DC Motor Control with TMC4671

The TMC4671 hardware controller performs field-oriented control (FOC) for two-phase stepper motors and three-phase permanent magnet synchronous motors (PMSM). It also supports direct current (DC) motor control.

Why TMC4671 for DC motor servo control? The TMC4671 provides hardware closed loop torque control, velocity control, and position control, even for DC motors decoupling motor control from the application.

Contents

One Analog Way, Wilmington, MA 01887-2356, U.S.A | Tel: 781.329.4700

1 DC Motor Closed Loop Control

When using the TMC4671 for closed loop DC motor control, one current controller is used for torque control while the coordinate transformations required for FOC are skipped. Servo control functions as velocity control and position control of DC motors are similar to two-phase stepper motors and three-phase permanent magnet synchronous motors. The TMC4671 hardware provides an analog digital converter (ADC) engine, an encoder engine, PI controllers for closed loop current control, velocity control, position control, and a PWM engine usable in a unified way for DC motor control.

1.1 DC Motor Control Configuration

The base for the torque control of a DC motor includes configuration of ADC for current measurement, optional PWM adjustment, and the essential parameterization of proportional (P) parameter and integral (I) parameter of the PI closed loop current controller. The DC motor control of the TMC4671 is selected by the dedicated DC motor type control mode.

A proper setup of closed loop current control is essential when using closed loop velocity control. For velocity control, set up some parameters of the position sensor. The TMC4671 uses a position sensor to measure the speed. Proper setups of closed loop current current control and velocity control are essential when using position control.

The TMC4671 is equipped with integrated limiters to define a safe operation area even for faults caused by incorrect configuration. These limiters are useful, especially during initial setup.

An important fault that might damage a motor or a power supply is a wrong sign of current measurement. With it, the current controller opens the PWM duty cycle up to 100%. With a programmable limiter, it is possible to clip the PWM duty cycle to save the operation, where the resulting current is limited by the inner resistance of the motor. However, this limiter limits the reachable performance of the motor.

2 TMC4671 Evaluation Boards for Application Note

The evaluation kit used as an exemplary hardware platform configuration for this application note is Landungsbruecke v2.0 + TMC4671-EVAL v.1.1 + TMC-UPS-10A/70V-EVAL v.1.1 + DC Motor, respectively, a coil with an inductance L = 1mH and resistance of 1Ω with a 24V power supply. A coil is useful for emulating a blocked DC motor for the initial setup of the closed loop current regulation.

NOTE Make sure the DC motor current matches the power stage current. The sense resistor may need to be adapted (refer to the TMC4671 current calculation sheet).

2.1 DC Motor Turning

Why initially turn the DC motor open loop? Initially, the motor should be operated in open loop mode to verify the current measurement for the motor, confirm the correct mapping between the ADC channel selection and DC motor terminals, and adjust the ADC scaling and offset parameter.

To turn a DC motor open-loop, apply a supply voltage V_M to the DC motor. Together with PWM, the effective voltage U applied to the motor is U = V_M × PWMdutyCycle[%]. For DC motor mode, the PWM duty cycle of the TMC4671 is programmed using the *UQ_EXT* value of the register *UQ_UD_EXT*, where the *UQ_EXT* is a 16-bit signed value, with the sign representing the sign of the effective voltage applied to the motor. The Upwm[V] = V_M[V] × *UQ_EXT*[-32767, ..., 0, ..., +32767] is the effective voltage between the terminals

of the PWM power stage. The relationship between *UQ_EXT* and the motor phase PWMdutyCycle[%] is given as:

$$\text{PWMdutyCycle}[\%] = \left(\frac{UQ\_EXT}{2^{16}\text{-}2} + 0.5\right) \times 100 \quad (1)$$

Table 1 provides the UQ_EXT example and the resulting PWM duty cycle.

UQ_EXT	PWMdutyCycle	Motor Operation
-32767	0%	Motor run (CCW)
-3000	45.4%	Motor run (CCW)
0	50%	Motor standstill
3000	54.5%	Motor run (CW)
32767	100%	Motor run (CW)

Table 1: UQ_EXT and Resulting PWM Duty Cycle

2.2 Select DC Motor Type

Choose *MOTOR_TYPE* = 1 for the DC motor with number of pole pairs NPP = 1. The number of pole pairs is not relevant for DC motor control but with it, electrical angles are the same as mechanical angels when measuring angles with encoders. The setting *MOTOR_TYPE* = 1 configures the PWM assigning the terminals U and V for the DC motor (Figure 1).

MOTOR_TYPE = 0 : No motor (PWM choppers in zero voltage mode)

MOTOR_TYPE = 1 : DC motor

MOTOR_TYPE = 2 : Two-phase permanent magnet synchronous motor (stepper motor)

MOTOR_TYPE = 3 : Three-phase permanent magnet synchronous motor (PMSM, brushless motor)

2.2.1 DC Motor Turning - Open Loop for Current Measurement Setup

For open loop motor turning, apply PWM with a programmed duty cycle for the two half-bridges with the DC motor connected between the terminals U and V. The supply voltage, together with the PWM duty cycle, determines the speed of the motor.

For FOC2 (stepper motor) and FOC3 (BLDC motor), the *UD_EXT* is used to turn the motor open loop to determine the D direction of the current. In contrast to FOC3 and FOC2, for the DC motor (named FOC1 because the motor mechanically makes FOC by its mechanical commutator brushes), the *UQ_EXT* is used to turn the DC motor because the current through the DC motor generates torque similar to the torque generating current IQ in case of FOC2 and FOC3.

2.2.2 DC Motor Turning - Closed Loop with PI Regulator

For closed loop current control, a PI regulator controls the current by measuring the actual current and regulating the difference to the desired target current to zero. Initially, set the P and I parameters of the PI regulator to zero.

First, the P parameter should be incremented with temporarily set I = 0 when determining the P parameter until the PI regulator reaches half of the desired target current. With this determined P parameter,

the parameter I should be incremented until the PI regulator reaches the full desired target current. With this, there is an initial setup for the current regulation. The magnitude of the parameter I determines how fast the PI current regulator reaches the desired target current. A too large parameter I causes control loop oscillations.

3 Setup ADC for Measurement of Current

The TMC4671 supports two parallel sampling ADC channels for motor current measurement. For DC motor current measurement, the *ADC_IUX* is associated to measure the current of the DC motor. Thus, ADC_I0_RAW (or ADC_I1_RAW, depending on the setup) must be assigned accordingly. Some basic ADC parameter must be initialized. The other ADC channel is not needed.

3.1 ADC Delta Sigma - Initial Base Parameters

The TMC4671 is equipped with internal Delta Sigma ADCs. These ADCs provide programmable filtering of input signals to adjust resolution versus speed. The Delta Sigma ADCs are organized into two groups to enable different resolutions and speeds for these groups.

- Group A: Primarily used for current measurement.
- Group B: Primarily used for processing of analog encoder signals.

The following default settings are suitable for most standard applications and are useful as initial parameter settings. Those settings are automatically initialized by the TMCL-IDE Wizard by clicking *Set defaults for DC motor* on the *Settings* page.

ADDR	Address Name	Data	Function
0x04	*dsADC_MCFG_B_MCFG_A*	0x00100010	ADC configuration group B and A
0x05	*MCLK_A*	0x20000000	Delta Sigma Clock A
0x06	*MCLK_B*	0x20000000	Delta Sigma Clock B
0x07	*dsADC_MDEC_B_MDEC_A*	014E014E	Decimation for B and A
0x0A	*ADC_I_SELECT*	0x09000100	Select ADC channel for DC motor
0x1B	*MOTOR_TYPE_N_POLE_PAIRS*	0x00010001	DC motor type, number of pole pairs
0x24	*UQ_UD_EXT*	0x00000000	UQ_EXT for PWM duty cycle, first zero
0x63	*MODE_RAMP_MODE_MOTION*	0x00000008	Classical PID type, UQ_UD_EXT mode

Table 2: ADC Delta Sigma Parameter - Data for TMC6100-EVAL

The values of registers 0x04, 0x07, 0x0A in the wizard may vary depending on the power stage used.

3.1.1 Adjust ADC Offsets, and ADC Scaling and Sign

The integrated ADCs deliver unsigned raw ADC values (ADC_RAW) within the 16-bit unsigned range 0 …65535. The PI current controller needs scaled and offset corrected signed ADC values within the 16-bit signed range of -32767 …+32767 to perform closed loop current control. Similar to FOC2 and FOC3, it is essential to set the correct ADC scale parameter, and correct ADC offset parameter for real-time correction by the integrated ADC scaler and ADC offset compensator. Closed loop current control of the DC motor needs correct association between applied voltage and measured current. For positive voltage UQ, a positive current IQ must be measured. For negative voltage -UQ, a negative current -IQ must be measured.

3.1.1.1 Measure Zero Current to Determine ADC and Sense Amplifier Offset

First, measure the zero current to determine the offset of ADC and sense amplifier. The wizard provides automatic offset compensation (4.2.7).

ADDR	Address Name	Data	Function
0x1A	*PWM_SV_CHOP*	0x0000000	Switch PMW_OFF (zero voltage)
0x03	*ADC_RAW_ADDR*	0x0000000	Set ADC RAW address to read ADC_I0_RAW
0x02	*ADC_RAW_DATA*	ADC raw data	Read actual ADC raw data
0x1A	*PWM_SV_CHOP*	0x0000007	Switch PMW_ON

Table 3: Determine ADC and Sense Amplifier Offset

3.2 PWM Engine and Associated Motor Connectors

The PWM engine of the TMC4671 has eight gate control outputs to control up to four power MOS half-bridges. For three-phase motors, three half-bridges are used (U, V, W). For two-phase stepper motors, four half-bridges are used for (U, V, W, Y). For DC motor control, the first two half-bridges (U, V) are used.

Gate Control Signals	Three-Phase-Motor : 3	Two-Phase-Motor : 2	DC Motor : 1
PWM_UX1_H	U	X1	U
PWM_UX1_L			
PWM_VX2_H	V	X2	V
PWM_VX2_L			
PWM_WY1_H	W	Y1	-
PWM_WY1_L			
PWM_Y2_H	-	Y2	-
PWM_Y2_L			

Table 4: Motor Connection

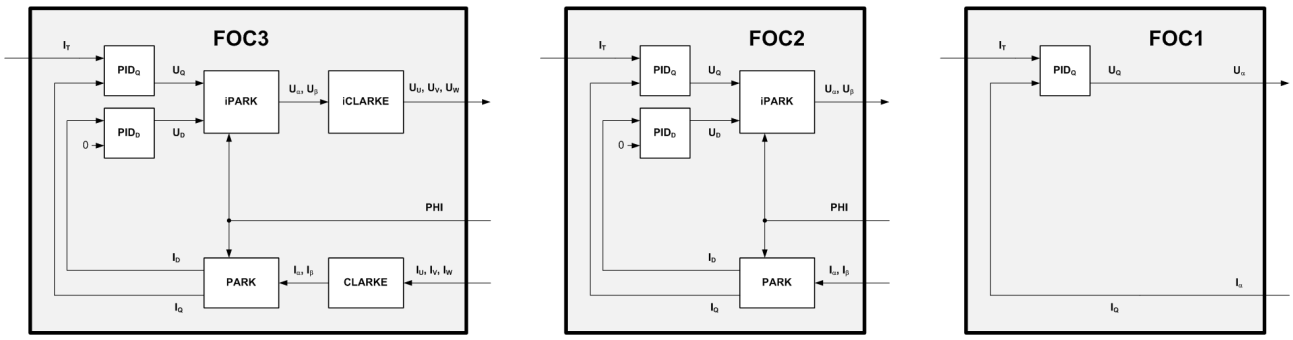

Figure 1: TMC4671 FOC Transformations

For the DC motor current control (FOC1 here), the number of pole pairs is not relevant. In contrast to closed loop current control of two-phase stepper motors (FOC2) and three-phase permanent magnet motors (FOC3), it should be set to 1 to equal the mechanical angle and electrical angle for velocity control and position control.

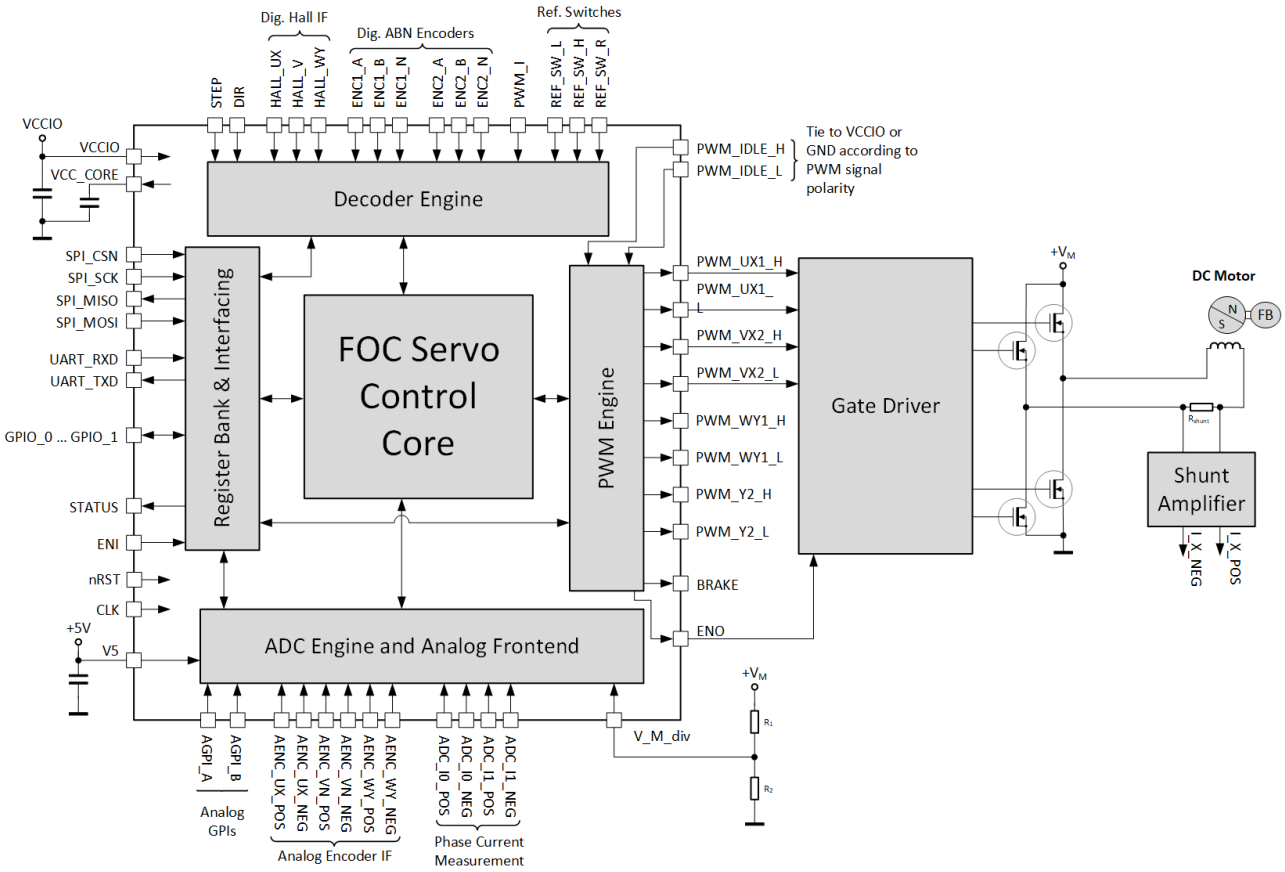

Figure 2: DC Motor Connection to TMC4671

4 DC Motor Setup with TMCL-IDE and its Wizards - Coil

4.1 Setup

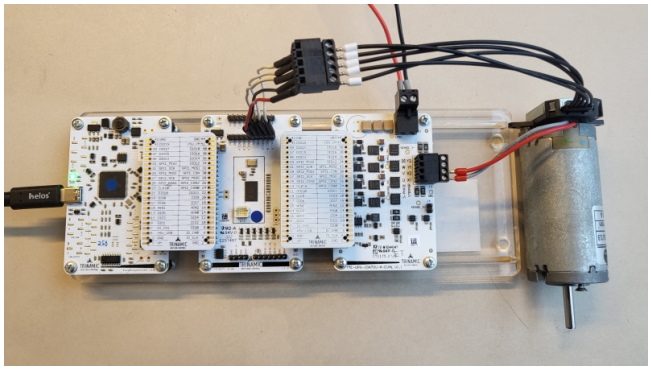

Figure 3: TMC4671 DC Motor Setup

4.2 TMCL-IDE Wizard

4.2.1 Select TMC4671

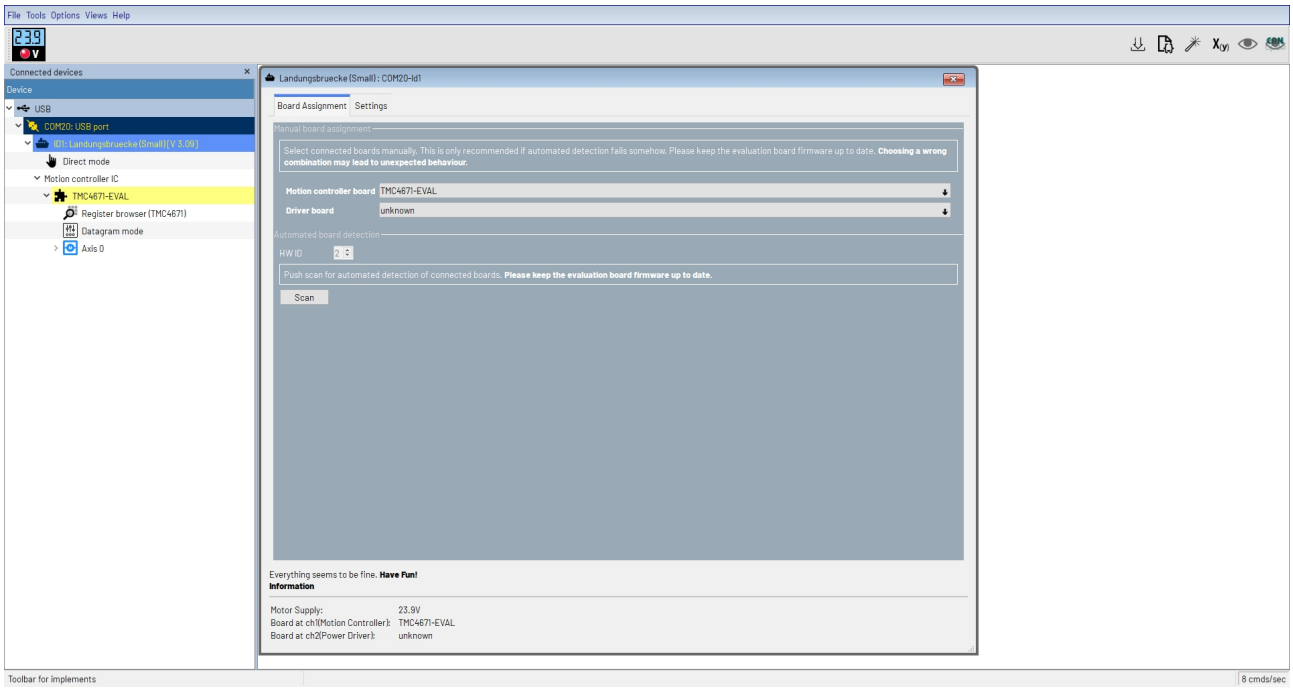

Figure 4: Landungsbruecke TMC4671-EVAL Selection

4.2.2 Start TMCL-IDE Wizard - Click the Weasel

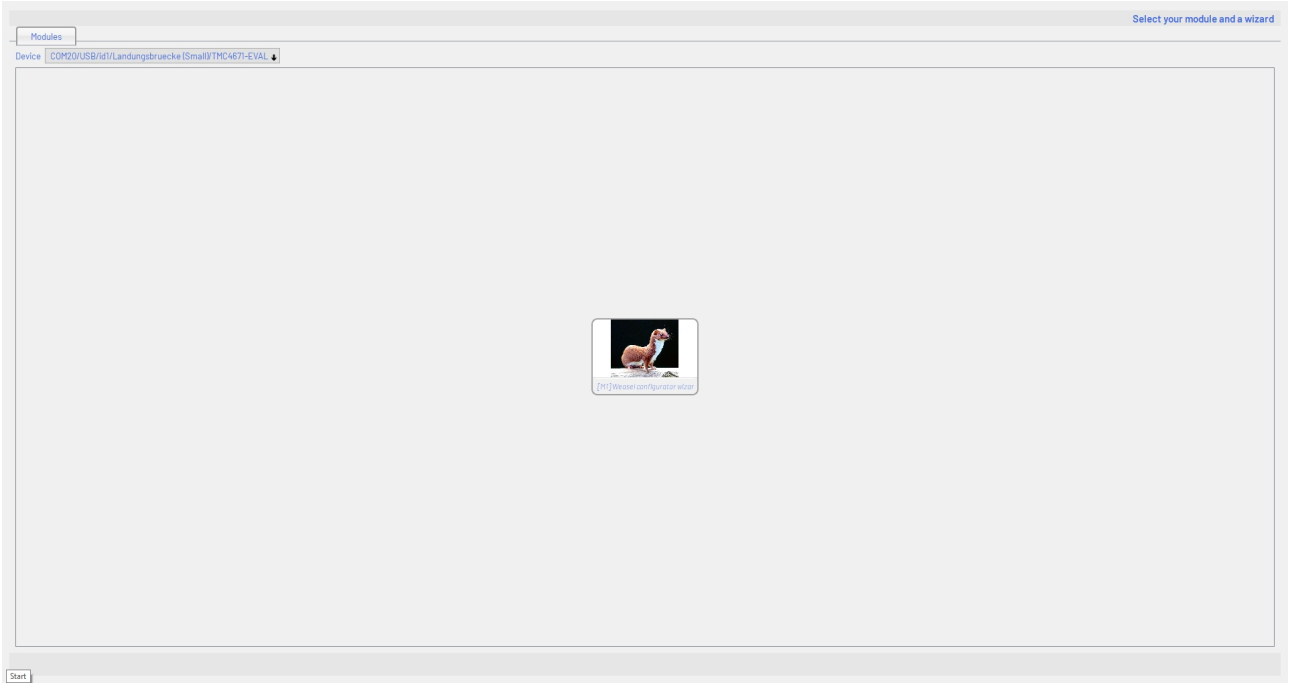

Figure 5: TMC4671 Wizard Selection

4.2.3 TMCL-IDE Wizard - Introduction

Select the relevant configuration pages based on the sensor type: hall sensor, ABN encoder, or analog encoder. For the DC motor, ensure that open loop and ADC selection and configuration are set.

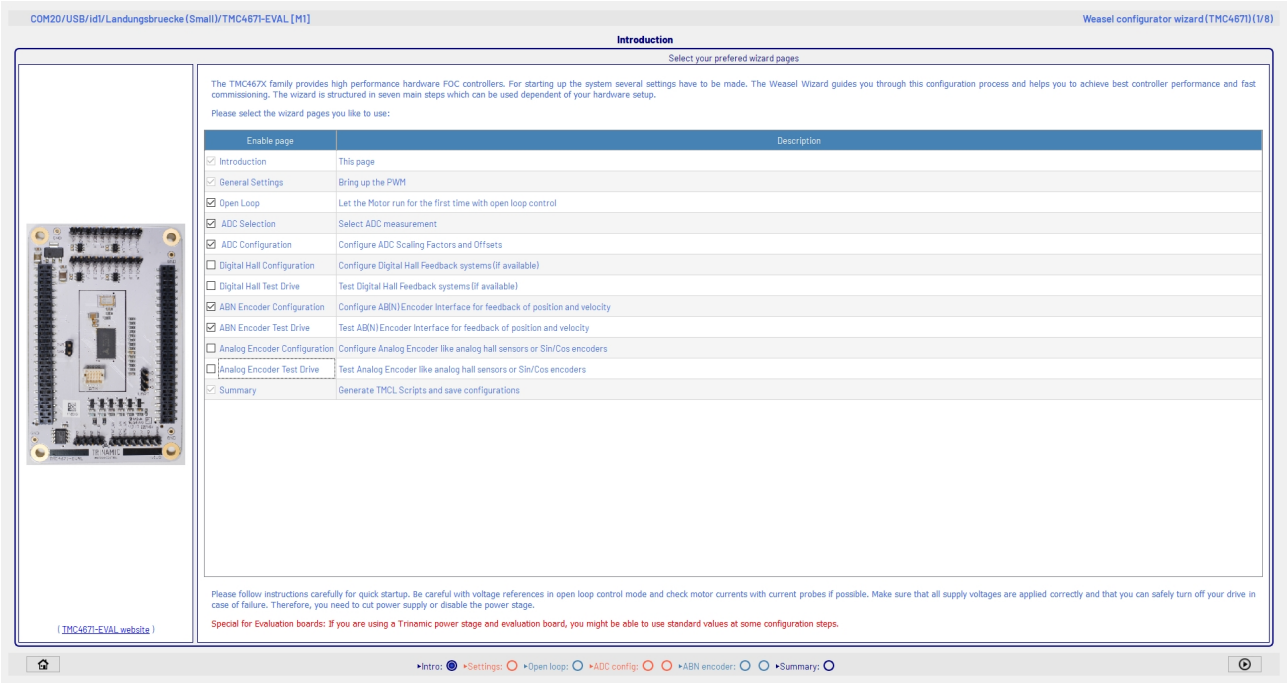

Figure 6: TMC4671 Wizard Selection

4.2.4 TMCL-IDE Wizard - Main Settings - Set Defaults for DC Motor

Select the used power stage and click *Set defaults for DC motor*.

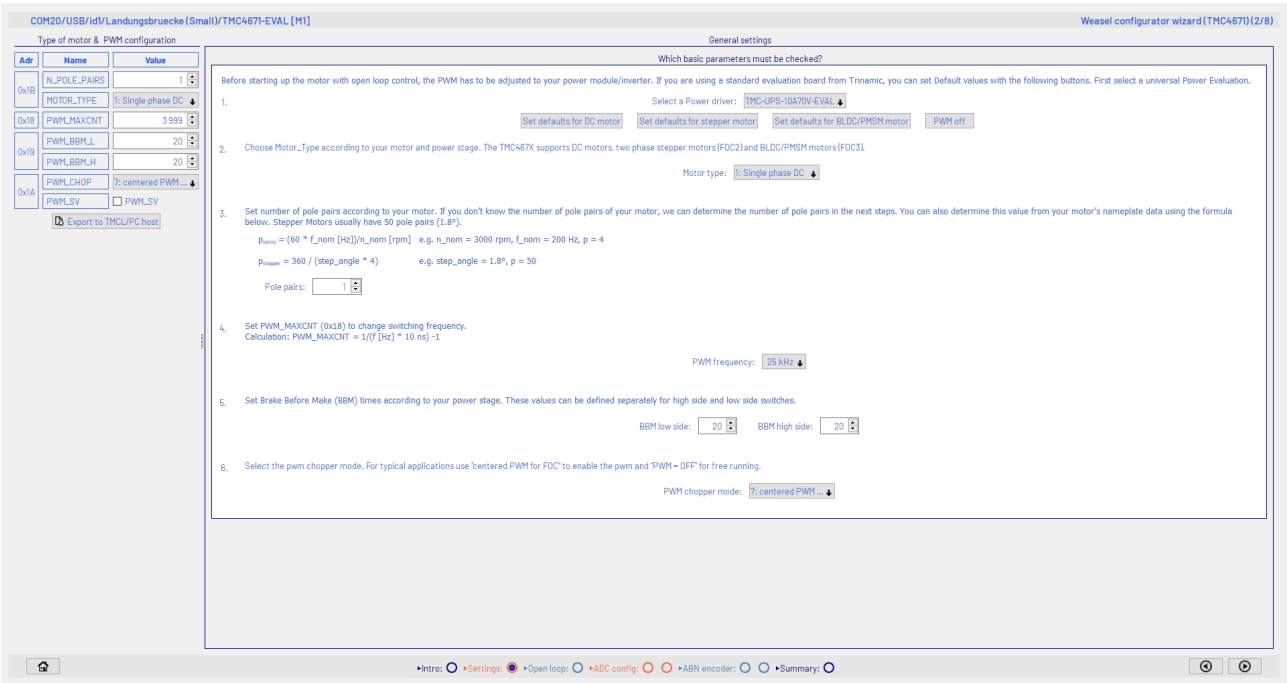

Figure 7: TMC4671 DC Motor Selection

Set **Number of Pole Pairs = 1** for possible later use of encoders.

4.2.5 TMCL-IDE Wizard - Open Loop Settings

Figure 8: Open Loop Test Drive

Use *UQ_EXT* to set the PWM duty cycle for the DC motor to run it open loop. First press *Set Defaults*, then increase the slider and confirm the motor rotates. Start with small values. The range of *UQ_EXT* is -32767, . . . , 0,. . . , +32767 associated with PWM duty cycle -100%, . . . , 0%, . . . , +100%, where -100% stands for negative supply voltage and +100% stands for positive supply voltage. Set *UQ_EXT* before moving on to the ADC pages.

For initial ADC setup, set *UQ_EXT* = 0 and use a coil with inductance L[mH] resistance R[Ω] according to the DC motor or block the DC motor that it does not turn. With this, set up the current measurement and the PI closed loop current control.

4.2.6 TMCL-IDE Wizard - ADC Selection

Figure 9: ADC Selection

Set *ADC_I0_SELECT* = 0 (*ADCSD_I0_RAW*), set *ADC_I_UX_SELECT* = 0 (*UX*). Since only one current channel is required for DC motor operation, ADCSD_I1_RAW can be ignored in this case.

4.2.7 TMCL-IDE Wizard - ADC Configuration

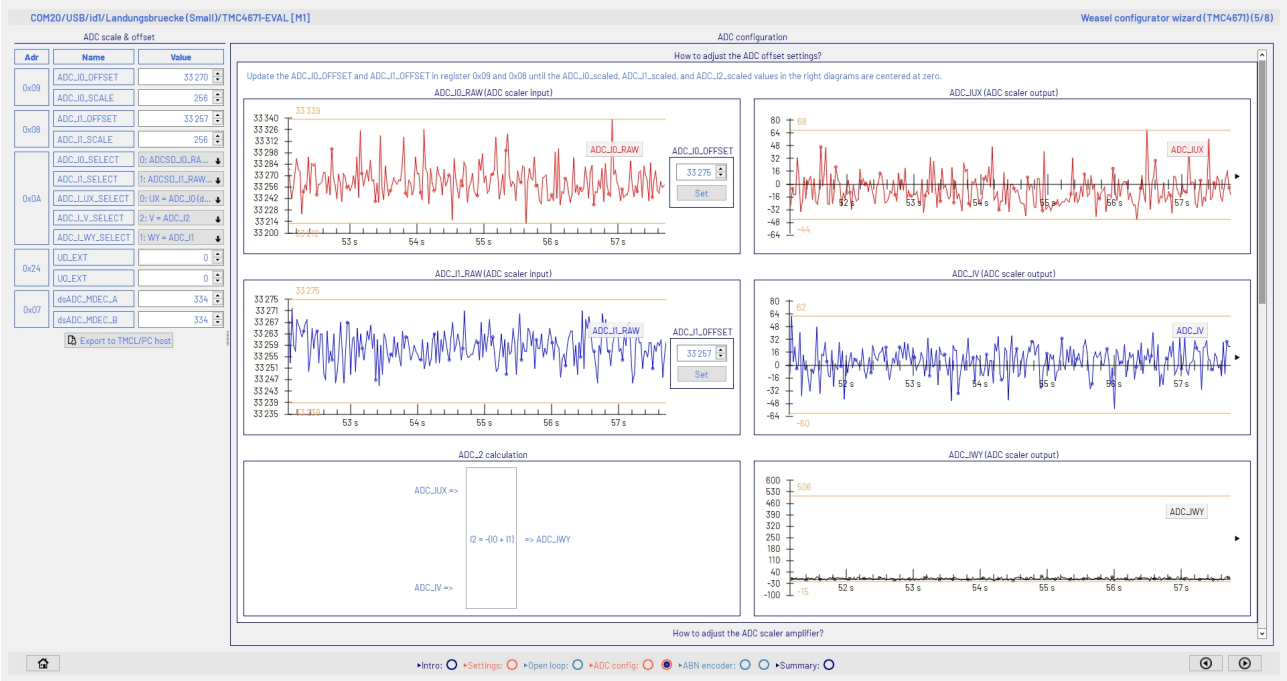

Figure 10: ADC Offset Compensation

Set *ADC_I0_OFFSET* using the *Set* button. Set *ADC_I0_SCALE* = 256. *ADC_IV* and *ADC_IWY* graph can be ignored.

4.2.8 TMCL-IDE Wizard - ADC Configuration - Check Current Scaling

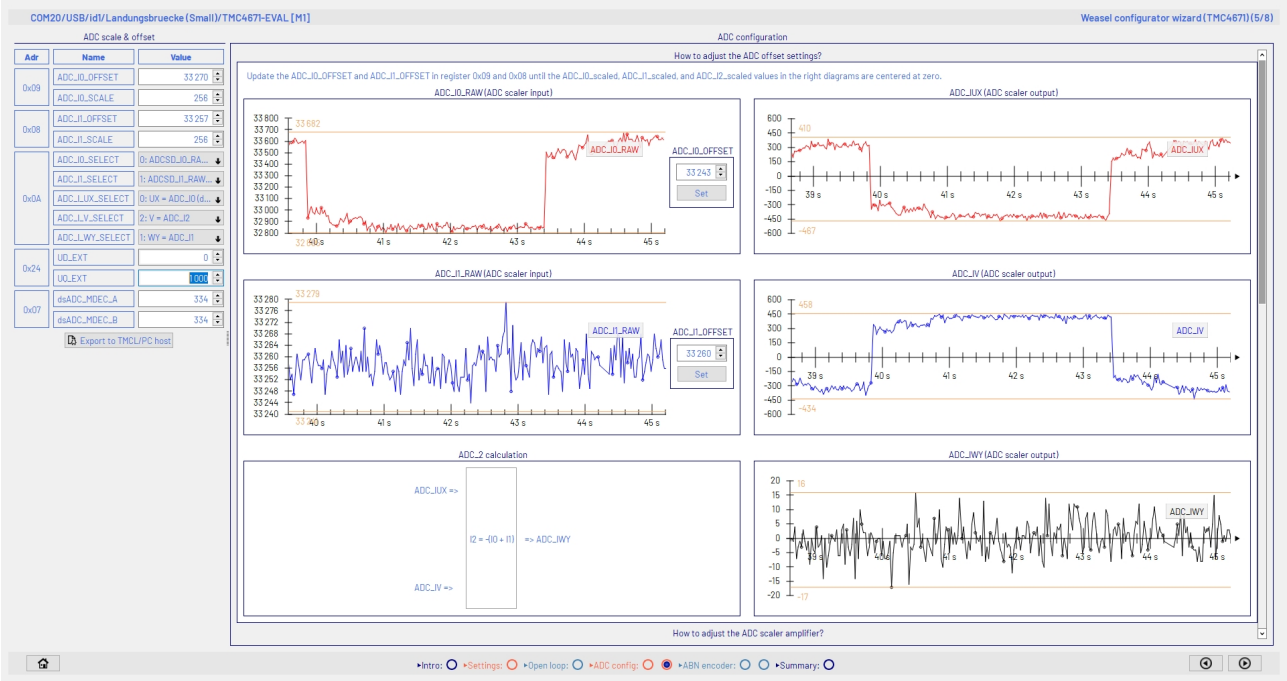

Figure 11: ADC Check Correct Sign

Set *UQ_EXT* = 1000 and *UQ_EXT* = -1000, observe *ADC_IUX*, and check for correct sign. The sign is correct if a positive voltage *UQ_EXT* results in a positive current *ADC_IUX*. During this test, the motor must remain stationary (example, through a brake).

4.2.9 TMCL-IDE Wizard - Encoder Configuration

Turn the motor in open loop (using the *Open Loop* page). Set the encoder pulses per revolution (PPR) and direction. The encoder is correctly set up if positive *ADC_IUX* leads to positive encoder direction (*ABN_DECODER_PHI_E*).
Figure 12 shows the resulting positive current *ADC_IUX*.

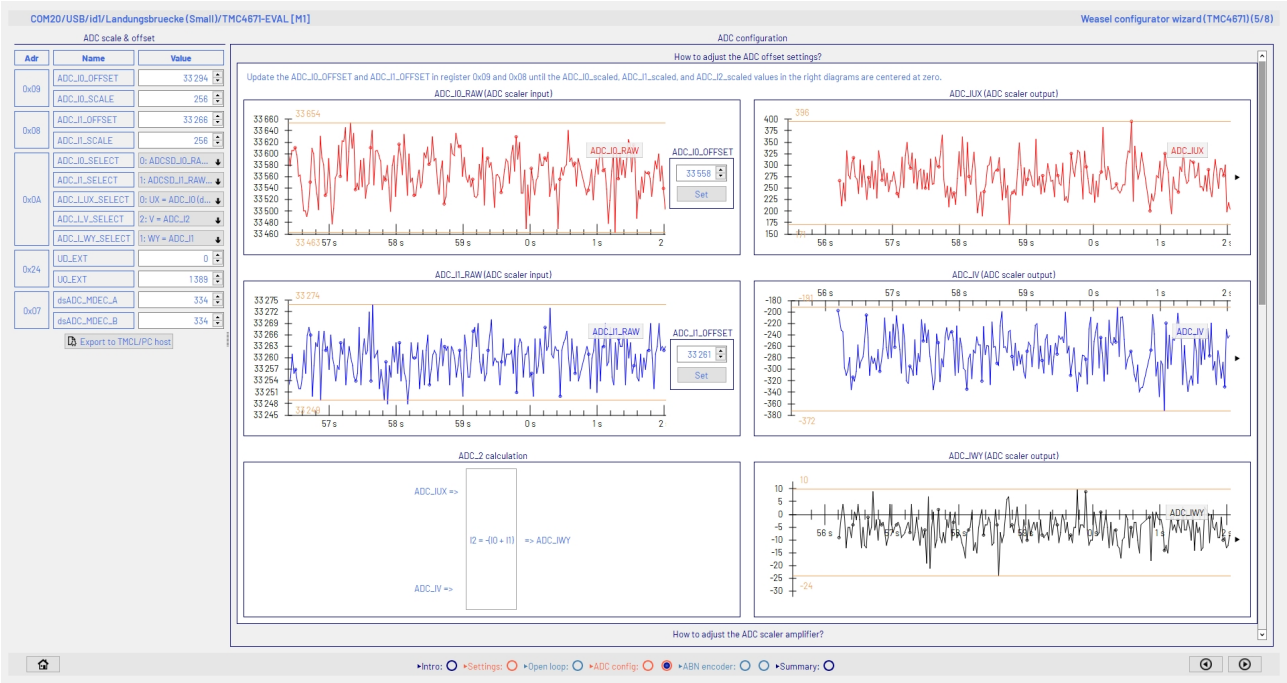

Figure 12: ADC_IUX current

Set the encoder PPR (*ABN_DECODER_PPR*).

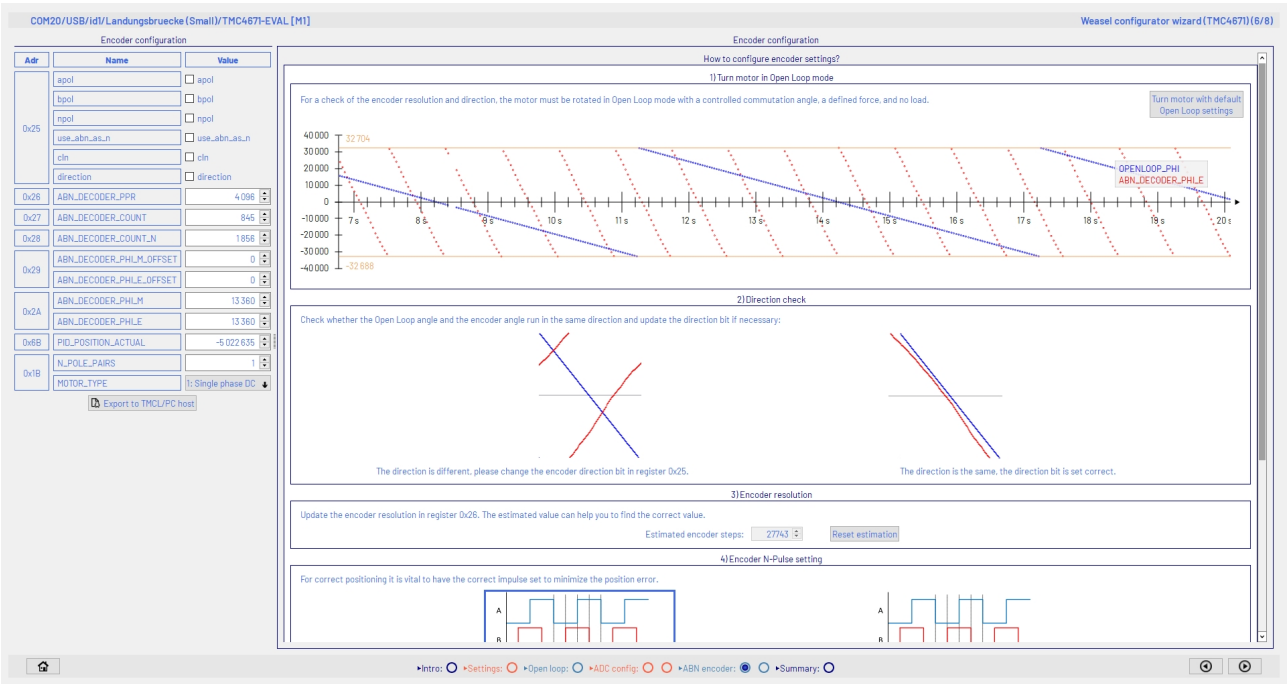

Figure 13: Encoder Signal, Wrong Direction

The *OPENLOOP_PHI* can be ignored.

Since the encoder direction is negative (in contrast to the current *ADC_IUX*), change the encoder *direction* (0x25).

Figure 14: Configured Encoder

4.2.10 TMCL-IDE Wizard - Encoder Test Drive (Torque Mode)

Figure 15: Torque Mode

Set Defaults and Start. Toggle *PID_TORQUE_TARGET* -500 and +500. Increase the absolute value of *PID_TORQUE_TARGET* if the target current is too low to turn the DC motor.

To turn a DC motor in torque mode, there is no need for an encoder. To turn a DC motor in velocity mode or in position mode, an encoder is required. For a DC motor, the encoder setup is easier compared to the encoder setup for the FOC with stepper or BLDC.

4.3 PI Tuning

4.3.1 PI tuning torque mode

Open the *Torque Graph*, *Torque Mode*, and *PI control* tools. Incrementally increase the torque P (*PID_TORQUE_P*) value until the actual current reaches 50% to 75% of the target torque. Target torque current should be selected according to the motor rated current (example, a quarter of the rated current). Ideally, the motor remains stationary during this tuning.

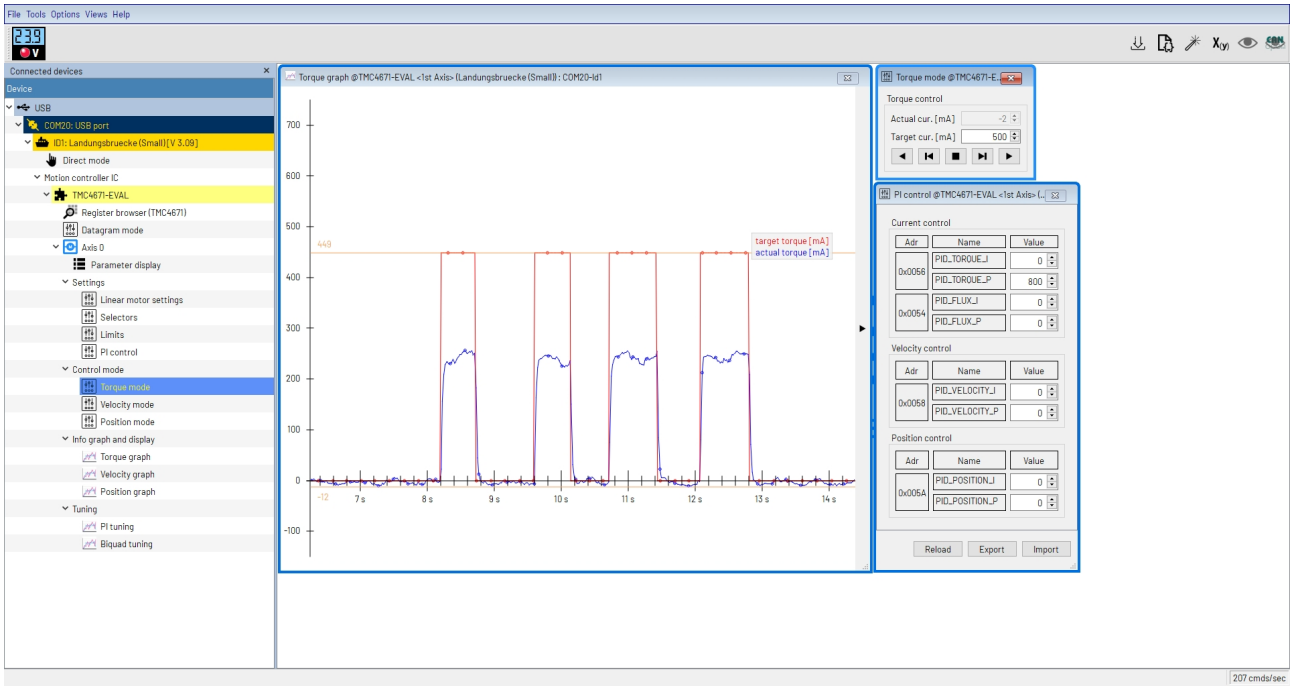

Figure 16: Torque Step Response, P Tuning

Incrementally increase the torque I (*PID_TORQUE_I*) value until the actual current reaches the target torque.

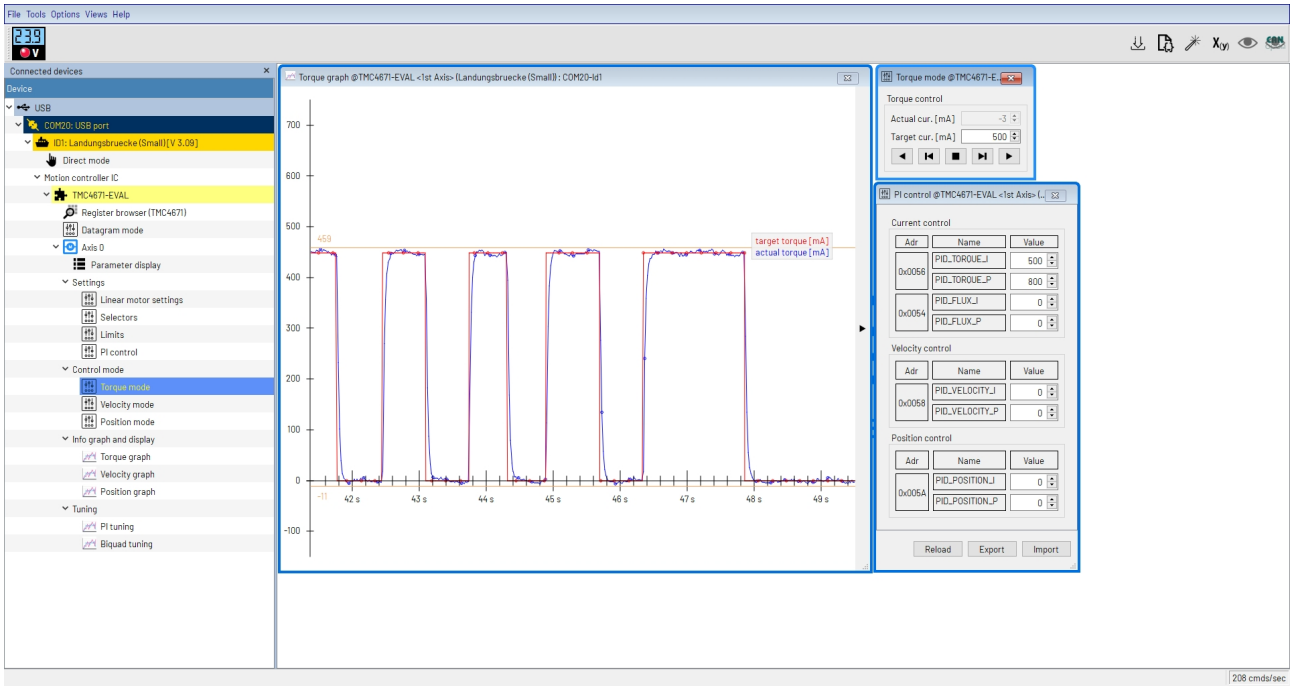

Figure 17: Torque Step Response, I Tuning

4.3.2 PI tuning velocity mode

Use the *PI tuning* tool to determine suitable P and I values:

1. Set a target velocity (example, 500 rpm) according to the application requirement.
2. Enable the ramp (recommended).
3. Configure the acceleration for the ramp. Rule of thumb: set it to ten times the target velocity.
4. Click *Start* to begin the PI tuning.

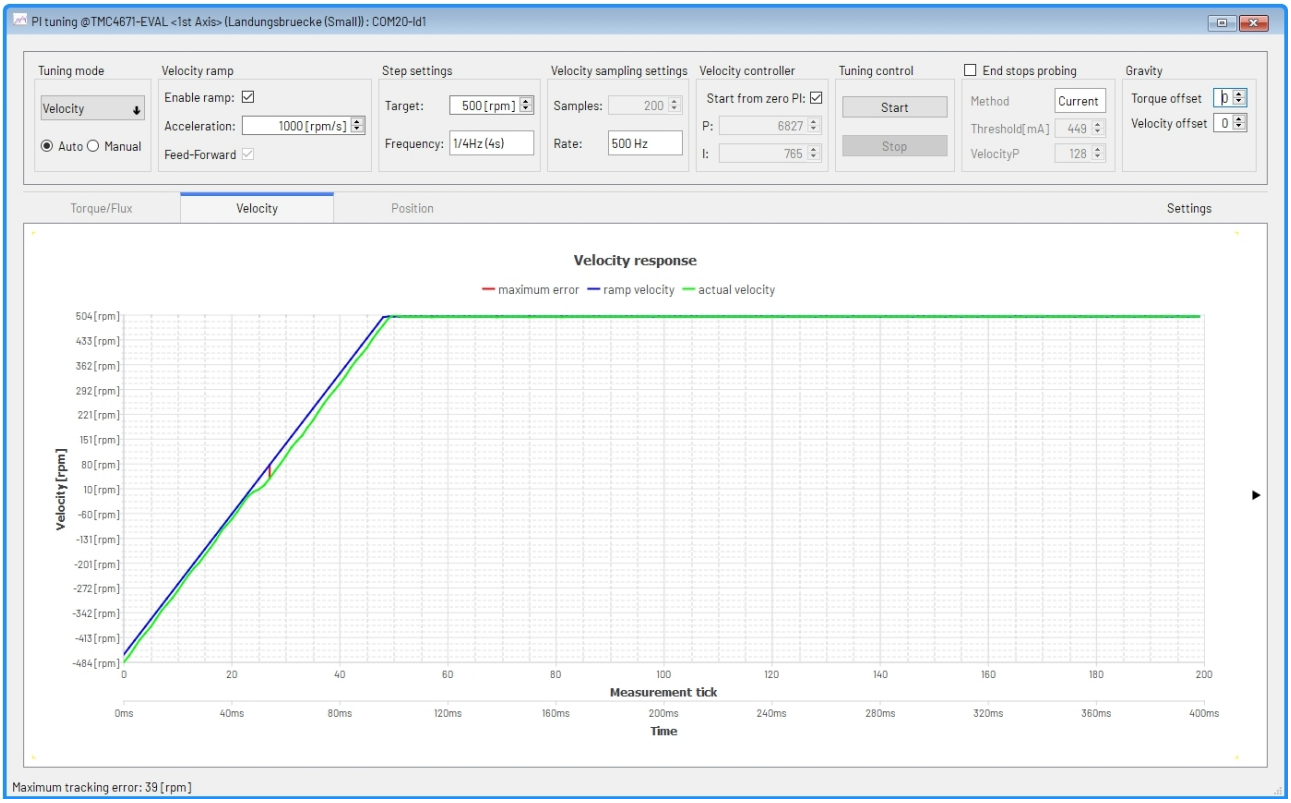

Figure 18: PI Tuning Tool, Automatic Mode

Verify and optimize PI values with manual mode:

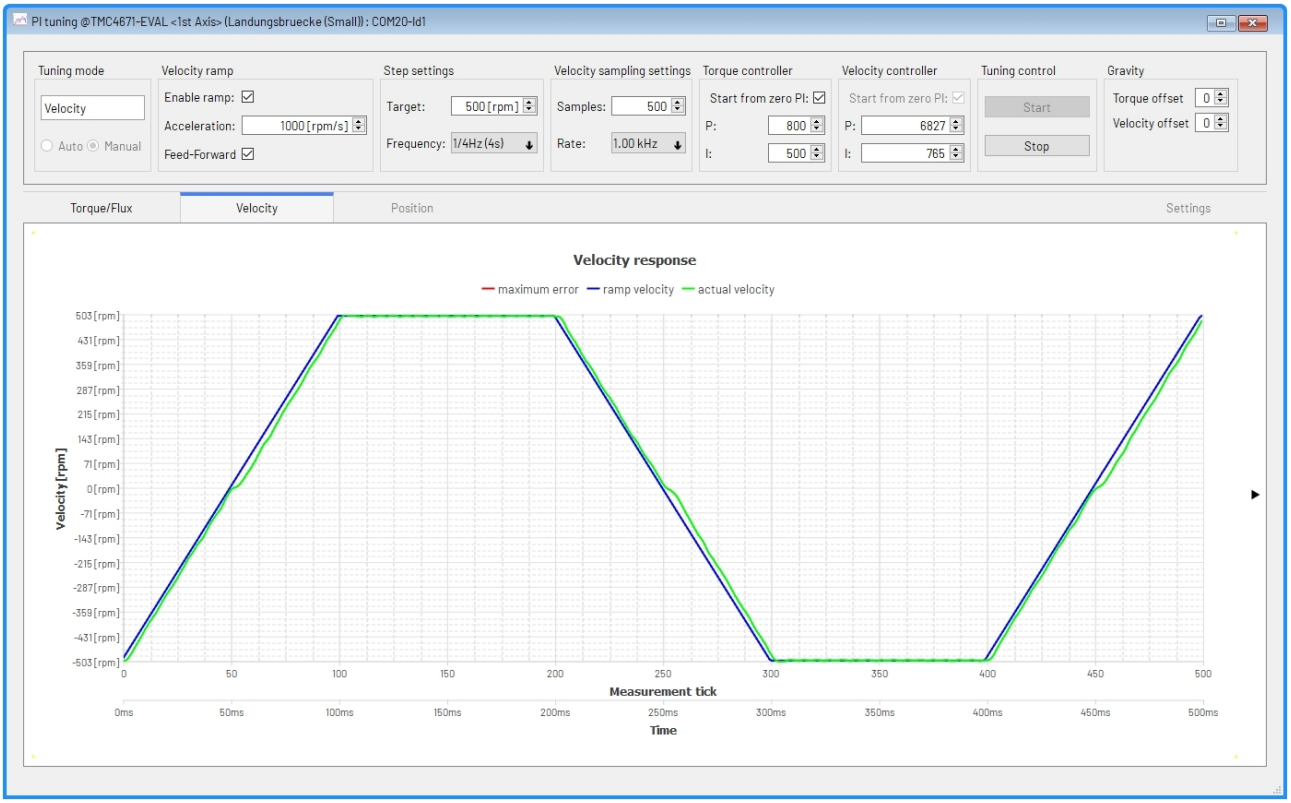

Figure 19: PI Tuning Tool, Manual Mode

4.3.3 PI tuning position mode

Use the *PI tuning* tool to determine a suitable P (*PID_POSITION_P*) value.

1. Set a target position (example, 65536 for one full revolution).
2. Click *Start* to begin the PI tuning.

NOTE In the TMC4671, positions are normalized to a fixed range. For a DC motor, one full motor revolution equals 65536 increments, regardless of the sensor feedback used.

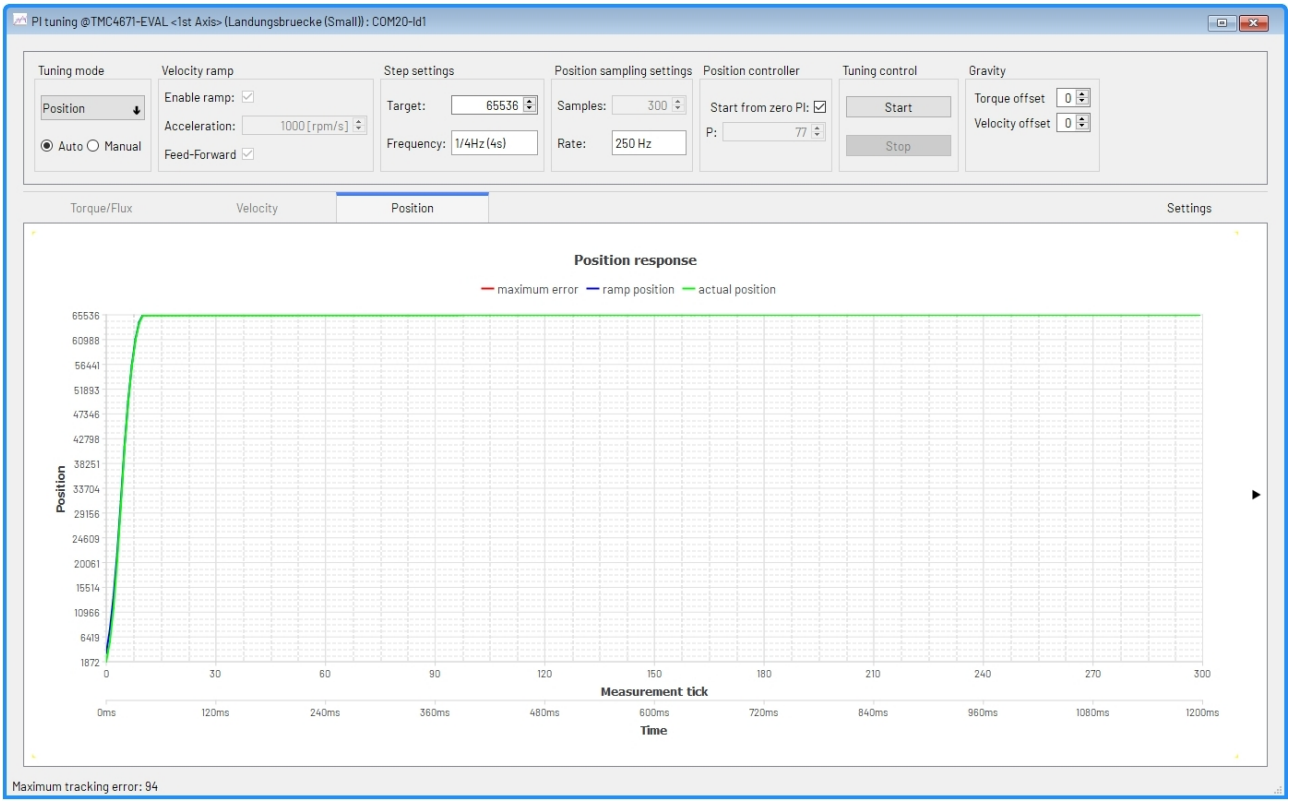

Figure 20: PI Tuning Tool, Automatic Mode

Verify and optimize PI values with manual mode:

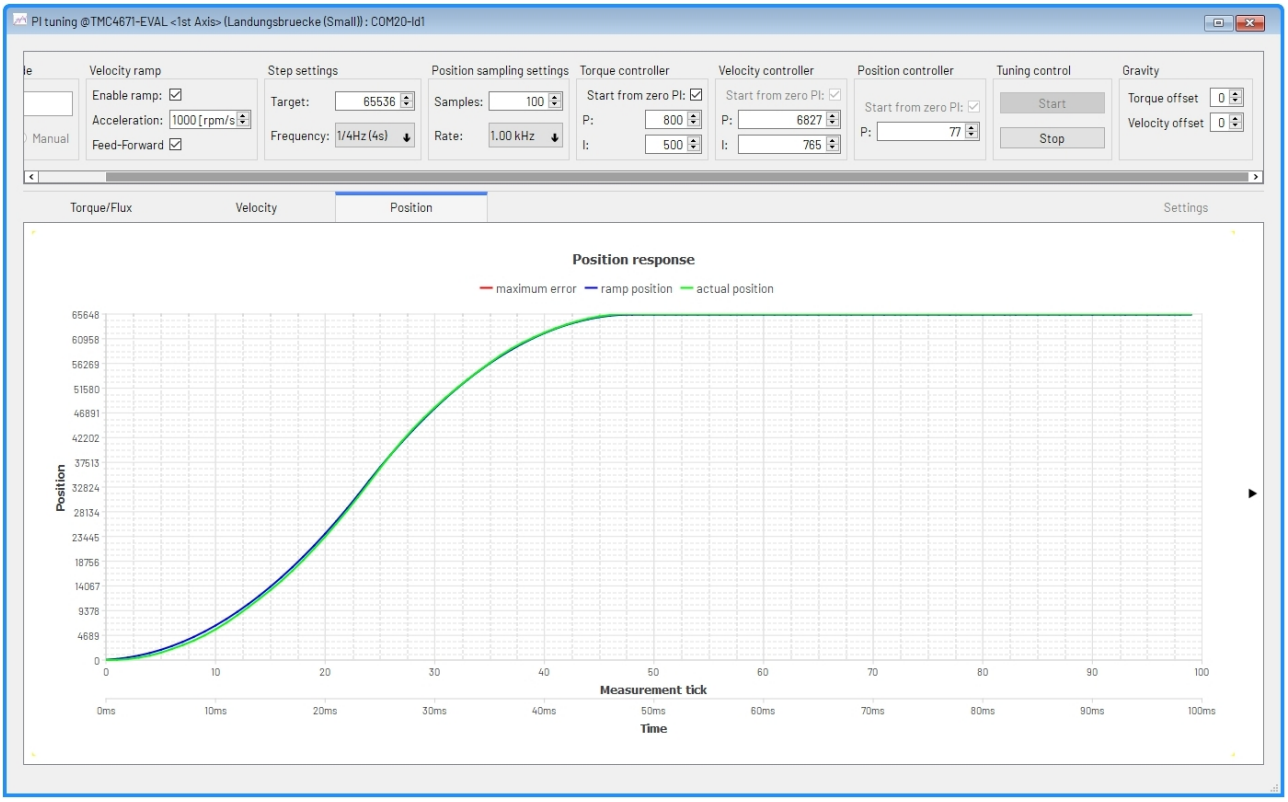

Figure 21: PI Tuning Tool, Manual Mode

4.3.4 TPC Script - DC Motor Closed Loop Position Mode

The following script shows an example configuration in closed loop position mode. Adapt motor/encoder configuration to own setup.

Listing 1: TPC Script Example

```
// Select module
#module 1 "COM20/USB/id1/Landungsbruecke (Small)"

// Use TMC4671 register addresses
#include TMC4671_register_addresses.tpc
// (C:/Users/user.name/AppData/Roaming/TRINAMIC Motion Control GmbH & Co. KG
    ↪ /TMCL-IDE/TMCL-Script/TMC4671_register_addresses.tpc)

// Motor type &  PWM configuration
WMC MOTOR_TYPE_N_POLE_PAIRS, 0, $00010001, 1
WMC PWM_POLARITIES,          0, $00000000, 1
WMC PWM_MAXCNT,              0, $00000F9F, 1
WMC PWM_BBM_H_BBM_L,         0, $00001414, 1
WMC PWM_SV_CHOP,             0, $00000007, 1

// ADC configuration
WMC ADC_I_SELECT,         0, $18000100, 1
WMC dsADC_MCFG_B_MCFG_A, 0, $00100010, 1
WMC dsADC_MCLK_A,         0, $20000000, 1
WMC dsADC_MCLK_B,         0, $20000000, 1
WMC dsADC_MDEC_B_MDEC_A, 0, $014E014E, 1
```

```
WMC ADC_I0_SCALE_OFFSET, 0, $0100820E, 1
WMC ADC_I1_SCALE_OFFSET, 0, $010081F2, 1

// ABN encoder settings
WMC ABN_DECODER_MODE,               0, $00001000, 1
WMC ABN_DECODER_PPR,                0, $00001000, 1
//WMC ABN_DECODER_COUNT,              0, $00000000, 1
WMC ABN_DECODER_PHI_E_PHI_M_OFFSET, 0, $00000000, 1

// Limits
WMC PID_TORQUE_FLUX_LIMITS, 0, $000003E8, 1
WMC PID_VELOCITY_LIMIT, 0, $000001F4, 1 //500rpm
SAP 4, 0, 500, 1    //set maximum speed : TMC4671
SAP 11, 0, 1000, 1    //set acceleration : TMC4671

// PI settings
WMC PID_FLUX_P_FLUX_I, 0, $00000000, 1
WMC PID_TORQUE_P_TORQUE_I, 0, $032001F4, 1
WMC PID_VELOCITY_P_VELOCITY_I, 0, $1AAB02FD, 1
WMC PID_POSITION_P_POSITION_I, 0, $004D0000, 1
SAP 52, 0, 1, 1    //set enable velocity feed forward : TMC4671

// ===== ABN encoder test drive =====
WMC PID_POSITION_TARGET, 0, $0, 1
WMC PID_POSITION_ACTUAL, 0, $0, 1
SAP 179, 0, 0, 1    //set actual position

SAP 15, 0, 50, 1    //set max position deviation : TMC4671

// Feedback selection
WMC PHI_E_SELECTION, 0, $00000003, 1
WMC VELOCITY_SELECTION, 0, $00000009, 1

// Switch to position mode
WMC MODE_RAMP_MODE_MOTION, 0, $00000003, 1

// Rotate right
MVP ABS, 0, 655360, 1
WAIT POS, 0, 0, 1

// Rotate left
MVP ABS, 0, 0, 1
WAIT POS, 0, 0, 1
```

5 References

- TM4671 product page
- TMC4671 PI tuning appnote
- Driving a linear stage with TMC4671
- TMC4671 API on github
- TMC4671 Python resources

6 Revision History

Revision Number	Revision Date	Description	Number of Pages Changed
Rev. 0	03/26	Initial release	-

Table 5: Document Revision