRVCONF register of TMC26x/389 (cover datagram): SLPH=3, SLPL=3, DISS2G=off, TS2G=0-3.2us, SDOFF=on, VSENSE=0	0xEC000EF080
16.	Wait for an interrupt and then clear events.	...0x0E00000000
17.	Encoder resolution: 500 pole pairs \rightarrow ENC_IN_RES = 2000 (\rightarrow ENC_CONST = 25.6)	0xD4000007D0
18.	Read out XACTUAL	0x2100000000
19.	Set ENC_POS = XACTUAL	0xD0xxxxxxxx



20.	Hold and positioning mode for full step calibration	0xA000000004
21.	Any velocity (here: 10 kpps) (VMAX has 8 decimal places!)	0xA400271000
22.	Set XTARGET = XACTUAL + 51200	0xB7xxxxxxxx
23.	Wait for until TARGET_REACHED is set again. (can also be set as interrupt!)	...
24.	Read out ENC_POS	0x5000000000
24. a)	If ENC_POS \approx XTARGET (slight variances can occur due to encoder resolution), encoder and motor direction are equal	
24. b1)	If ENC_POS \approx XTARGET - 102400 (slight variances can occur due to encoder resolution), encoder & motor direction are inverted → Invert encoder direction	0x8720000000
24. b2)	Set ENC_POS = XACTUAL	0xD0xxxxxxxx
25.	CL setup: CL_BETA = 255, CL_GAMMA = 255	0x9C00FF00FF
26.	Set CL_DELTA_P = 1.00	0xDC00010000
27.	Set CL_TOLERANCE = 32 (slightly above ENC_CONST)	0xDF00000020
28.	Set CL_TR_TOLERANCE = 60 (if absolute position mismatch is smaller than 3 encoder transitions, TARGET_REACHED can be set)	0xD20000003C
29.	Disable current scaling	0x8500000000
30.	Read out MSCNT	0x7900000000
31.	Read out XACTUAL	0x2100000000
32.	Set XTARGET = XACTUAL + 384 - (MSCNT mod 256) (Move to fullstep position)	0xB7xxxxxxxx
33.	Wait for until TARGET_REACHED is set again. (can also be set as interrupt!)	...
34.	Read out ENC_POS_DEV.	0x5200000000
34. a)	Storage of minimum and maximum encoder deviations: If ENC_POS_DEV < DEV_MIN → DEV_MIN = ENC_POS_DEV If ENC_POS_DEV > DEV_MAX → DEV_MAX = ENC_POS_DEV	
35.	Move to next fullstep position XTARGET = XTARGET + 256	0xB7xxxxxxxx
36.	Repeat steps 33..35 400 times (2 revolutions) It is recommended to also move into the opposite direction.	
37.	Check DEV_SEARCH = (DEV_MAX - DEV_MIN) / 2	
38.	Move to next fullstep position XTARGET = XTARGET - 256	0xB7xxxxxxxx
39.	Wait for until TARGET_REACHED is set again. (can also be set as interrupt!)	...
40.	Seek of the calibration point: If ENC_POS_DEV \approx DEV_SEARCH → CALIB_POS = XTARGET Repeat steps 38..30 400 times at most (2 revolutions) and define CALIB_POS	
41.	Set XTARGET = CALIB_POS	0xB7xxxxxxxx
42.	Wait for until TARGET_REACHED is set again (can also be set as interrupt!)	...



43.	Set VMAX = 0	0xA400000000
44.	Proportional term for velocity limitation = 1000	0xDA000003e8
45.	Integral term for velocity limitation = 50	0xDB00000032
46.	Clipping value for catch-up velocity = 50 kpps	0xDE0000C350
47.	Clipping value for integral term = 1000	0xDD000003e8
48.	Set CL_VMIN_EMF (velocity limits have to be found by open loop movements before) (see chapter 6)	0xE0xxxxxxxx
49.	Set CL_VADD_EMF (velocity limits have to be found by open loop movements before) (see chapter 6)	0xE1xxxxxxxx
50.	Set encoder mean velocity settings (see section 6.2)	0xE3xxxxxxxx
51.	Turn on closed-loop operation and calibration	0x87x1400000
52.	Wait for 10us and then turn off closed-loop calibration	...0x87x0400000
53.	Turn on velocity limit for closed-loop operation	0x87x8400000
54.	Turn on back-emf consideration for closed-loop operation	0x87xA400000
55.	Current scale limits: CL_IMAX = 255, CL_IMIN = 100, CL_START_UP = 100	0x860064FF64
56.	CL_UPSCALE = 200 clock cycles	0x98000007D0
57.	CL_DNSCALE = 100000 clock cycles	0x99000186A0
58.	Enable closed-loop scaling	0x8500000080
59.	Set ramp conditions according to the required setup...	



Example 3: Motor driver TMC2130 (SPI mode) is used with an incremental ABN encoder; back-emf is not considered; simple calibration process

The following closed-loop calibration process is simplified to one movement which can lead to torque loss during closed-loop operation if the calibration point is not selected well.

Preliminary considerations:

- Non-differential ABN encoder
- No back-emf consideration
- Catch-up velocity limit = ± 50 kpps
- 400 full steps per revolution
- $I_{RMS} = 1.1$ A, Sense resistor: $R_{SENSE} = 0.15$ Ω
- $V_{SENSE} = 0 \rightarrow I_{FS} = 1.88$ A
- $I_{PEAK} = I_{RMS} \cdot \sqrt{2} = 1.56$ A $\rightarrow CS = 25$ (\rightarrow Maximum value set in TMC2130)
- Encoder direction is inverted to motor direction

Sequence of register access:

Sequence	Comments	Order $\mu C \rightarrow$ TMC4361A
1.	Differential encoder off	0x8000007020
2.	256 microsteps per full step, 400 full steps per revolution	0x8A00001900
3.	Filtering of encoder input signals (Sample rate = $f_{CLK} / 8$, filter length = 4)	0x8300003300
4.	SPI output configuration: Current datagrams for TMC2130 SPI mode (SPI_OUT_BLOCK_TIME / SPI_OUT_HIGH_TIME / SPI_OUT_LOW_TIME = 8/4/4 clock cycles)	0x848440000D
5.	Assign COVERDONE as INTR	0x8D02000000
6.	Clear events	0x0E00000000
7.	Set the GCONF register of TMC2130 (cover datagrams): direct_mode=1, Set any other switch of this register according to your requirements	0xED00000080 0xEC00010000
8.	Wait for an interrupt and then clear events.	...0x0E00000000
9.	Set the CHOPCONF register of TMC2130 (cover datagrams): 256 microsteps, vsense=0, TBL=36 clock cycles, spread cycle chopper, HEND=1, HSTRT=2, TOFF=3	0xED000000EC 0xEC00010223
10.	Wait for an interrupt and then clear events.	...0x0E00000000
11.	Set the IHOLD_IRUN register of TMC2130 (cover datagrams): IHOLD_DELAY=5=0, IHOLD=IRUN=25	0xED00000090 0xEC00051919
12.	Wait for an interrupt and then clear events.	...0x0E00000000
13.	Set any other register of TMC2130 according to your requirements	0xED000000xx 0xECxxxxxxxx
14.	Wait for an interrupt and then clear events.	...0x0E00000000
15.	Repeat steps 13 and 14 until all relevant registers are set.	



16.	Clear events	0x0E00000000
17.	Encoder resolution: 1000 pole pairs → ENC_IN_RES = 4000 (→ ENC_CONST = 25.6)	0xD400000FA0
18.	Invert encoder direction	0x8720000000
19.	CL setup: CL_BETA = 255, CL_GAMMA = 0	0x9C000000FF
20.	CL_DELTA_P = 1.25	0xDC00014000
21.	CL_TOLERANCE = 32 (slightly above ENC_CONST)	0xDF00000020
22.	Disable current scaling	0x8500000000
23.	Hold and positioning mode for full step calibration	0xA000000004
24.	Any velocity (here: 10 kpps) (VMAX has 8 decimal places!)	0xA400271000
25.	Read out MSCNT	0x7900000000
26.	Read out XACTUAL	0x2100000000
27.	Set XTARGET = XACTUAL + 384 - (MSCNT mod 256)	0xB7xxxxxxxx
28.	Wait for until TARGET_REACHED is set again (can also be set as interrupt!)	...
29.	Set VMAX = 0	0xA400000000
30.	Proportional term for velocity limitation = 1000	0xDA000003e8
31.	Integral term for velocity limitation = 50	0xDB00000032
32.	Clipping value for catch-up velocity = 50 kpps	0xDE0000C350
33.	Clipping value for integral term = 1000	0xDD000003e8
34.	Turn on closed-loop operation and calibration	0x8721400000
35.	Wait for 10us and then turn off closed-loop calibration	...0x8720400000
36.	Turn on velocity limit for closed-loop operation	0x8728400000
37.	Current scale limits: CL_IMAX = 255, CL_IMIN = 100, CL_START_UP = 100	0x860064FF64
38.	CL_UPSCALE = 1000 clock cycles	0x98000003E8
39.	CL_DNSCALE = 100000 clock cycles	0x99000186A0
40.	Enable closed-loop scaling	0x8500000080
41.	CL_TR_TOLERANCE = 60 (if absolute position mismatch is smaller than 3 encoder transitions, TARGET_REACHED can be set)	0xD20000003C
42.	Set ramp conditions according to the required setup...	



Example 4: Motor driver TMC2130 (SD mode) is used with an incremental differential ABN encoder; advanced calibration process

In the following the closed-loop calibration process is omitted after movement in open loop to calibration position. By setting XACTUAL equal to MSCNT, turning on closed-loop operation is sufficient for calibration purpose if a change of XACTUAL before closed-loop operation is appropriate (see TMC4361A manual section 16.3.2 option 1B).

Preliminary considerations:

- Differential ABN encoder
- Back-emf consideration
- Catch-up velocity limit = ± 50 kpps
- 200 full steps per revolution
- $I_{RMS} = 1.1$ A, Sense resistor: $R_{SENSE} = 0.15$ Ω
- $V_{SENSE} = 0 \rightarrow I_{FS} = 1.88$ A
- $I_{PEAK} = I_{RMS} \cdot \sqrt{2} = 1.56$ A $\rightarrow CS = 25$ (\rightarrow Maximum value set in scale register of TMC4361A)

Sequence of register access:

Sequence	Comments	Order $\mu C \rightarrow$ TMC4361A
1.	Differential encoder on	0x8000006020
2.	256 microsteps per full step, 200 full steps per revolution	0x8A00000C80
3.	Filtering of encoder input signals (Sample rate = $f_{CLK} / 8$, filter length = 4)	0x8300003300
4.	SPI output configuration: Current datagrams for TMC2130 SD mode (SPI_OUT_BLOCK_TIME / SPI_OUT_HIGH_TIME / SPI_OUT_LOW_TIME = 8/4/4 clock cycles; scaling values are transferred during motion)	0x848440022C
5.	Assign COVERDONE as INTR	0x8D02000000
6.	Clear events	0x0E00000000
7.	Set the GCONF register of TMC2130 (cover datagrams): direct_mode=0, Set any other switch of this register according to your requirements	0xED00000080 0xEC00000000
8.	Wait for an interrupt and then clear events.	...0x0E00000000
9.	Set the CHOPCONF register of TMC2130 (cover datagrams): 256 microsteps, vsense=0, TBL=36 clock cycles, spread cycle chopper, HEND=1, HSTRT=2, TOFF=3	0xED000000EC 0xEC00010223
10.	Wait for an interrupt and then clear events.	...0x0E00000000
11.	Set the IHOLD_IRUN register of TMC2130 (cover datagrams): IHOLD_DELAY=5=0, IHOLD=10; IRUN=25	0xED00000090 0xEC00050A19
12.	Wait for an interrupt and then clear events.	...0x0E00000000
13.	Set any other register of TMC2130 according to your requirements	0xED000000xx 0xECxxxxxxxx
14.	Wait for an interrupt and then clear events.	...0x0E00000000



15.	Repeat steps 13 and 14 until all relevant registers are set.	
16.	Clear events	0x0E00000000
17.	Encoder resolution: 500 pole pairs → ENC_IN_RES = 2000 (→ ENC_CONST = 25.6)	0xD4000007D0
18.	Read out XACTUAL	0x2100000000
19.	Set ENC_POS = XACTUAL	0xD0xxxxxxxx
20.	Hold and positioning mode for full step calibration	0xA000000004
21.	Any velocity (here: 10 kpps) (VMAX has 8 decimal places!)	0xA400271000
22.	Set XTARGET = XACTUAL + 51200	0xB7xxxxxxxx
23.	Wait for until TARGET_REACHED is set again. (can also be set as interrupt!)	...
24.	Read out ENC_POS	0x5000000000
24. a)	If ENC_POS ≈ XTARGET (slight variances can occur due to encoder resolution), encoder and motor direction are equal	
24. b1)	If ENC_POS ≈ XTARGET - 102400 (slight variances can occur due to encoder resolution), encoder & motor direction are inverted → Invert encoder direction	0x8720000000
24. b2)	Set ENC_POS = XACTUAL	0xD0xxxxxxxx
25.	CL setup: CL_BETA = 255, CL_GAMMA = 255	0x9C00FF00FF
26.	Set CL_DELTA_P = 1.00	0xDC00010000
27.	Set CL_TOLERANCE = 32 (slightly above ENC_CONST)	0xDF00000020
28.	Set CL_TR_TOLERANCE = 60 (if absolute position mismatch is smaller than 3 encoder transitions, TARGET_REACHED can be set)	0xD20000003C
29.	Disable current scaling	0x8500000000
30.	Read out MSCNT	0x7900000000
31.	Read out XACTUAL	0x2100000000
32.	Set XTARGET = XACTUAL + 384 - (MSCNT mod 256) (Move to fullstep position)	0xB7xxxxxxxx
33.	Wait for until TARGET_REACHED is set again. (can also be set as interrupt!)	...
34.	Read out ENC_POS_DEV.	0x5200000000
34. a)	Storage of minimum and maximum encoder deviations: If ENC_POS_DEV < DEV_MIN → DEV_MIN = ENC_POS_DEV If ENC_POS_DEV > DEV_MAX → DEV_MAX = ENC_POS_DEV	
35.	Move to next fullstep position XTARGET = XTARGET + 256	0xB7xxxxxxxx
36.	Repeat steps 33..35 400 times (2 revolutions) It is recommended to also move into the opposite direction.	
37.	Check DEV_SEARCH = (DEV_MAX - DEV_MIN) / 2	
38.	Move to next fullstep position XTARGET = XTARGET - 256	0xB7xxxxxxxx



39.	Wait for until TARGET_REACHED is set again. (can also be set as interrupt!)	...
40.	Seek of the calibration point: If $ENC_POS_DEV \approx DEV_SEARCH$ → $CALIB_POS = XTARGET$ Repeat steps 38..30 400 times at most (2 revolutions) and define CALIB_POS	
41.	Set $XTARGET = CALIB_POS$	0xB7xxxxxxxx
42.	Wait for until TARGET_REACHED is set again (can also be set as interrupt!)	...
43.	Set $VMAX = 0$	0xA400000000
44.	Proportional term for velocity limitation = 1000	0xDA000003e8
45.	Integral term for velocity limitation = 50	0xDB00000032
46.	Clipping value for catch-up velocity = 50 kpps	0xDE0000C350
47.	Clipping value for integral term = 1000	0xDD000003e8
48.	Set CL_VMIN_EMF (velocity limits have to be found by open loop movements before) (see chapter 6)	0xE0xxxxxxxx
49.	Set CL_VADD_EMF (velocity limits have to be found by open loop movements before) (see chapter 6)	0xE1xxxxxxxx
50.	Set encoder mean velocity settings (see section 6.2)	0xE3xxxxxxxx
51.	Read out MSCNT	0x7900000000
52.	Set $XACTUAL = MSCNT$	0xA1xxxxxxxx
53.	Turn on closed-loop operation	0x87x0400000
54.	Turn on velocity limit for closed-loop operation	0x87x8400000
55.	Turn on back-emf consideration for closed-loop operation	0x87xA400000
56.	Current scale limits: $CL_IMAX = 25$, $CL_IMIN = 10$, $CL_START_UP = 50$	0x860032190A
57.	$CL_UPSCALE = 200$ clock cycles	0x98000007D0
58.	$CL_DNSCALE = 100000$ clock cycles	0x99000186A0
59.	Enable closed-loop scaling	0x8500000080
60.	Set ramp conditions according to the required setup...	



8 Revision History

Version	Date	Author	Description
0.5	2014-Aug-11	HS	Initial version
0.6	2014-Aug-26	HS	Clarifications Reference to CL_MAX und CL_FIT events and flag Initial back emf notes
0.9	2014-Nov-21	HS	Clarifications Enc Compensation added Back-EMF-consideration written out in full (scope shots and settings for VMIN_EMF and VADD_EMF are still missing and in progress)
0.91	2015-Feb-06	HS	Hints added in chapter 5 Revision of all chapters
0.92	2015-Apr-15	HS	90° = 255 microsteps
0.93	2015-Apr-17	HS	Review of example 1, example2 added
0.93	2015-Nov-10	HS	MSTEP_PER_FS MUST be 256!, added
0.99	2016-Jun-03	HS	Sections considering TMC2130 added
0.991	2017-Jul-21	HS	Advanced calibration process in examples: Step40 "If ENC_POS_DEV = DEV_SEARCH → CALIB_POS = XTARGET" Instead of "If ENC_POS_DEV = DEV_MIN → CALIB_POS = XTARGET"
1.0	2020-Mar-10	HS	Final version, complete overhaul: several adaptations

Table 1 Document Revisions

