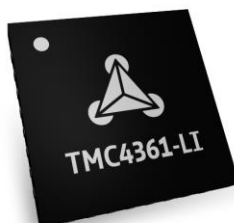




# Application note: Closed Loop motor control with TMC4361 for stepper motor drivers

+

+

*Closed-Loop*

TRINAMIC® Motion Control GmbH & Co. KG  
GERMANY

[www.trinamic.com](http://www.trinamic.com)

This application note explains functional details of closed-loop motor control of stepper motor (gate) drivers in combination with the motion control IC TMC4361. It serves as technical background information.

Please refer to the specific product documentation for specific parameter information and configuration instructions.

## TABLE OF 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 TMC4361 WITH $\mu$ 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 TMC4361 REGISTER SETUP HINTS .....	5
2.4	SAMPLE SETUPS .....	6
3	BASIC SETUP FOR CLOSED LOOP CONTROL .....	9
3.1	GENERAL ENCODER SELECTION AND FILTERING .....	9
3.2	SETUP SPI OUTPUT CONFIGURATION FOR TMC MOTOR (GATE) DRIVERS .....	10
3.3	SENDING COVER DATAGRAMS TO SETUP TMC26x/389 .....	10
3.4	ENCODER SETUP .....	12
3.5	CLOSED LOOP SETUP .....	16
3.6	CLOSED LOOP CALIBRATION .....	17
3.7	LIMITING THE MAXIMUM CORRECTION VELOCITY .....	18
3.8	CLOSED LOOP OPERATION DURING RAMP POSITIONING MODE .....	18
3.9	CLOSED LOOP VELOCITY MODE .....	18
4	CURRENT SCALING DURING CLOSED LOOP MOTOR CONTROL .....	19
4.1	EXAMPLE FOR A COMBINED USAGE OF CLOSED LOOP ANGLE TRACKING AND CURRENT SCALING .....	21
5	CONSIDERING THE BACK-EMF .....	23
5.1	VELOCITY SETUP .....	24
5.2	VELOCITY CALCULATION .....	26
6	EXAMPLES .....	28
7	REVISION HISTORY .....	38
7.1	DOCUMENT REVISIONS .....	38

# 1 Closed Loop Motor Control

TMC4361 provides closed-loop motor control capabilities for stepper motor (gate) drivers. Typically, current values or step impulses from TMC4361 for the motor drivers are based only on internal calculations. With the closed loop unit of the TMC4361 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 the velocity will be controlled, finally the position. The current control loop assigns always a current which holds a commutation angle of  $90^\circ$  to gain maximum torque. Feedback will be 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 TMC4361 follows a different approach than PID control cascades to consider stepper motor driver characteristics. The ramp generator which assigns target and velocity is independent on the position control (commutation angle control) which is also independent on the current control. The closed loop motor control scheme is depicted in the following picture.

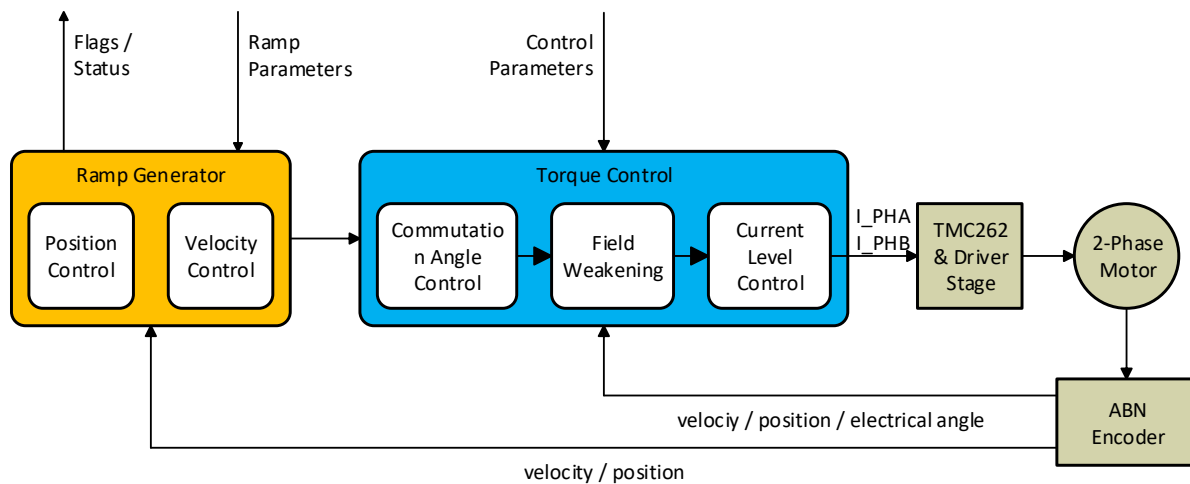


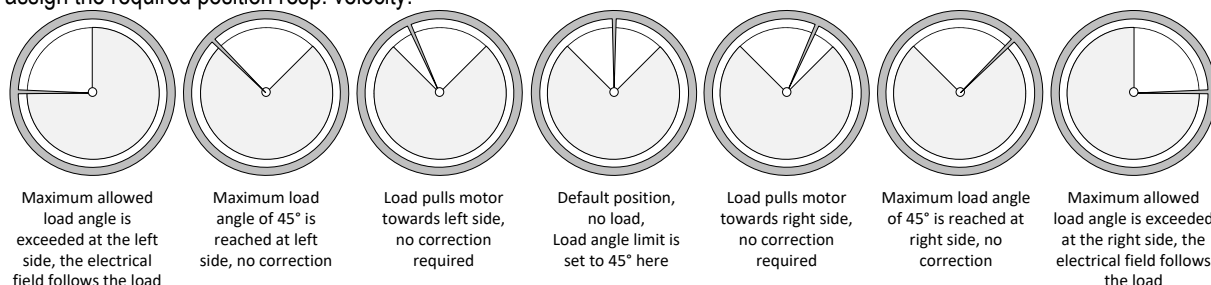
Figure 1.1 Structure of the closed loop motor control

**ATTENTION** The encoder should be mounted directly on the motor axis for closed loop operation to prevent unwanted coupling effects. Further on, if the encoder is not mounted directly on the motor, the PID regulation feature should be used due to the fact that e.g. slipping of belt drives can not be handled correctly with the closed loop motor control.

## 1.2 Closed Loop motor control prevents step loss

As typical for stepper motor drivers, phase currents will be assigned directly to the 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.

Further on, the direction of the current vector will track the rotor position if the load angle should impend to exceed a certain limit. The result is a load angle which will be never exceed the given limit and as a result no step loss will occur. Thus, the current vector will follow an overpowered load until the load is reduced. After overcoming the overload, the current control will assign the required position resp. velocity.



**Figure 1.2 Load angle control and current synchronization if the load angle limit (here 45°) is exceeded**

**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 TMC4361 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 die setup have to be connected correctly. In the next chapter 2 the possible connections between  $\mu C$ , TMC4361, the motor, the encoder and the motor (gate) drivers will be introduced. Further information about the correct pinning and layout considerations of the different devices can be found in the data sheets of the particular components. Please note also that the correct SPI and encoder communication schemes for TMC4361 can be find in corresponding manual where it will explained in detail.

## 2 Connecting TMC4361 with $\mu$ C, motor (gate) driver and encoder

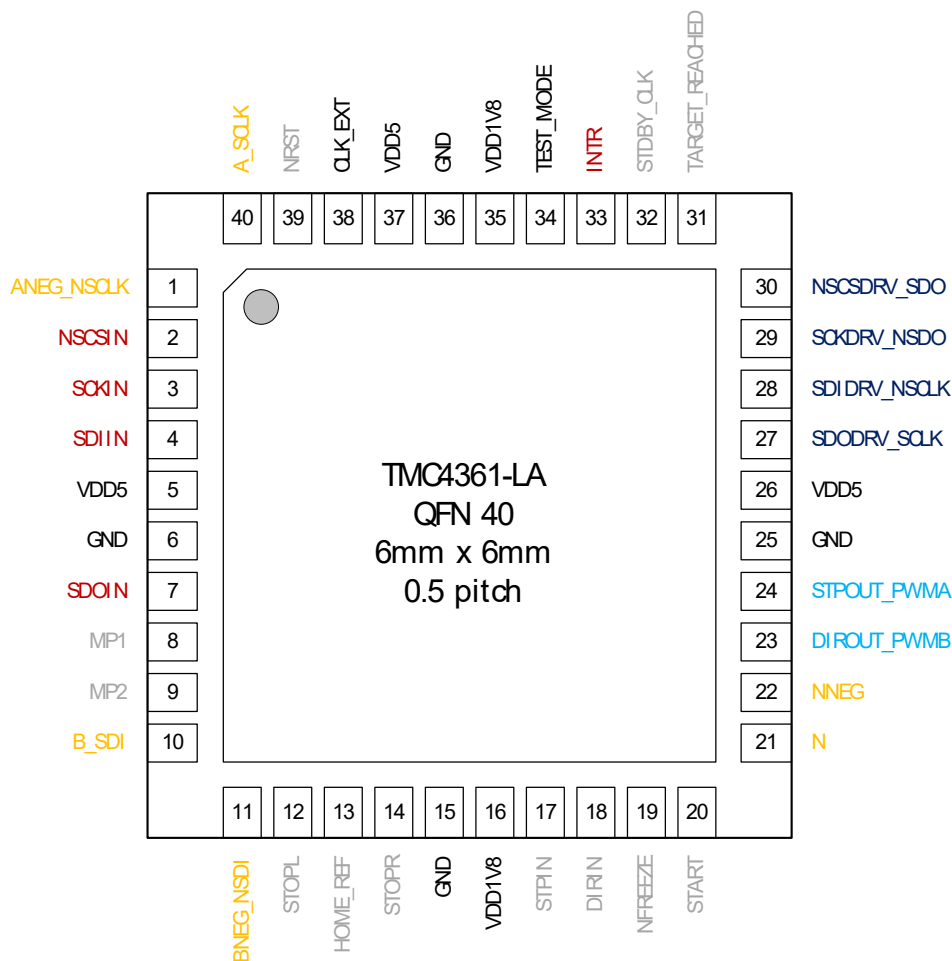


Figure 2.1 Pinning (top view)

### 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 <b>internal generated</b> core voltage of 1.8V
CLK_EXT	38	<b>External Clock</b> input to provide an clock with the frequency $f_{CLK}$ for all internal operations.
TEST_MODE	34	Test mode input. <b>Tie to low</b> 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 $\mu$ C.
A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NS DI, N, NNEG	40, 1, 10, 11, 21, 22	Encoder interface for incremental ABN encoders and absolute SSI and SPI encoder types.
STPOUT_PWMA, DIROUT_PWM B	24, 23	Step Direction output for motor stepper drivers
NSCSDRV_SDO, SCKDRV_NS DO, SDIDRV_NSCLK, SDODRV_SCLK	30, 29, 28, 27	SPI interface to Trinamic Motor drivers.

In the following the required pin signals of TMC4361 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 TMC4361 manual.

## 2.2 Connecting the particular pins for a minimalist setup

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  $\mu$ 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  $\mu$ C to TMC4361. Thus, connect NSCSIN, SCKIN, SDIIN and SDOIN with the  $\mu$ C (please refer to the TMC4361 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 TMC4361 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

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

## 2.3 General TMC4361 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 TMC4361 manual.

Further encoder settings will be available with the **ENC\_IN\_CONF register 0x07** which will be explained in detail in the next chapters.

## 2.4 Sample Setups

In the following examples, dashed lines defines optional wires.

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

The TMC26x can be configured and driven completely via the SPI output interface of the TMC4361. The configuration values for the five registers of the TMC26x have to be addressed to the cover register 0x6C of TMC4361. It will sent automatically after writing to this register. Thus, no connection from  $\mu$ C to the TMC motor driver is required because the communication between both devices can be tunneled through TMC4361.

If SPI output will be only used for configuration the StepDir output pins of TMC4361 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.

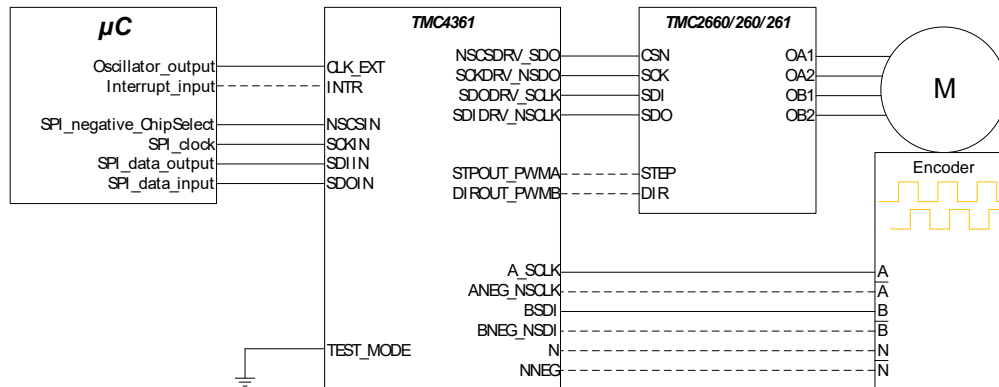


Figure 2.2 Example 1: Connections between  $\mu$ C, TMC4361, TMC26x, motor M and an incremental encoder which is connected directly to the motor M

### Example 2: Motor driver TMC262 and an absolute SSI encoder

This setup depicts the direct configuration of the stepper driver by the  $\mu$ C. Anyhow, the TMC262 can be also configured and driven completely via the SPI output interface of the TMC4361 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.

**ATTENTION** TMC4361 supports absolute SSI encoders which provides multturn as well as singleturn data. In the latter case, calculation of multturn data have to be turned on internally (TMC4361) for the most closed loop motor control applications.

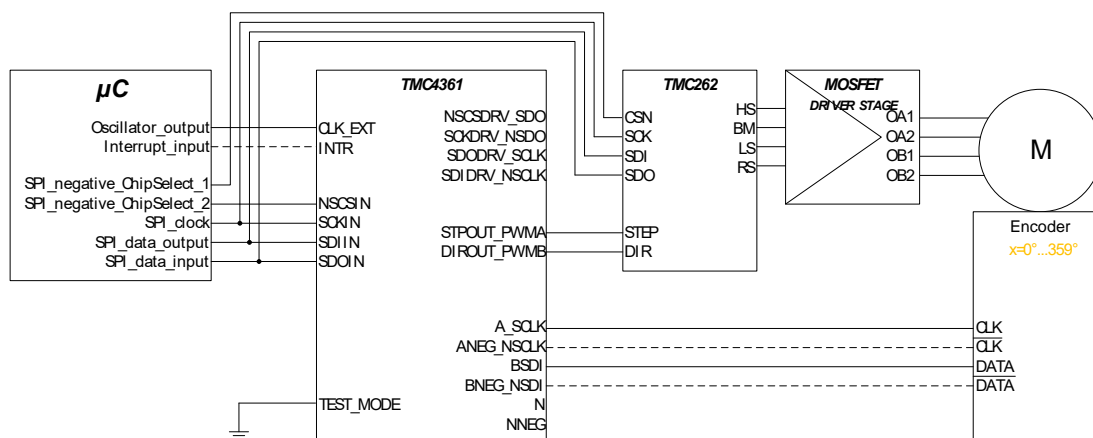


Figure 2.3 Example 2: Connections between  $\mu$ C, TMC4361, TMC262, motor M and an absolute SSI encoder which is connected directly to the motor M

**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.

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

This setup covers the connection between TMC4361 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.

**ATTENTION** TMC4361 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.

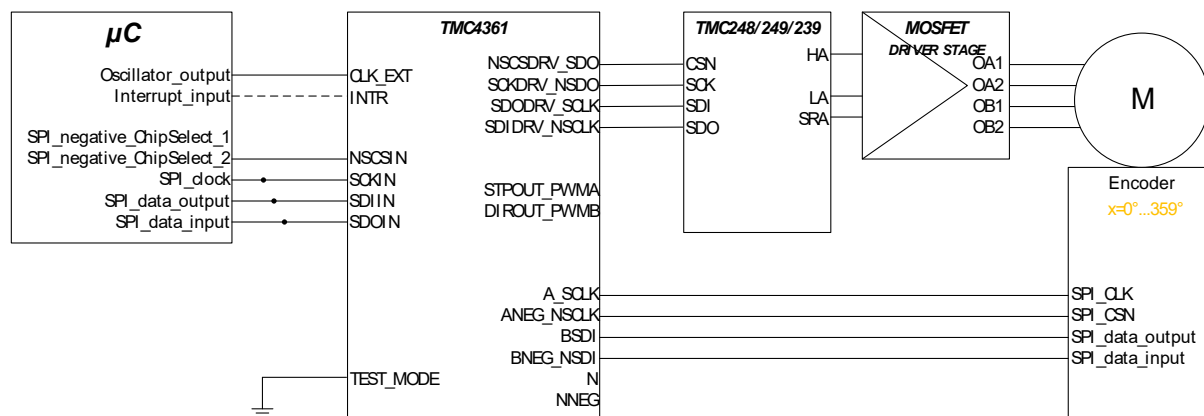


Figure 2.4 Example 3: Connections between  $\mu$ C, TMC4361, TMC248/239/249, motor M and an absolute SPI encoder which is connected directly to the motor M. If TMC249/239 is used, no MOSFET driver stage is necessary

### Example 4: Motor driver TMC2130 and an incremental ABN encoder

The TMC2130 can be configured and driven completely via the SPI output interface of the TMC4361. The configuration address values for the registers of the TMC2130 have to be addressed to the cover register 0x6D of TMC4361, 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  $\mu$ C to the TMC motor driver is required because the communication between both devices can be tunneled through TMC4361.

If SPI output will be only used for configuration the StepDir output pins of TMC4361 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.

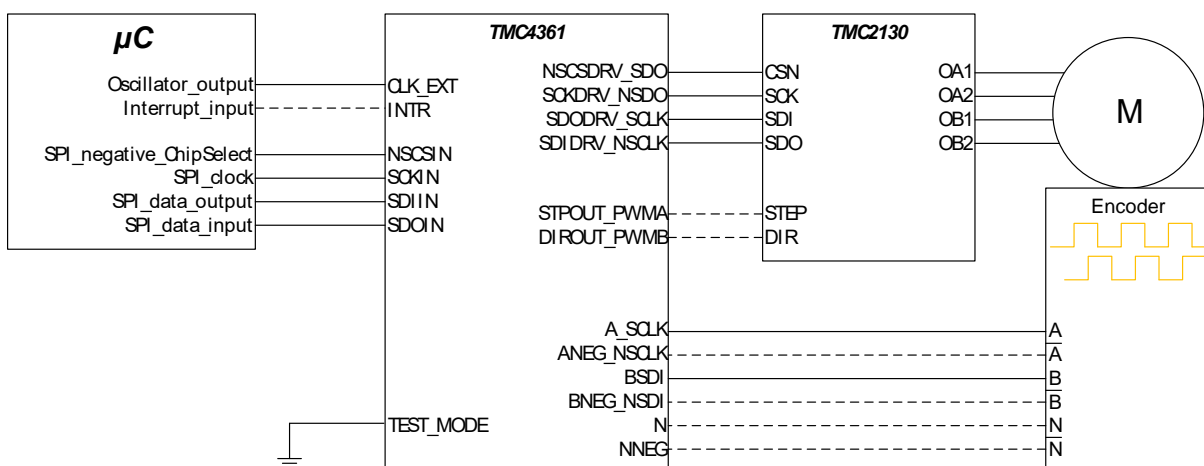


Figure 2.5 Example 4: Connections between  $\mu$ C, TMC4361, TMC2130, motor M and an incremental ABN encoder which is connected directly to the motor M.

### Example 5: Any motor driver and an incremental ABN encoder

Here, TMC4361 is connected with any motor driver. The configuration have to be done via the  $\mu$ C 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 TMC4361. In most cases, step direction output is required. Only if DAC are connected directly to the TMC4361 SPI output current datagrams can be used for motor motion. Please refer to TMC4361 for further information.

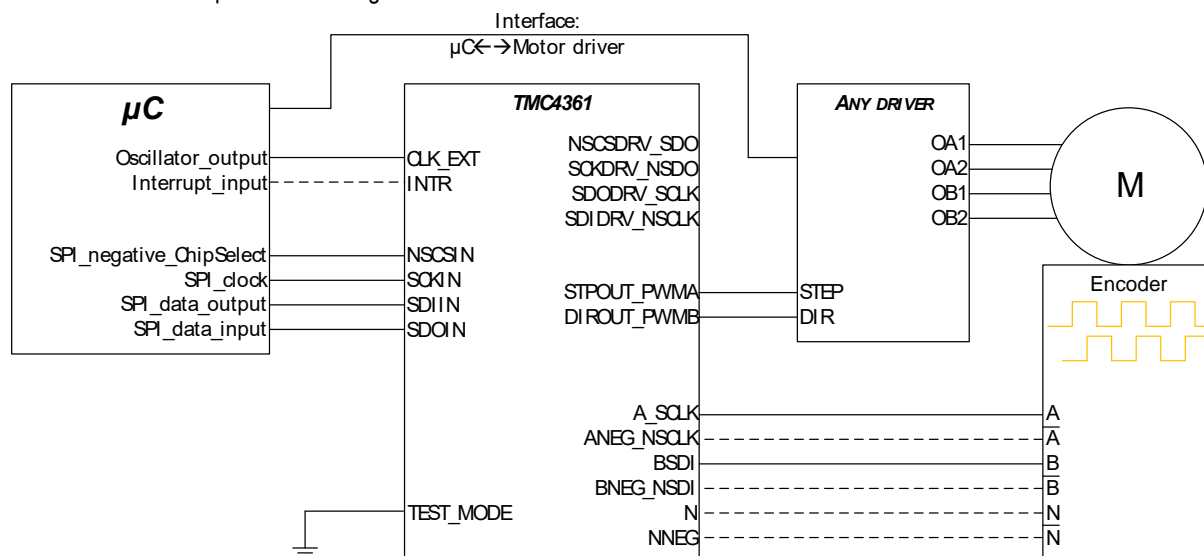


Figure 2.6 Example 5: Connections between μC, TMC4361, any motor driver stage, motor M and an incremental ABN encoder which is connected directly to the motor M.



### 3 Basic setup for closed loop control

In the following the basic setup for TMC4361 closed loop operation will be explicated in detail (please keep the introduced sequence of datagrams to avoid unwanted behavior). Ensuing datagrams (from the  $\mu$ C to TMC4361) 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 TMC4361 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 && 0xFFFFBFFF || 0x8000002000.
GENERAL_CONF (example)=      0x0012345678
&& 0xFFFFBFFF                →      0x0012305678
|| 0x8000002000                →      0x8012307678
```

Now, 0x8012307678 can be sent to TMC4361 and the particular switches will be set.

#### 3.1 General encoder selection and filtering

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 register **0x00**. **Per default, differential inputs are enabled, except for SPI encoders!**

Following possibilities are practicable:

Encoder type	Differential output	Order $\mu$ C→TMC4361
ABN encoder	✓	GENERAL_CONF && 0x00FFFFE3FF    0x8000000000
	-	GENERAL_CONF && 0x00FFFFE3FF    0x8000001000
SSI encoder	✓	GENERAL_CONF && 0x00FFFFE3FF    0x8000000400
	-	GENERAL_CONF && 0x00FFFFE3FF    0x8000001400
SPI encoder	-	GENERAL_CONF && 0x00FFFFF3FF    0x8000000C00

**ATTENTION** 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 TMC4361. 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 TMC4361 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 TMC4361 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**.

**HINT** 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 TMC4361 manual for further information.

**ATTENTION** The microstep per fullsteps **MSTEP\_PER\_FS** **MUST** be set to **256** in register **0x0A** in the case of closed loop control.

### 3.2 Setup SPI output configuration for TMC motor (gate) drivers

If current datagrams should be sent from TMC4361 to TMC motor (gate) drivers and/or configuration datagrams from the  $\mu\text{C}$  through TMC4361 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 TMC4361. 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 information about the timing, please refer to TMC4361 manual.

TMC motor (gate) driver	Using only StepDir for motion control	Order $\mu\text{C} \rightarrow \text{TMC4361}$
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

### 3.3 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 TMC4361. The mechanism will be explained with the help of TMC26x/389. Thereby, five registers have to be set to put the driver into operation. 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**) 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 sent 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 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 assign automatically to the status flag register **0x0F** of TMC4361.

In the following the sequence of datagrams from  $\mu\text{C}$  to TMC4361 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 $\mu\text{C} \rightarrow \text{TMC4361}$
1.	Assign COVERDONE as INTR	0x8D02000000
2.	Clear events	0x0E00000000
3.	Set the DRVCTRL register of TMC26x/389 (cover datagram): single edge steps, disable step interpolation, microstep resolution: 256	0xEC00000000
4.	Wait for an interrupt and then clear events.	...0x0E00000000

5.	Set the CHOPCONF register of TMC26x/389 (cover datagram): tbl=36, standard chopper, HDEC=16, HEND=11, HSTR=1, TOFF=5, RNDTF=off	0xEC00090585
6.	Wait for an interrupt and then clear events.	...0x0E00000000
7.	Disable the SMARTEN register of TMC26x/389 (cover datagram)	0xEC000A0000
8.	Wait for an interrupt and then clear events.	...0x0E00000000
9.	Set the SGSCONF register of TMC26x/389 (cover datagram): SGT=0, CS=31 ( <b>maximum value!</b> )	0xEC000C001F*
10.	Wait for an interrupt and then clear events.	...0x0E00000000
11. a)	Set the DRVCONF register of TMC26x/389 (cover datagram): SLPH=3, SLPL=3, DISS2G=off, TS2G=0-3.2us, SDOFF=on, VSENSE=0* - Use this configuration if TMC26x/389 should evaluate current datagrams and not StepDir input signals) - TMC4361 SPI_OUT CONF register has to be set accordingly	0xEC000EF080*
11. b)	Set the DRVCONF register of TMC26x/389 (cover datagram): SLPH=3, SLPL=3, DISS2G=off, TS2G=0-3.2us, SDOFF=off, VSENSE=0* - Use this configuration if TMC26x/389 should evaluate StepDir input signals and not current datagrams - TMC4361 SPI_OUT CONF register has to be set accordingly	0xEC000EF000*
12.	Wait for an interrupt and then clear events.	...0x0E00000000

**\* Setting appropriate current settings:**

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  $I_{RMS} = 0.85\text{ A}$ , Sense resistor:  $R_{SENSE} = 0.22\ \Omega$

Settings:  $V_{SENSE} = 0 \rightarrow$  Sense resistor voltage  $V_{FS} = 305\text{ mV}$  (FS - Full scale)

$\rightarrow$  Maximum possible current  $I_{FS} = V_{FS} / R_{SENSE} = 1.39\text{ A}$

$I_{PEAK} = I_{RMS} \cdot \sqrt{2} = 1.2\text{ A} \rightarrow CS = I_{PEAK} / I_{FS} \cdot 32 - 1 = 26$

Cover datagram at sequence no. 9: **0xEC000C001A** and

Cover datagram at sequence no. 11a): **0xEC000EF080** or

Cover datagram at sequence no. 11b): **0xEC000EF000**

Example 2:

Conditions: Motor current  $I_{RMS} = 0.5$ , Sense resistor:  $R_{SENSE} = 0.15\ \Omega$

Settings:  $V_{SENSE} = 1 \rightarrow$  Sense resistor voltage  $V_{FS} = 165\text{ mV}$

$\rightarrow$  Maximum possible current  $I_{FS} = V_{FS} / R_{SENSE} = 1.1\text{ A}$

$I_{PEAK} = I_{RMS} \cdot \sqrt{2} = 0.71\text{ A} \rightarrow CS = I_{PEAK} / I_{FS} \cdot 32 - 1 = 19$

Cover datagram at sequence no. 9: **0xEC000C0013** and

Cover datagram at sequence no. 11a): **0xEC000EF0C0** or

Cover datagram at sequence no. 11b): **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!

### 3.4 Example configuration for TMC2130

In the following the sequence of cover datagrams from  $\mu\text{C}$  to TMC4361 will be displayed with an example for a TMC2130 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 TMC2130. 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 TMC2130 should be adapted to the actual board setup. Please refer the TMC2130 manual for detailed information.

Sequence	Comments	Order $\mu\text{C} \rightarrow \text{TMC4361}$
1.	Assign COVERDONE as INTR	0x8D02000000
2.	Clear events	0x0E00000000
3. a)	Set the GCONF 0x00 register of TMC2130 (cover datagrams): direct_mode=0 $\rightarrow$ StepDir mode on - Use this configuration if TMC26x/389 should evaluate current datagrams and not StepDir input signals) - TMC4361 SPI_OUT CONF register has to be set accordingly	0xED00000080 0xEC00000000
3. b)	Set the GCONF 0x00 register of TMC2130 (cover datagrams): direct_mode=1 $\rightarrow$ StepDir mode off - Use this configuration if TMC26x/389 should evaluate current datagrams and not StepDir input signals) - TMC4361 SPI_OUT CONF register has to be set accordingly	0xED00000080 0xEC00010000
4.	Wait for an interrupt and then clear events.	...0x0E00000000
5.	Set the CHOPCONF register 0x6C of TMC2130 (cover datagrams): 256 microsteps, vsense=0, TBL=36 clock cycles, spread cycle chopper, HEND=1, HSTRT=2, TOFF=3	0xED000000EC 0xEC00010223**
6.	Wait for an interrupt and then clear events.	...0x0E00000000
7. a)	Set the IHOLD_IRUN register 0x10 of TMC2130 (cover datagram): IHOLD_DELAY=5, IHOLD=31, IRUN=31 - SPI mode (Step Dir mode off): Set IHOLD to maximum as all incoming current datagrams are scaled by this value	0xED00000090 0xEC00051F1F**
7. b)	Set the IHOLD_IRUN register 0x10 of TMC2130 (cover datagram): IHOLD_DELAY=5, IHOLD=15, IRUN=31 (maximum value!)	0xED00000090 0xEC00050F1F**
8.	Wait for an interrupt and then clear events.	...0x0E00000000

#### \*\* Setting appropriate current settings:

The current settings are dependent on the used motor type. In the following two examples are explained, how to set correct current settings for IRUN and/or IHOLD (current scaling) and VSENSE of TMC2130. 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  $I_{\text{RMS}} = 0.85 \text{ A}$ , Sense resistor:  $R_{\text{SENSE}} = 0.22 \Omega$   
Settings: VSENSE = 0  $\rightarrow$  Sense resistor voltage  $V_{\text{FS}} = 320 \text{ mV}$  (FS - Full scale)  
 $\rightarrow$  Maximum possible current  $I_{\text{FS}} = V_{\text{FS}} / (R_{\text{SENSE}} + 20 \text{ m}\Omega) = 1.33 \text{ A}$   
 $I_{\text{PEAK}} = I_{\text{RMS}} \cdot \sqrt{2} = 1.2 \text{ A} \rightarrow \text{CS} = I_{\text{PEAK}} / I_{\text{FS}} \cdot 32 - 1 = 27$

Cover datagrams at sequence no. 5: set vsense=0  
 Cover datagrams at sequence no. 7a): **0xEC00051B1B** or  
 Cover datagrams at sequence no. 7b): **0xEC0005xx1B**

## 3.5 Encoder setup

Before setting the encoder resolution, the correct encoder setup has to be adapted.

### Incremental ABN encoder

The N event configuration can be adapted in the **ENC\_IN\_CONF** register **0x07**: bits11:2 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.

### Absolute encoder

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 TMC4361. Thus, set **bit16** (calc\_multi\_turn\_behav) of register **0x07** to '1'. This way, TMC4361 will add or remove one revolution if an overflow for the singleturn data is recognized. **This will lead to reliable results as long as the difference between the values of consecutive data transfers do not exceed the half of one revolution!**

**HINT** To restrict TMC4361 to take over valid data from serial encoder data transfer, bit31 (serial\_enc\_variation\_limit) of **ENC\_IN\_CONF** register **0x07** can be set to '1'. That way, a difference of **more than one eighth of the revolution** between consecutive data (due to erroneous transfer or due to a slow data transfer related to the motor velocity) will lead to a rejection of the last data.

### 3.5.1 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, in the register the microsteps per full **MSTEP\_PER\_FS** steps can be adapted. For best performance, it is recommended not to change the default 256 micro steps per full step.

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, TMC4361 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. In the following, 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 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 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:

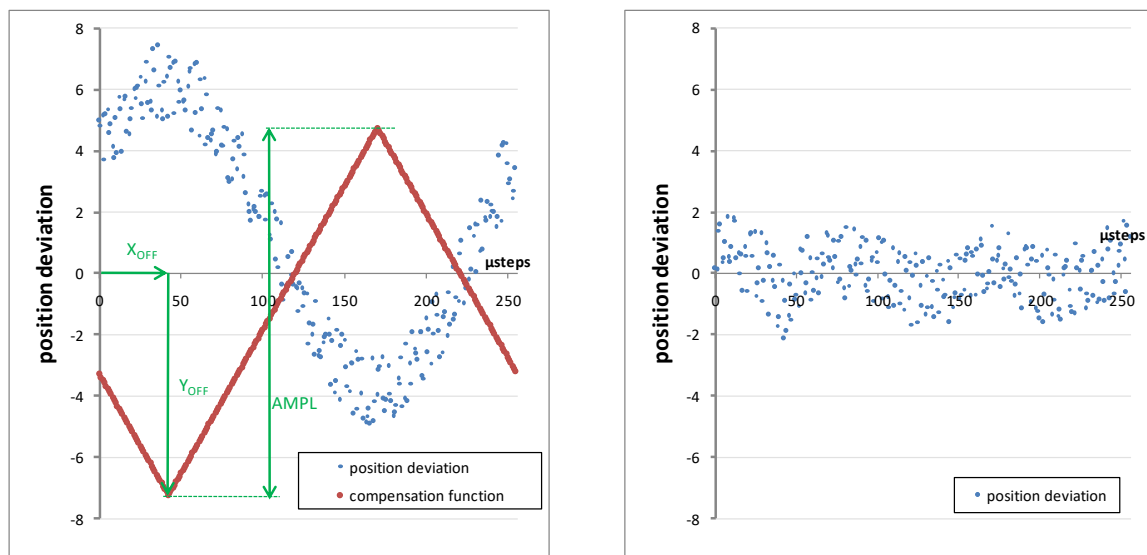
1. Invert the encoder direction by setting **bit29** of ENC\_IN\_CONF register **0x07** or
2. Invert the internal motor direction by setting **bit28** of GENERAL\_CONF register **0x00** or

3. Interchange the motor cables or
4. Remount the encoder.

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

### 3.5.2 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  $\mu$ 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. TMC4361 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!**

**ATTENTION** 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.

**ATTENTION** The abscissa value of the maximum of the triangular function is defined a **half revolution distant to the minimum!**

**HINT** 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.

**HINT** 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  $2^{16}$** .

*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.)

**HINT** 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!

### 3.6 Closed Loop setup

After setting the TMC4361 encoder properties properly and before calibrating the closed loop operation, closed loop properties should be setup accordingly. 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 TMC4361 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.

**HINT** 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 3.2). 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
<code>CL_BETA =</code>	255 (0x0FF)	200 (0x0C8)	275 (0x113)
<code>CL_DELTA_P =</code>	1.5 (0x018000)	0.9375 (0x00F000)	2.75 (0x02C000)
<u>Position mismatch</u> (microsteps/angle)	<u>Commutation angle</u> (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°

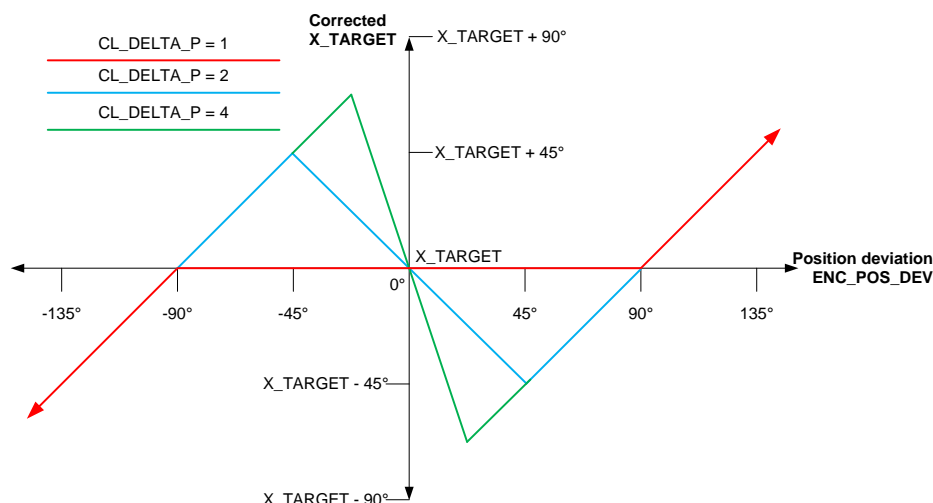


Figure 3.2 Calculation of the corrected target position with `CL_BETA = 255 (90°)` and 256 microsteps per full step

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.

**HINT** 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.



### 3.7 Closed Loop calibration

Now, the basic setup is ready to drive in closed loop motor control modus. Thus, 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.

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

1. Disable any current scaling.
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 TMC4361 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.

**ATTENTION** 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 TMC4361 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.

**ATTENTION** Writing the closed loop offset **CL\_OFFSET** register **0x59** during closed loop operation should be avoided under any circumstances except for calibration purpose.

**ATTENTION** 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.

**HINT** 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 be 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.

**ATTENTION** 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!

### 3.8 Limiting the maximum correction velocity

After turning on closed loop motor control and calibrating, TMC4361 will always try to counter position mismatches. If no other settings are made, the correction will be assigned as fast as possible. To limit the catch-up velocity, the internal PI regulator of TMC4361 can be used. Try to assign different values for the following parameters to fit for the best performance of the closed loop velocity limitation:

- The proportional term **CL\_VMAX\_CALC\_P** register **0x5A** and the integral term **CL\_VMAX\_CALC\_I** register **0x5B** define the **PI regulator** for the **maximum velocity limitation**. The PID regulator will be updated every 128<sup>th</sup> clock cycle (update frequency =  $f_{CLK}/128$ ). The integral term is a product of  $CL\_VMAX\_CALC\_I / 256 \cdot PID\_ISUM / 256$ . The current integrator sum  $PID\_ISUM$  can be read out at register 0x5B.
- By setting the value of the **PID\_DV\_CLIP** register **0x5E**, the maximum velocity value (pps - pulses per second, no decimal places!) which will be added/subtracted (in dependence of the internal direction) to the current internal velocity  $V_{ACTUAL}$  will be defined. For instance, if  $PID\_DV\_CLIP$  is set to 20000 pps (0x00004E20) and  $V_{ACTUAL} = 100000.0$  pps, the resulting maximum catch-up velocity will not exceed 120000 pps. Hence, setting  $PID\_DV\_CLIP$  to 0 will result in a static behavior where no catch-up is possible.
- It is also possible to set the value of the **PID\_I\_CLIP** register **0x5D(14:0)** which is the clipping parameter for  $PID\_ISUM$  the real clipping value  $PID\_ISUM_{CLIPPED} = PID\_ISUM \cdot 2^{16} / PID\_I\_CLIP$ . The maximum value of  $PID\_I\_CLIP$  should meet the condition  $PID\_I\_CLIP \leq PID\_DV\_CLIP / PID\_I$ . If the error sum  $PID\_ISUM$  is not clipped, it is increased with each time step by  $PID\_I \cdot PID\_E$ . This continues as long as the motor does not follow.

If the values are set properly, velocity limitation can be turned on by activating (=1) the **cl\_vlimit\_en** switch of the **ENC\_IN\_CONF** register **0x07**:

```
ENC_IN_CONF && 0x00FE7FFFFFFF || 0x8708400000.
```

In case position deviation at the end of an internal ramp calculation is still left, the SPI and/or Step/Dir output ramp for correction is a linear deceleration ramp, independently from the preset ramp type. This final ramp for error compensation is a function of  $ENC\_POS\_DEV$  and PI control parameters ( $CL\_VMAX\_CALC\_P$ ,  $CL\_VMAX\_CALC\_I$ ,  $PID\_I\_CLIP$ , and  $PID\_DV\_CLIP$ ).

### 3.9 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.

### 3.10 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,  $X_{ACTUAL}$  will be manipulated if the position difference  $ENC\_POS\_DEV$  exceeds 768 microsteps. If so,  $X_{ACTUAL}$  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 TMC4361 will always try to match the given velocity  $V_{MAX}$ .

Further on, if the maximum velocity  $V_{MAX}$  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  $V_{MAX}$ .

**ATTENTION** 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  $X_{ACTUAL}$ , it is possible that the internal ramp can miss the end point of the internal positioning ramp.

## 4 Current scaling during closed loop motor control

The TMC4361 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:

$$\text{real\_scale\_value} = (x + 1) / 256.$$

**ATTENTION** If TMC26x or TMC21xx motor (gate) driver are used in step direction 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:

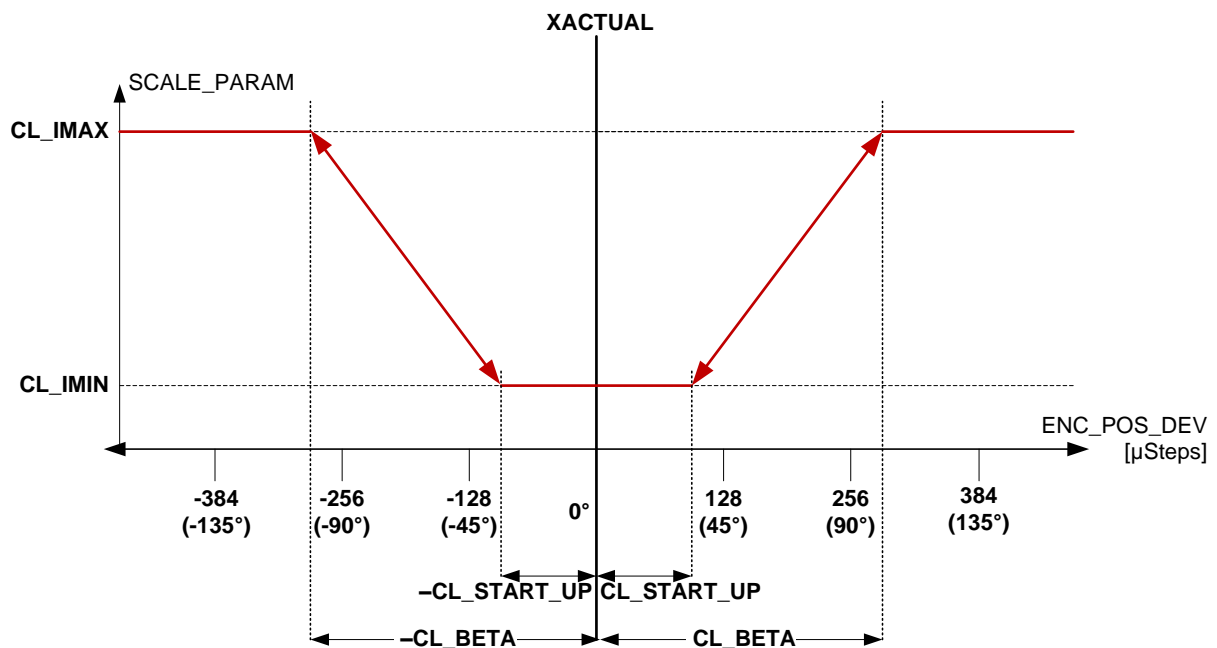
$$\text{real\_scale\_value} = (x + 1) / 32$$

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:

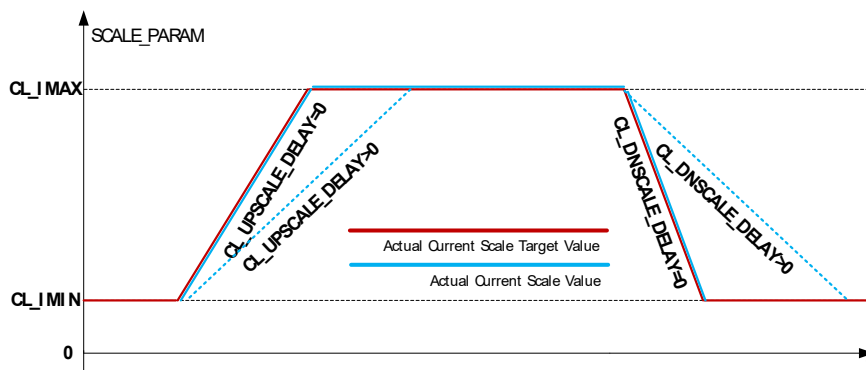
- **Closed loop maximum current scale value CL\_IMAX = SCALE\_VALUES(15:8):** By setting this register, the maximum current will be set. This current scale value is reached as soon as the position mismatch ENC\_POS\_DEV exceeds the CL\_BETA limit. If during TMC26x configuration (see section 3.3) the 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 next figure, 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 4.1** Current scaling value dependent on position mismatch during closed loop scaling modus 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 4.2 depicts the current scaling timing behavior as a function of `CL_UPSCALE_DELAY` and `CL_DNSCALE_DELAY`.



**Figure 4.2** Current scaling timing behavior

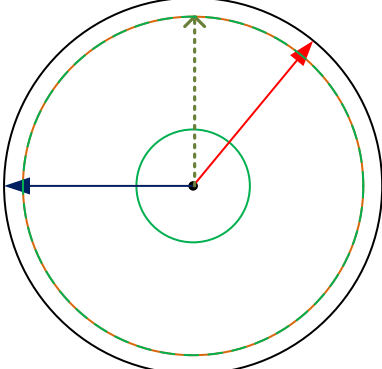
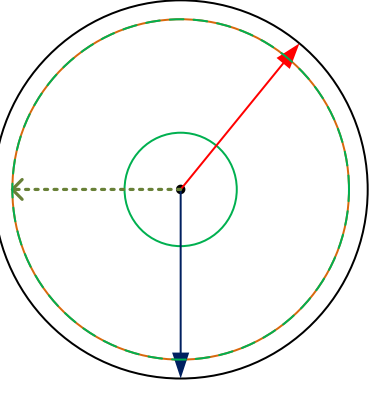
After setting the closed loop scale values properly, closed loop scaling can be enabled: 0x8500000080.

**ATTENTION** Note that, even other scaling modes are enabled, any scale mode becomes disabled if closed loop scaling is enabled!

## 4.1 Example for a combined usage of closed loop angle 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 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 · IMAX (green circle), CL_IMAX = 0.9 · IMAX (orange circle), CL_STARTUP = 31.6° (90 microsteps)	
1. Small deviation= 29.7°, minimum current = 0.3 · IMAX , position will be hold	
2. Moderate deviation= 53.7°, medium current = 0.53 · IMAX, position will be hold	
3. Large deviation= 79.9°, large current = 0.8 · IMAX, position will be hold	

<p>4. Very large deviation <math>&gt; 90^\circ</math>, maximum current = <math>0.9 \cdot I_{MAX}</math>, „Motor follows encoder/load“</p>	
<p>5. The overload situation holds on, maximum current = <math>0.9 \cdot I_{MAX}</math>, „Motor follows encoder/load“</p>	

## 5 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  $\gamma$ -correction where a velocity dependent angle in motion direction will be added to the current commutation angle.

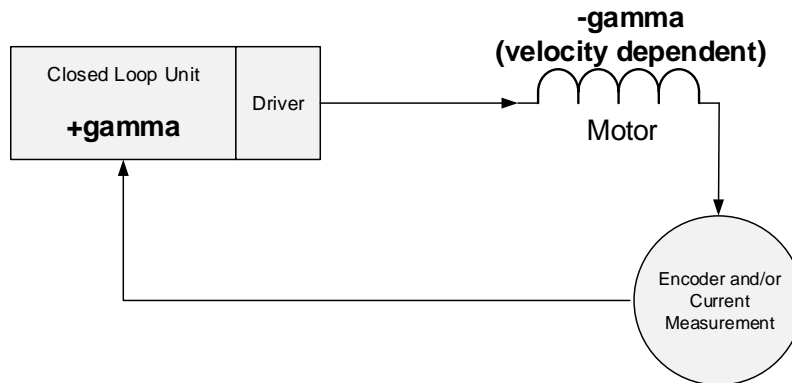


Figure 5.1 Back-EMF consideration with  $\gamma$ -correction unit based on measured encoder velocity

The **maximum gamma angle** can be set to almost  $90^\circ$  (= 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  $180^\circ$  if **CL\_BETA** is also set  $90^\circ$  (= 255 microsteps).

**ATTENTION** If the  $\gamma$ -correction is turned on, the maximum possible commutation will be **( CL\_BETA + CL\_GAMMA )**. This value **must not exceed  $180^\circ$**  (512 microsteps at 256 microsteps per fullstep) because angles of  $180^\circ$  and more will result in **unwanted motion direction changes!**

Switching on  $\gamma$ -correction, set **cl\_emf\_en** to '1' in the **ENC\_IN\_CONF** register 0x07. Furthermore, the velocity limits for the  $\gamma$ -correction have to be assigned:

ENC\_IN\_CONF && 0x00FE7FFFFFFF || 0x8702400000

- **CL\_VMIN\_EMF 0x60**: Below this velocity value [pps] the  $\gamma$ -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  $\gamma$ -correction angle.
- **Beyond ( CL\_VMIN\_EMF + CL\_VADD\_EMF )** the current  $\gamma$ -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  $\gamma$ -correction angle **GAMMA** will be calculated by a linear function which is depicted in the following figure

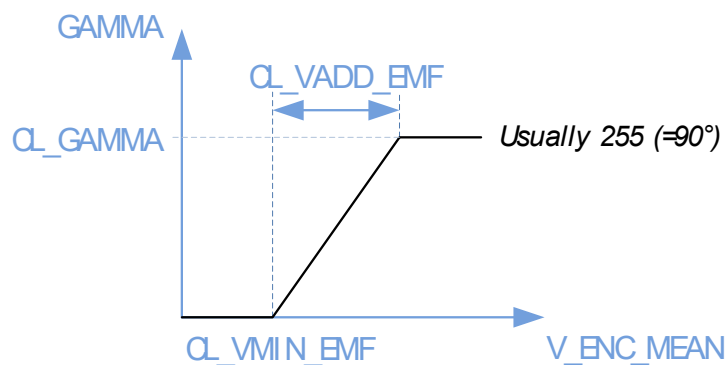


Figure 5.2 Calculation of the current back emf angle **GAMMA**

## 5.1 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 5.3 depicts an oscilloscope shot where no distortion of the motor current is observed which should be the normal case during low velocities.

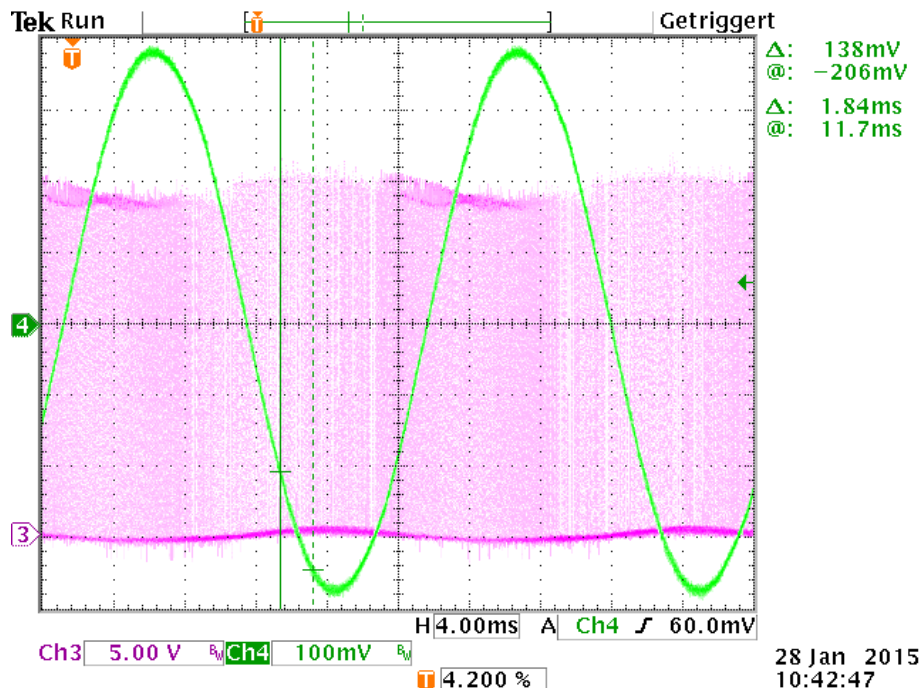


Figure 5.3 Oscilloscope shot of a motor signal line (green= motor current, magenta= motor output driver sense resistor voltage) when the current curve is not be distorted

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 5.4.

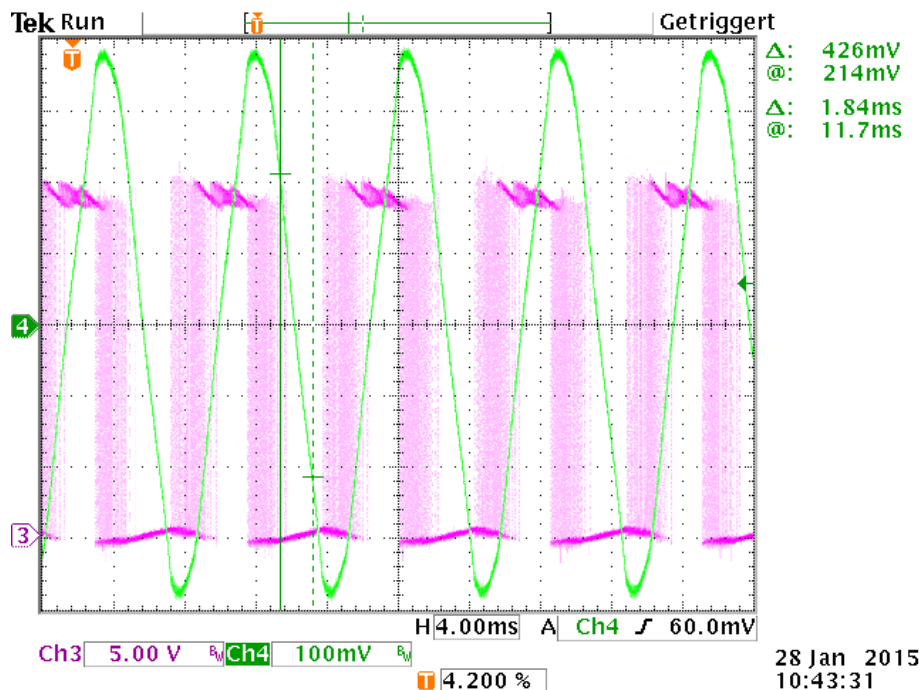


Figure 5.4 Oscilloscope shot of a motor signal line (green= motor current, magenta= motor output driver sense resistor voltage) when the current curve will begin to be distorted due to an increased back emf voltage

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  $\gamma$ -correction is the vanishing of chopper cycles in the voltage curve of the sense resistor as it can be also seen in Figure 5.5 if the chopper control is turned on.



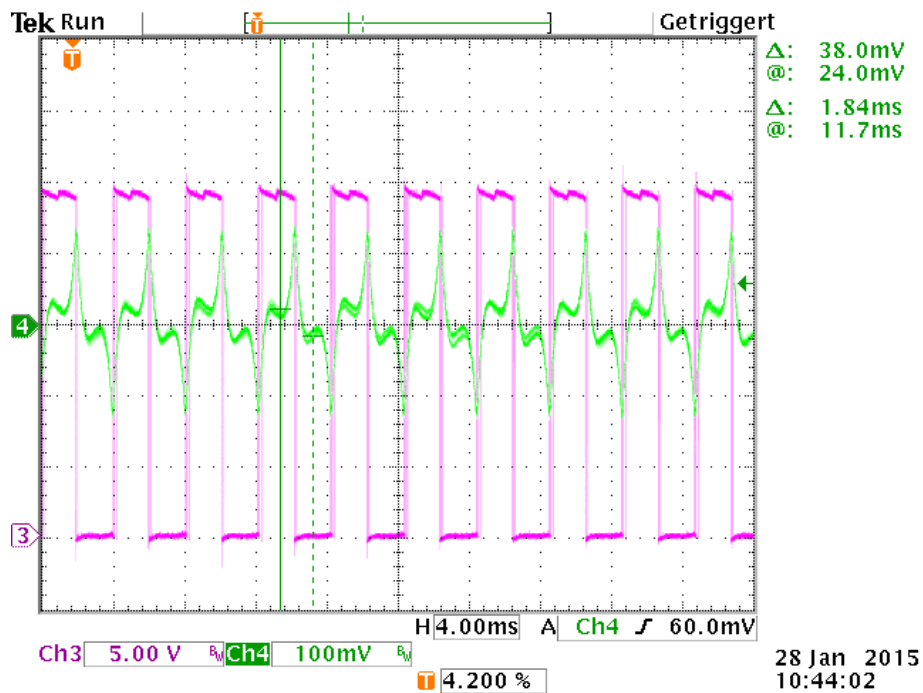


Figure 5.5 Oscilloscope shot of a motor signal line (green= motor current, magenta= motor output driver sense resistor voltage) when the current curve is completely distorted due to the back emf voltage at high velocities. Further on, no chopper cycles can be identified in the magenta voltage curve.

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.

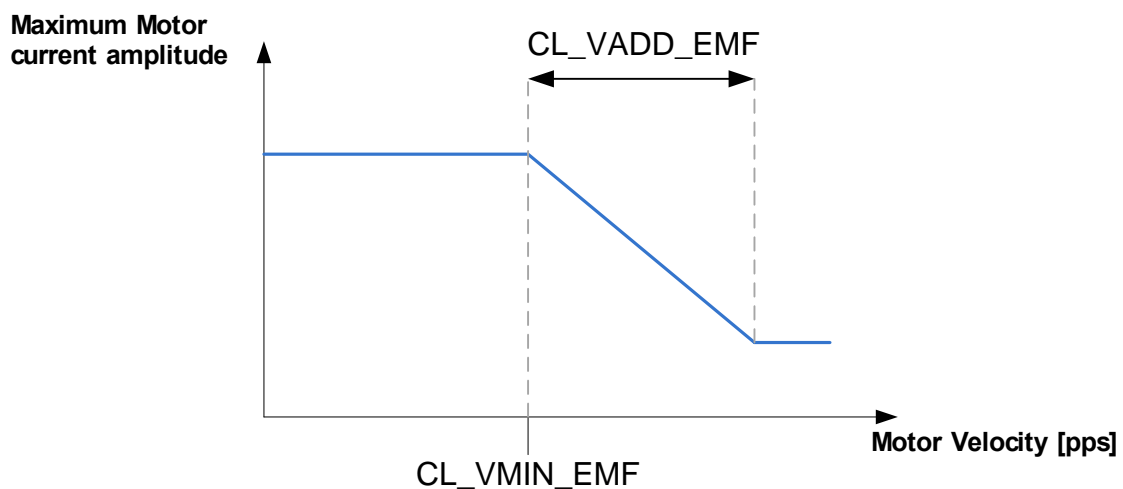


Figure 5.6 Plotting maximum amplitude of the motor current over the motor velocity will also lead to the identification of the velocity limits for the  $\gamma$ -correction

**ATTENTION** To get most sufficient result for motor behavior with  $\gamma$ -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.

## 5.2 Velocity calculation

As it can be seen from Figure 5.2, the trimming is based on the **real motor velocity**. This velocity ENC\_VEL [pps] is calculated internally by TMC4361 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	
	<b>ENC_VMEAN_WAIT &gt; 32</b> 8bit at <b>0x63(7:0)</b>	<b>Automatically set</b> to the assigned encoder update rate <b>SER_PTIME 0x58</b>
ENC_VMEAN_FILTER	Filter exponent for encoder mean velocity calculation 4bit at <b>0x63(11:8)</b>	
ENC_VMEAN_INT	Maximum update delay period [#clock cycles] for the V_ENC calculation	
	16bit at <b>0x63(31:16)</b> If ENC_VMEAN_INT < 256, it will be set automatically to 256!	<b>Automatically set</b> to the assigned encoder update rate <b>SER_PTIME 0x58</b>
CL_CYCLE	Delay period [#clock cycles] for the closed loop control → closed loop control update rate = 1 / CL_CYCLE	
	<b>Fixed for ABN encoders for a minimum possible delay</b> (mostly 5 clock cycles)	16bit at <b>0x63(31:16)</b> Set to at least <b>SER_PTIME 0x58</b>
SER_ENC_VARIATION	---	8bit at <b>0x63(7:0)</b> Adaptation of maximum accepted deviation between two consecutive encoder values If set to 0: maximum accepted deviation = 1/8 · ENC_IN_RES Else: maximum accepted deviation = 1/8 · SER_ENC_VARIATION/256 · ENC_IN_RES <b>serial_enc_variation_limit has to be set to '1'</b> (ENC_IN_RES register 0x07)

**HINT** 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.

**ATTENTION**    **TMC4361-LA only:** 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  $\gamma$ -correction if  $CL\_VMIN\_EMF$  has been already exceeded.

## 6 Examples

In the following complete examples are illustrated by assigning the different registers.

### Example 1: Motor driver TMC26x (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 =  $\pm 50$  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 $\mu C \rightarrow TMC4361$
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.	<b>Wait for an interrupt</b> 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.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
11.	Disable the SMARTEN register of TMC26x/389 (cover datagram)	0xEC000A0000
12.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
13.	Set the SGSCONF register of TMC26x/389 (cover datagram): SGT=0, CS=24	0xEC000C0018
14.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
15.	Set the DRVCONF register of TMC26x/389 (cover datagram): SLPH=3, SLPL=3, DISS2G=off, TS2G=0-3.2 $\mu$ s, SDOFF=on, VSENSE=0	0xEC000EF080
16.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
17.	Encoder resolution: 1000 pole pairs $\rightarrow$ ENC_IN_RES = 4000 ( $\rightarrow$ 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.	<b>Wait for until TARGET_REACHED is set again</b> (can also be set as interrupt!)	...
29.	Turn on closed loop operation and calibration	0x8721400000
30.	<b>Wait for 10us</b> and then turn off closed loop calibration	...0x8720400000
31.	Proportional term for velocity limitation = 1000	0xDA000003e8
32.	Integral term for velocity limitation = 50	0xDB00000032
33.	Clipping value for catch-up velocity = 50 kpps	0xDE0000C350
34.	Clipping value for integral term = 1000	0xDD000003e8
35.	Turn on velocity limit for closed loop operation	0x8728400000
36.	Current scale limits: CL_IMAX = 255, CL_IMIN = 100, CL_START_UP = 100	0x860064FF64
37.	CL_UPSCALE = 1000 clock cycles	0x98000003E8
38.	CL_DNSCALE = 100000 clock cycles	0x99000186A0
39.	Enable closed loop scaling	0x8500000080
40.	CL_TR_TOLERANCE = 60 (if absolute position mismatch is smaller than 3 encoder transitions, TARGET_REACHED can be set)	0xD20000003C
41.	Set VMAX = 0 to prevent unwanted ramp starting	0xA400000000
42.	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 =  $\pm 50$  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 $\mu C \rightarrow TMC4361$
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.	<b>Wait for an interrupt</b> 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.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
11.	Disable the SMARTEN register of TMC26x/389 (cover datagram)	0xEC000A0000
12.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
13.	Set the SGSCONF register of TMC26x/389 (cover datagram): SGT=0, CS=24	0xEC000C0018
14.	<b>Wait for an interrupt</b> 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.	<b>Wait for an interrupt</b> 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	0xD0xxxxxxx
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	0xB7xxxxxxx

23.	<b>Wait for until TARGET_REACHED is set again.</b> (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	0xD0xxxxxxx
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)	0xB7xxxxxxx
33.	<b>Wait for until TARGET_REACHED is set again.</b> (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	0xB7xxxxxxx
36.	Repeat steps 33..35 400 times (2 revolutions) <b>It is recommended to also move into the opposite direction.</b>	
37.	Check DEV_SEARCH = (DEV_MAX + DEV_MIN) / 2	
38.	Move to next fullstep position XTARGET = XTARGET - 256	0xB7xxxxxxx
39.	<b>Wait for until TARGET_REACHED is set again.</b> (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	0xB7xxxxxxx
42.	Turn on closed loop operation and calibration	0x87x1400000
43.	<b>Wait for 10us</b> and then turn off closed loop calibration	...0x87x0400000
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.	Turn on velocity limit for closed loop operation	0x87x8400000
49.	Set CL_VMIN_EMF (velocity limits have to be found by open loop movements before) (see chapter 5)	0xE0xxxxxxx
50.	Set CL_VADD_EMF (velocity limits have to be found by open loop movements before) (see chapter 5)	0xE1xxxxxxx
51.	Set encoder mean velocity settings (see section 5.2)	0xE3xxxxxxx
52.	Turn on back-emf consideration for closed loop operation	0x87xA400000
53.	Current scale limits: CL_IMAX = 255, CL_IMIN = 100, CL_START_UP = 100	0x860064FF64
54.	CL_UPSCALE = 200 clock cycles	0x98000007D0
55.	CL_DNSCALE = 100000 clock cycles	0x99000186A0
56.	Enable closed loop scaling	0x8500000080
57.	Set VMAX = 0 to prevent unwanted ramp starting	0xA400000000
58.	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 =  $\pm 50$  kpps
- 400 full steps per revolution
- $I_{RMS} = 1.1$  A, Sense resistor:  $R_{SENSE} = 0.15 \Omega$
- $V_{SENSE} = 0 \rightarrow I_{FS} = 1.88$  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 $\mu C \rightarrow TMC4361$
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.	<b>Wait for an interrupt</b> 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, HSTR=2, TOFF=3	0xED000000EC 0xEC00010223
10.	<b>Wait for an interrupt</b> 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.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
13.	Set any other register of TMC2130 according to your requirements	0xED000000xx 0xECxxxxxxx
14.	<b>Wait for an interrupt</b> 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 $\rightarrow ENC\_IN\_RES = 4000$ ( $\rightarrow ENC\_CONST = 25.6$ )	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)	0xB7xxxxxxx
28.	<b>Wait for until TARGET_REACHED is set again</b> (can also be set as interrupt!)	...
29.	Turn on closed loop operation and calibration	0x8721400000
30.	<b>Wait for 10us</b> and then turn off closed loop calibration	...0x8720400000
31.	Proportional term for velocity limitation = 1000	0xDA000003e8
32.	Integral term for velocity limitation = 50	0xDB00000032
33.	Clipping value for catch-up velocity = 50 kpps	0xDE0000C350
34.	Clipping value for integral term = 1000	0xDD000003e8
35.	Turn on velocity limit for closed loop operation	0x8728400000
36.	Current scale limits: CL_IMAX = 255, CL_IMIN = 100, CL_START_UP = 100	0x860064FF64
37.	CL_UPSCALE = 1000 clock cycles	0x98000003E8
38.	CL_DNSCALE = 100000 clock cycles	0x99000186A0
39.	Enable closed loop scaling	0x8500000080
40.	CL_TR_TOLERANCE = 60 (if absolute position mismatch is smaller than 3 encoder transitions, TARGET_REACHED can be set)	0xD20000003C
41.	Set VMAX = 0 to prevent unwanted ramp starting	0xA400000000
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

#### Preliminary considerations:

- Differential ABN encoder
- Back-emf consideration
- Catch-up velocity limit =  $\pm 50$  kpps
- 200 full steps per revolution
- $I_{RMS} = 1.1$  A, Sense resistor:  $R_{SENSE} = 0.15 \Omega$
- $V_{SENSE} = 0 \rightarrow I_{FS} = 1.88$  A
- $I_{PEAK} = I_{RMS} \cdot \sqrt{2} = 1.56$  A  $\rightarrow CS = 25$  ( $\rightarrow$  Maximum value set in scale register of TMC4361)

#### Sequence of register access:

Sequence	Comments	Order $\mu C \rightarrow TMC4361$
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.	<b>Wait for an interrupt</b> 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.	<b>Wait for an interrupt</b> 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.	<b>Wait for an interrupt</b> and then clear events.	...0x0E00000000
13.	Set any other register of TMC2130 according to your requirements	0xED000000xx 0xECxxxxxxxx
14.	<b>Wait for an interrupt</b> 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 $\rightarrow ENC\_IN\_RES = 2000$ ( $\rightarrow ENC\_CONST = 25.6$ )	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	0xB7xxxxxxx
23.	<b>Wait for until TARGET_REACHED is set again.</b> (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	0xD0xxxxxxx
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)	0xB7xxxxxxx
33.	<b>Wait for until TARGET_REACHED is set again.</b> (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	0xB7xxxxxxx
36.	Repeat steps 33..35 400 times (2 revolutions) <b>It is recommended to also move into the opposite direction.</b>	
37.	Check DEV_SEARCH = (DEV_MAX + DEV_MIN) / 2	
38.	Move to next fullstep position XTARGET = XTARGET - 256	0xB7xxxxxxx
39.	<b>Wait for until TARGET_REACHED is set again.</b> (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	0xB7xxxxxxx
42.	Turn on closed loop operation and calibration	0x87x1400000
43.	<b>Wait for 10us</b> and then turn off closed loop calibration	...0x87x0400000

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.	Turn on velocity limit for closed loop operation	0x87x8400000
49.	Set CL_VMIN_EMF (velocity limits have to be found by open loop movements before) (see chapter 5)	0xE0xxxxxxx
50.	Set CL_VADD_EMF (velocity limits have to be found by open loop movements before) (see chapter 5)	0xE1xxxxxxx
51.	Set encoder mean velocity settings (see section 5.2)	0xE3xxxxxxx
52.	Turn on back-emf consideration for closed loop operation	0x87xA400000
53.	Current scale limits: CL_IMAX = 25, CL_IMIN = 10, CL_START_UP = 50	0x860032190A
54.	CL_UPSCALE = 200 clock cycles	0x98000007D0
55.	CL_DNSCALE = 100000 clock cycles	0x99000186A0
56.	Enable closed loop scaling	0x8500000080
57.	Set VMAX = 0 to prevent unwanted ramp starting	0xA400000000
58.	Set ramp conditions according to the required setup...	

## 7 Revision History

### 7.1 Document Revisions

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	2016-Jul-21	HS	Advanced calibration process in examples: Step40 "If ENC_POS_DEV $\approx$ DEV_SEARCH $\rightarrow$ CALIB_POS = XTARGET" Instead of "If ENC_POS_DEV $\approx$ DEV_MIN $\rightarrow$ CALIB_POS = XTARGET"
0.992	2017-Jan-16	HS	Advanced calibration process in examples: Step37 "Check DEV_SEARCH = (DEV_MAX + DEV_MIN) / 2" Instead of "Check DEV_SEARCH = (DEV_MAX - DEV_MIN) / 2"

Table 1 Document Revisions