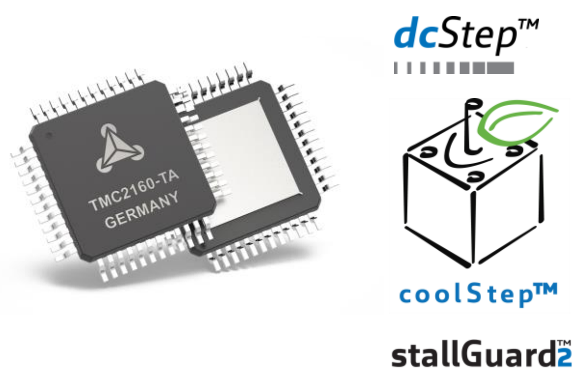
POWER DRIVER FOR STEPPER MOTORS

INTEGRATED CIRCUITS

TMC2160 / TMC2160A DATASHEET

Universal high voltage driver for two-phase bipolar stepper motor. StealthChop™ for quiet movement. External MOSFETs for 1A to several 10A coil current. With Step/Dir Interface and SPI.



APPLICATIONS

- Robotics & Industrial Drives
- Textile, Sewing Machines
- Packing Machines
- Factory & Lab Automation
- High-speed 3D Printers
- Liquid Handling
- Medical
- Office Automation
- CCTV
- ATM, Cash Recycler
- Pumps and Valves

FEATURES AND BENEFITS

2-phase stepper motors from 1A to several 10A coil current

Step/Dir Interface with microstep interpolation **MicroPlyer™**

Voltage Range 8 … 60V DC

SPI Interface

Highest Resolution 256 microsteps per full step

StealthChop2™ for quiet operation and smooth motion

Resonance Dampening for mid-range resonances

SpreadCycle™ highly dynamic motor control chopper

DcStep™ load dependent speed control

StallGuard2™ high precision sensorless motor load detection

CoolStep™ current control for energy savings up to 75%

Passive Braking and freewheeling mode

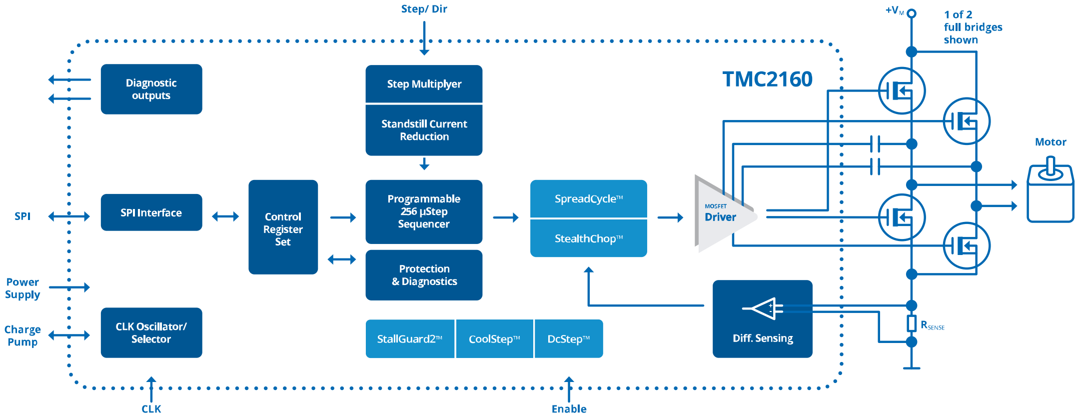
Full Protection & Diagnostics

Compact Size 7x7mm^2 (body) TQFP48 package

DESCRIPTION

The TMC2160 / TMC2160A is a high-power stepper motor driver IC with SPI interface. It features industries' most advanced stepper motor driver with simple Step / Direction interface. Using external transistors, highly dynamic, high torque drives can be realized. Based on TRINAMICs sophisticated SpreadCycle and StealthChop choppers, the driver ensures absolutely noiseless operation combined with maximum efficiency and best motor torque. High integration, high energy efficiency and a small form factor enable miniaturized and scalable systems for cost effective solutions. The fully compatible TMC5160 offers an additional motion controller to make stepper motor control even easier.

BLOCK DIAGRAM

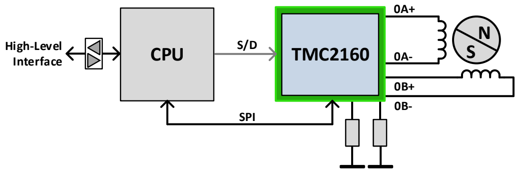


TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

APPLICATION EXAMPLES: HIGH VOLTAGE – MULTIPURPOSE USE

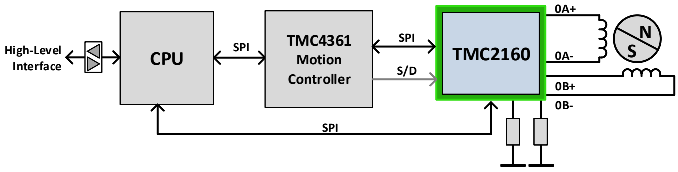
The TMC2160 scores with advanced motor commutation algorithms, combined with powerful external MOSFET driver stages, and high-quality current regulation. It offers a versatility that covers a wide spectrum of applications from battery powered high efficiency systems up to embedded applications with 20A motor current per coil. Based on TRINAMICs unique features StallGuard2, CoolStep, DcStep, SpreadCycle, and StealthChop, the TMC2160 optimizes drive performance. It trades off velocity vs. motor torque, optimizes energy efficiency, smoothness of the drive, and noiselessness. The small form factor of the TMC2160 keeps costs down and allows for miniaturized layouts. Extensive support at the chip, board, and software levels enables rapid design cycles and fast time-to-market with competitive products. High energy efficiency and reliability deliver cost savings in related systems such as power supplies and cooling. For smaller designs, the software compatible, fully integrated TMC2130 driver provides up to 1.4A of motor current.

MINIATURIZED DESIGN FOR ONE STEPPER MOTOR

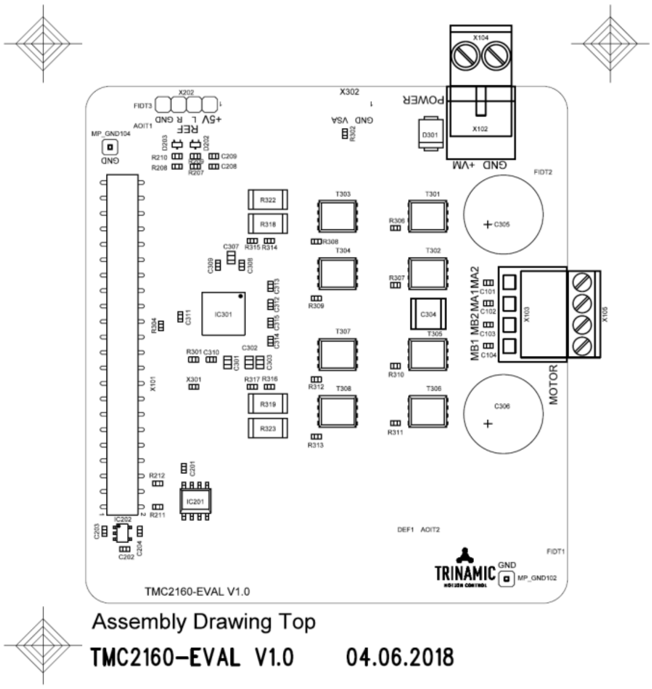


In this application, the CPU initializes the TMC2160 motor driver via SPI interface and controls motor movement by sending step and direction signals. A real time software realizes motion control.

DESIGN FOR DEMANDING APPLICATIONS WITH S-SHAPED RAMP PROFILES



The CPU initializes the TMC4361 motion controller and the TMC2160. Thereafter, it sends target positions to the TMC4361. Now, the TMC4361 takes control over the TMC2160. Combining the TMC4361 and the TMC2160 offers diverse possibilities for demanding applications including servo drive features.



The TMC2160-EVAL is part of TRINAMICs universal evaluation board system which provides a convenient handling of the hardware as well as a user-friendly software tool for evaluation. The TMC2160 evaluation board system consists of three parts: LANDUNGSBRÜCKE (base board), ESELSBRÜCKE (connector board including several test points), and TMC2160-EVAL.

Hint: TMC2160 in this manual always refers to both, the TMC2160A and TMC2160, unless explicitly noted with "*non-A-version*" or "*A-version*". The *A-version* compatibly replaces the *non-A-version*.

ORDER CODES

Order code	**Description**	**Size** [mm²]
TMC2160A-TA	Stepper Motor Driver IC, SPI, Step/Dir, UART, 8-60V Supply, 1.4A, TQFP48, Tray	7 x 7 (body)
TMC2160A-TA-T	Stepper Motor Driver IC, SPI, Step/Dir, UART, 8-60V Supply, 1.4A, TQFP48, Tape & Reel	7 x 7 (body)
TMC2160-EVAL-KIT	Full Evaluation Kit for TMC2160	126 x 85
TMC2160-EVAL	Evaluation Board for TMC2160 (excl. Landungsbrücke and Eselsbrücke)	85 x 55
TMC2160-BOB	Breakout Board with TMC2160	38 x 28

Table of Contents

1 Principles of Operation

The TMC2160 driver chip is an intelligent power component interfacing between a motion controller and a high-power stepper motor. It uses StealthChop, DcStep, CoolStep, and StallGuard2 automatically to optimize every motor movement. The TMC2160 ideally extends the TMC2100 and TMC2130 family to higher voltages and higher motor currents.

THE TMC2160 OFFERS TWO BASIC MODES OF OPERATION:

MODE 1: Step & Direction Driver

An external high-performance S-ramp motion controller like the TMC4361 or a central CPU generates step & direction signals synchronized to other components like additional motors within the system. The TMC2160 takes care of intelligent current and mode control and delivers feedback on the state of the motor. The MicroPlyer automatically smoothens motion.

MODE 2: Simple Step & Direction Driver

The TMC2160 positions the motor based on step & direction signals. The MicroPlyer automatically smoothens motion. No CPU interaction is required; configuration is done by hardware pins. Basic standby current control can be done by the TMC2160. Optional feedback signals allow error detection and synchronization. Enable this mode by tying pin SPI_MODE low.

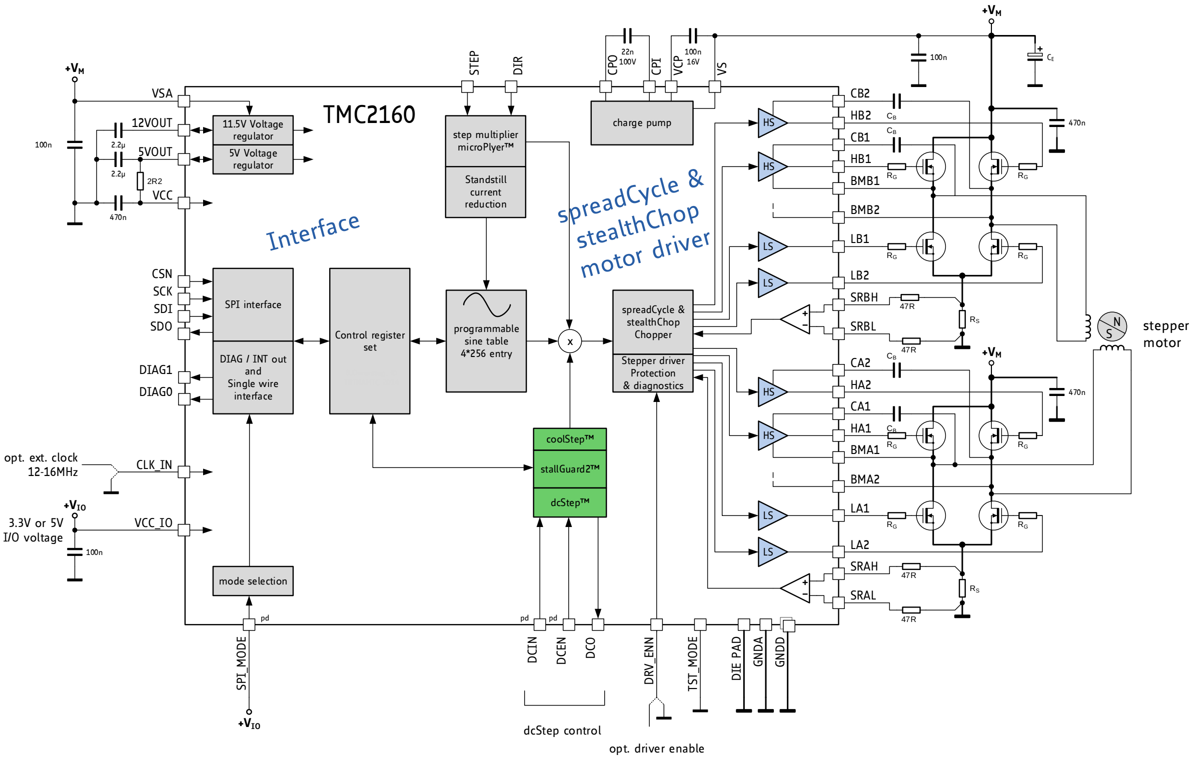


Figure 1.1 TMC2160 STEP/DIR application diagram

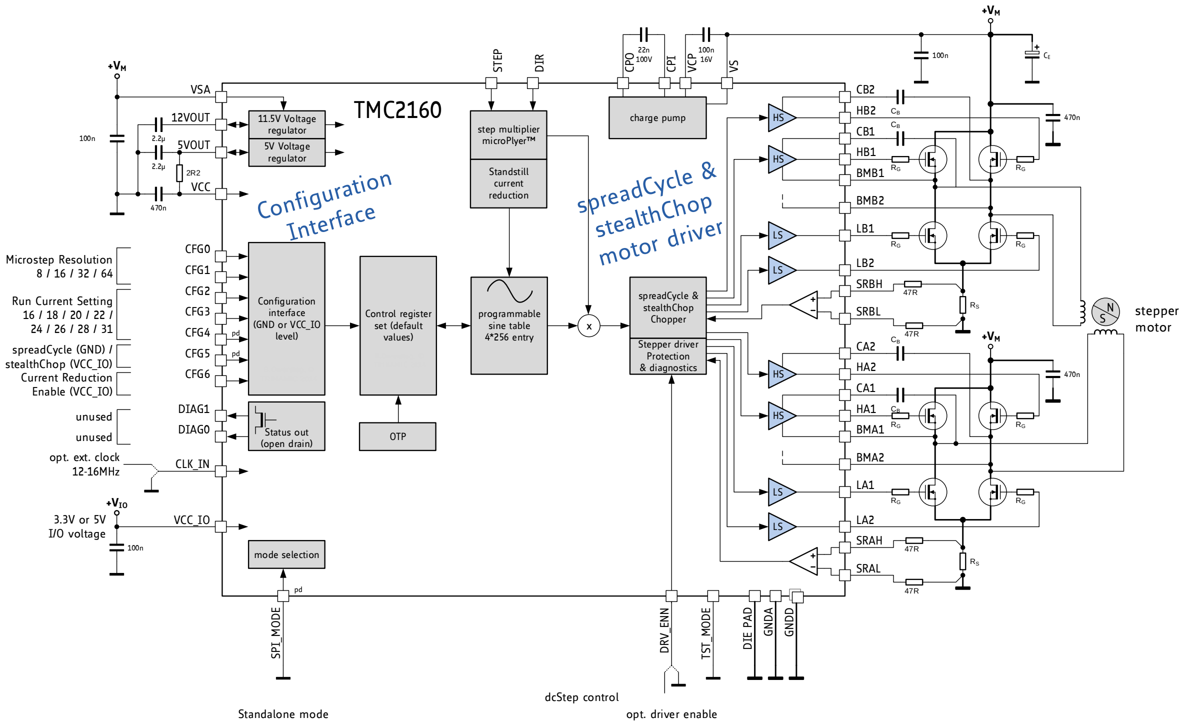


Figure 1.2 TMC2160 standalone driver application diagram

1.1 Key Concepts

The TMC2160 implements advanced features which are exclusive to TRINAMIC products. These features contribute toward greater precision, greater energy efficiency, higher reliability, smoother motion, and cooler operation in many stepper motor applications.

StealthChop2™	No-noise, high-precision chopper algorithm for inaudible motion and inaudible standstill of the motor. Allows faster motor acceleration and deceleration than StealthChop™ and extends StealthChop to low stand still motor currents.
SpreadCycle™	High-precision chopper algorithm for highly dynamic motion and absolutely clean current wave. Low noise, low resonance and low vibration chopper.
DcStep™	Load dependent speed control. The motor moves as fast as possible and never loses a step.
StallGuard2™	Sensorless stall detection and mechanical load measurement.
CoolStep™	Load-adaptive current control reducing energy consumption by as much as 75%.
MicroPlyer™	Microstep interpolator for obtaining full 256 microstep smoothness with lower resolution step inputs starting from fullstep

In addition to these performance enhancements, TRINAMIC motor drivers offer safeguards to detect and protect against shorted outputs, output open circuit, overtemperature, and undervoltage conditions for enhancing safety and recovery from equipment malfunctions.

1.2 Control Interfaces

The TMC2160 supports an SPI interface for parameter setting and diagnostics. Additionally, a standalone mode is provided for pure STEP/DIR operation without use of the serial interface. Selection of the actual interface is done via the configuration pin SPI_MODE, which can be hardwired to GND or VCC_IO depending on the desired interface.

1.2.1 SPI Interface

The SPI interface is a bit-serial interface synchronous to a bus clock. For every bit sent from the bus master to the bus slave another bit is sent simultaneously from the slave to the master. Communication between an SPI master and the TMC2160 slave always consists of sending one 40-bit command word and receiving one 40-bit status word.

The SPI command rate typically is a few commands per complete motor motion.

1.3 Software

From a software point of view the TMC2160 is a peripheral with a number of control and status registers. Most of them can either be written only or read only. Some of the registers allow both read and write access. In case read-modify-write access is desired for a write only register, a shadow register can be realized in master software.

1.4 Moving the Motor

1.4.1 STEP/DIR Interface

The motor is controlled by a step and direction input. Active edges on the STEP input can be rising edges or both rising and falling edges as controlled by another mode bit (*dedge*). Using both edges cuts the toggle rate of the STEP signal in half, which is useful for communication over slow interfaces such as optically isolated interfaces. On each active edge, the state sampled from the DIR input determines whether to step forward or back. Each step can be a fullstep or a microstep, in which there are 2, 4, 8, 16, 32, 64, 128, or 256 microsteps per fullstep. A step impulse with a low state on DIR increases the microstep counter and a high state decreases the counter by an amount controlled by the microstep resolution. An internal table translates the counter value into the sine and cosine values which control the motor current for microstepping.

1.4.2 SPI direct mode

The direct mode allows control of both motor coil currents and polarity via SPI. It mainly is intended for use with a dedicated external motion controller IC with integrated sequencer. The sequencer applies sine and cosine waves to the motor coils. This mode is specially designed for combination with the TMC4361 motion controller.

1.5 Automatic Standstill Power Down

An automatic current reduction drastically reduces application power dissipation and cooling requirements. Modify stand still current, delay time and decay via register settings. Automatic freewheeling and passive motor braking are provided as an option for stand still. Passive braking reduces motor standstill power consumption to zero, while still providing effective dampening and braking! An option for faster detection of standstill is provided for use with highly frequent motion commands.

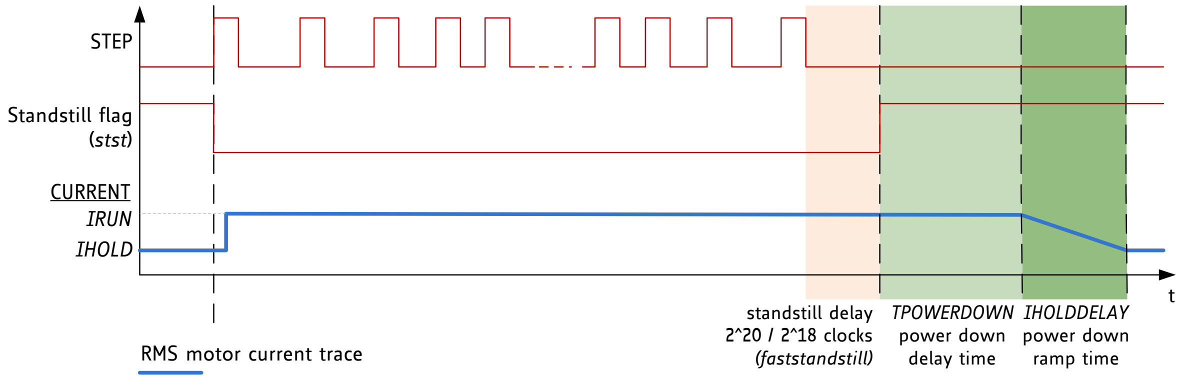


Figure 1.3 Automatic Motor Current Power Down

1.6 StealthChop2 & SpreadCycle Driver

StealthChop is a voltage chopper-based principle. It especially guarantees that the motor is absolutely quiet in standstill and in slow motion, except for noise generated by ball bearings. Unlike other voltage mode choppers, StealthChop2 does not require any configuration. It automatically learns the best settings during the first motion after power up and further optimizes the settings in subsequent motions. An initial homing sequence is sufficient for learning. Optionally, initial learning parameters can be pre-configured via the interface. StealthChop2 allows high motor dynamics, by reacting at once to a change of motor velocity.

For highest dynamic applications, SpreadCycle is an option to StealthChop2. It can be enabled via input pin (standalone mode) or via SPI interface. StealthChop2 and SpreadCycle may even be used in a combined configuration for the best of both worlds: StealthChop2 for no-noise stand still, silent and smooth performance, SpreadCycle at higher velocity for high dynamics and highest peak velocity at low vibration.

SpreadCycle is an advanced cycle-by-cycle chopper mode. It offers smooth operation and good resonance dampening over a wide range of speed and load. The SpreadCycle chopper scheme automatically integrates and tunes fast decay cycles to guarantee smooth zero crossing performance.

Benefits of using StealthChop2:

- Significantly improved microstepping with low-cost motors
- Motor runs smooth and quiet
- Absolutely no standby noise
- Reduced mechanical resonance yields improved torque

1.7 StallGuard2 – Mechanical Load Sensing

StallGuard2 provides an accurate measurement of the load on the motor. It can be used for stall detection as well as other uses at loads below those which stall the motor, such as CoolStep load-adaptive current reduction. This gives more information on the drive allowing functions like sensorless homing and diagnostics of the drive mechanics.

1.8 CoolStep – Load Adaptive Current

CoolStep drives the motor at the optimum current. It uses the StallGuard2 load measurement information to adjust the motor current to the minimum amount required in the actual load situation. This saves energy and keeps the components cool.

Benefits are:

- *Energy efficiency*	power consumption decreased up to 75%
- *Motor generates less heat*	improved mechanical precision
- *Less or no cooling*	improved reliability
- *Use of smaller motor*	less torque reserve required → cheaper motor does the job

Figure 1.4 shows the efficiency gain of a 42mm stepper motor when using CoolStep compared to standard operation with 50% of torque reserve. CoolStep is enabled above 60RPM in the example.

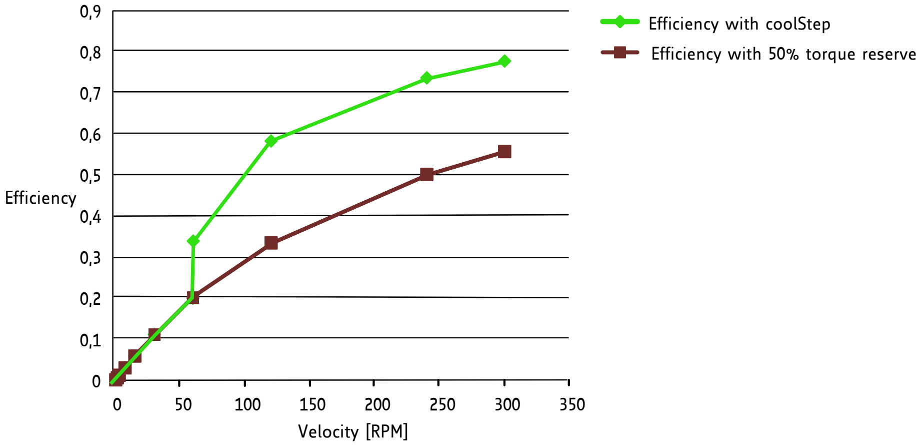


Figure 1.4 Energy efficiency with CoolStep (example)

1.9 DcStep – Load Dependent Speed

DcStep allows the motor to run near its load limit and at its velocity limit without losing a step. If the mechanical load on the motor increases to the stalling load, the motor automatically decreases velocity so that it can still drive the load. With this feature, the motor will never stall. In addition to the increased torque at a lower velocity, dynamic inertia will allow the motor to overcome mechanical overloads by decelerating. DcStep directly integrates with the ramp generator, so that the target position will be reached, even if the motor velocity needs to be decreased due to increased mechanical load. A dynamic range of up to factor 10 or more can be covered by DcStep without any step loss. By optimizing the motion velocity in high load situations, this feature further enhances overall system efficiency.

Benefits are:

- Motor does not loose steps in overload conditions
- Application works as fast as possible
- Highest possible acceleration automatically
- Highest energy efficiency at speed limit
- Highest possible motor torque using fullstep drive
- Cheaper motor does the job

2 Pin Assignments

2.1 Package Outline

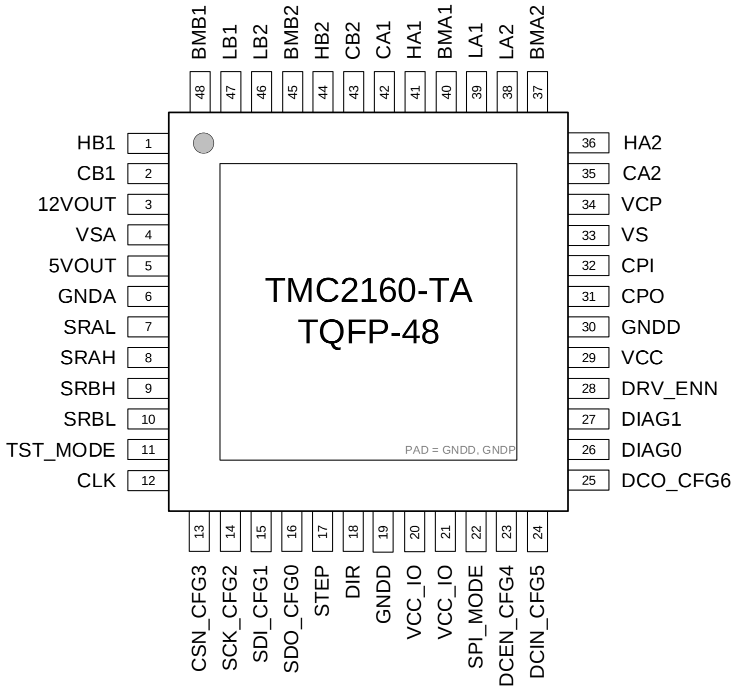


Figure 2.1 TMC2160-TA package and pinning TQFP-EP 48 (7x7mm² body, 9x9mm² with leads)

2.2 Signal Descriptions

Pin	TQFP	Type	Function
HB1	1		High side gate driver output.
CB1	2		Bootstrap capacitor positive connection.
12VOUT	3		Output of internal 11.5V gate voltage regulator and supply pin of low side gate drivers. Attach 2.2µF to 10µF ceramic capacitor to GND plane near to pin for best performance. Use at least 10 times more capacity than for bootstrap capacitors. In case an external gate voltage supply is available, tie VSA and 12VOUT to the external supply.
VSA	4		Analog supply voltage for 11.5V and 5V regulator. Normally tied to VS. Provide a 100nF filtering capacitor.
5VOUT	5		Output of internal 5V regulator. Attach 2.2µF to 10µF ceramic capacitor to GNDA near to pin for best performance. Output for VCC supply of the chip.
GNDA	6		Analog GND. Connect to GND plane near pin.
SRAL	7	AI	Sense resistor GND connection for phase A. Connect to the GND side of the sense resistor to compensate for voltage drop on the GND interconnection.

Pin	**TQFP**	**Type**	**Function**
SRAH	8	AI	Sense resistor for phase A. Connect to the upper side of the sense resistor. A Kelvin connection is preferred with high motor currents. Symmetrical RC-Filtering may be added for SRAL and SRAH to eliminate high frequency switching spikes from other drives or switching of coil B.
SRBH	9	AI	Sense resistor for phase B. Connect to the upper side of the sense resistor. A Kelvin connection is preferred with high motor currents. Symmetrical RC-Filtering may be added for SRBL and SRBH to eliminate high frequency switching spikes from other drives or switching of coil A.
SRBL	10	AI	Sense resistor GND connection for phase B. Connect to the GND side of the sense resistor to compensate for voltage drop on the GND interconnection.
TST_MODE	11	DI	Test mode input. Tie to GND using short wire.
CLK	12	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock. Internal clock-fail over circuit protects against loss of external clock signal.
CSN_CFG3	13	DI	SPI chip select input (negative active) (SPI_MODE=1) or Configuration input (SPI_MODE=0)
SCK_CFG2	14	DI	SPI serial clock input (SPI_MODE=1) or Configuration input (SPI_MODE=0)
SDI_CFG1	15	DI	SPI data input (SPI_MODE=1) or Configuration input (SPI_MODE=0) or Next address input (NAI) for single wire interface.
SDO_CFG0	16	DIO	SPI data output (tristate) (SPI_MODE=1) or Configuration input (SPI_MODE=0) or Next address output (NAO) for single wire interface.
STEP	17	DI	STEP input
DIR	18	DI	DIR input
GNDD	19, 30		Digital GND. Connect to GND plane near pin.
VCC_IO	20, 21		3.3V to 5V IO supply voltage for all digital pins. Does not supply internal logic circuitry.
SPI_MODE	22	DI (pd)	Mode selection input. When tied low, the chip is in standalone mode and pins have their CFG functions. When tied high, the SPI interface is enabled. Integrated pull down resistor.
DCEN_ CFG4	23	DI (pd)	DcStep enable input (SPI_MODE=1) – leave open or tie to GND for normal operation in this mode (no DcStep). Configuration input (SPI_MODE=0)
DCIN_ CFG5	24	DI (pd)	DcStep gating input for axis synchronization (SPI_MODE=1) or Configuration input (SPI_MODE=0)
DCO_ CFG6	25	DIO	DcStep ready output. Configuration input (SPI_MODE=0)
DIAG0	26	DO (pu+ pd)	Diagnostics output DIAG0. Interrupt output Use external pullup resistor with 47k or less in open drain mode.
DIAG1	27	DO (pd)	Diagnostics output DIAG1. Use external pullup resistor with 47k or less in open drain mode.
DRV_ENN	28	DI	Enable input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.

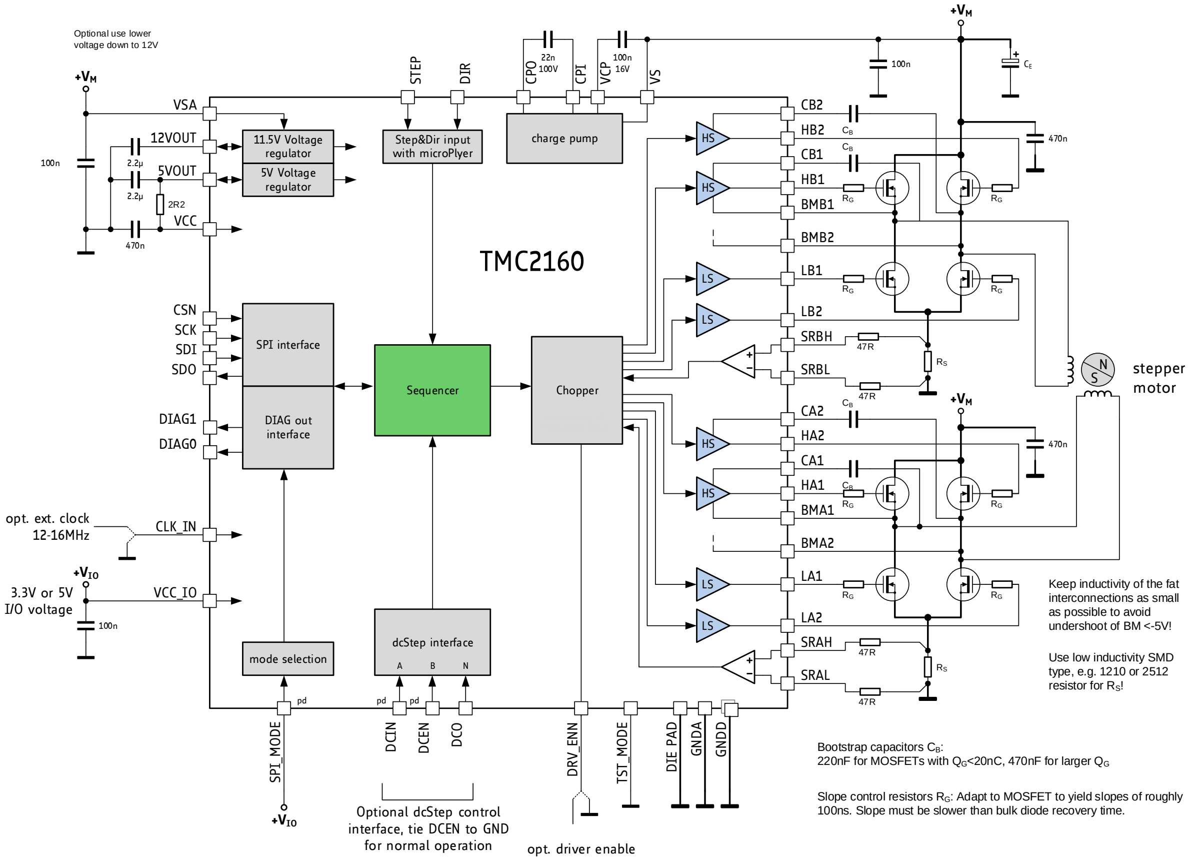
Pin	TQFP	Type	Function
VCC	29		5V supply input for digital circuitry within chip. Provide 100nF or bigger capacitor to GND (GND plane) near pin. Shall be supplied by 5VOUT. A 2.2 or 3.3 Ohm resistor is recommended for decoupling noise from 5VOUT.
CPO	31		Charge pump capacitor output.
CPI	32		Charge pump capacitor input. Tie to CPO using 22nF, 100V capacitor.
VS	33		Motor supply voltage. Provide filtering capacity near pin with short loop to GND plane. Must be tied to the positive bridge supply voltage.
VCP	34		Charge pump voltage. Tie to VS using 100nF capacitor.
CA2	35		Bootstrap capacitor positive connection.
HA2	36		High side gate driver output.
BMA2	37		Bridge Center and bootstrap capacitor negative connection.
LA2	38		Low side gate driver output.
LA1	39		Low side gate driver output.
BMA1	40		Bridge Center and bootstrap capacitor negative connection.
HA1	41		High side gate driver output.
CA1	42		Bootstrap capacitor positive connection.
CB2	43		Bootstrap capacitor positive connection.
HB2	44		High side gate driver output.
BMB2	45		Bridge Center and bootstrap capacitor negative connection.
LB2	46		Low side gate driver output.
LB1	47		Low side gate driver output.
BMB1	48		Bridge Center and bootstrap capacitor negative connection.
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for the low side gate drivers. Ensure low loop inductivity to sense resistor GND.

*(pd) denominates a pin with pulldown resistor
* All digital pins DI, DIO and DO use VCC_IO level and contain protection diodes to GND and VCC_IO
* All digital inputs DI and DIO have internal Schmitt-Triggers

3 Sample Circuits

The following sample circuits show the required external components in different operation and supply modes. The connection of the bus interface and further digital signals are left out for clarity.

3.1 Standard Application Circuit



Keep inductivity of the fat interconnections as small as possible to avoid undershoot of BM <-5V!

Use low inductivity SMD type, e.g. 1210 or 2512 resistor for R_S!

Bootstrap capacitors C_B:
220nF for MOSFETs with Q_G<20nC, 470nF for larger Q_G

Slope control resistors R_G: Adapt to MOSFET to yield slopes of roughly 100ns. Slope must be slower than bulk diode recovery time.

Figure 3.1 Standard application circuit

The standard application circuit uses a minimum set of additional components. Eight MOSFETs are selected for the desired current, voltage and package type. Two sense resistors set the motor coil current. See chapter 8 to choose the right value for sense resistors. Use low ESR capacitors for filtering the power supply. A minimum capacity of 100µF per ampere of coil current near to the power bridge is recommended for best performance. The capacitors need to cope with the current ripple caused by chopper operation. Current ripple in the supply capacitors also depends on the power supply internal resistance and cable length. VCC_IO can be supplied from 5VOUT, or from an external source, e.g., a 3.3V regulator. To minimize linear voltage regulator power dissipation of the internal 5V and 11.5V voltage regulators in applications where VM is high, a different (lower) supply voltage should be used for VSA (see chapter 0).

Basic layout hints
Place sense resistors and all filter capacitors as close as possible to the power MOSFETs. Place the TMC2160 near to the MOSFETs and use short interconnection lines to minimize parasitic trace inductance. Use a solid common GND for all GND, GNDA and GNDD connections, also for sense resistor GND. Connect 5VOUT filtering capacitor directly to 5VOUT and GNDA pin. See layout hints for more details. Low ESR electrolytic capacitors are recommended for VS filtering.

Hint
In safety critical applications, VS and the bridge may be supplied by a separate, switched supply to realize safe torque off. Make sure that the slope at VS does not exceed 1V/µs.

Attention

In case VSA is supplied by a different voltage source, make sure that VSA does not drop out during motor operation. Stop and disable the motor before VSA power down. This is not necessary, when VSA voltage is derived from VS supply, as both supplies go down in parallel in this case.

Attention

In addition to filtering capacity near to the power bridges, provide sufficient capacity on VS located close to the VS pin and the connection of the VCP capacitor, to ensure that high-frequency ripple, caused by the switching edges of the power bridge transistors are kept well below 0.5V. Keep power slopes below 1V/µs. Failure to do so could result in destructive currents via the charge pump circuit.

Provide overvoltage protection in case the motor could be manually turned at a high velocity, or in case the driver could become cut off from the main supply capacitors. Significant energy can be fed back from motor coils to the power supply in the event of quick deceleration, or when the driver becomes disabled.

3.2 External Gate Voltage Regulator

At high supply voltages like 48V, the internal gate voltage regulator and the internal 5V regulator have considerable power dissipation, especially with high MOSFET gate charges, high chopper frequency or high system clock frequency >12MHz. A good thermal coupling of the heat slug to the system PCB GND plane is required to dissipate heat. Still, the thermal thresholds will be lowered significantly by self-heating. To reduce power dissipation, supply an external gate driver voltage to the TMC2160. Figure 3.2 shows the required connection. The internal gate voltage regulator becomes disabled in this constellation. 12V +/-1V are recommended for best results.

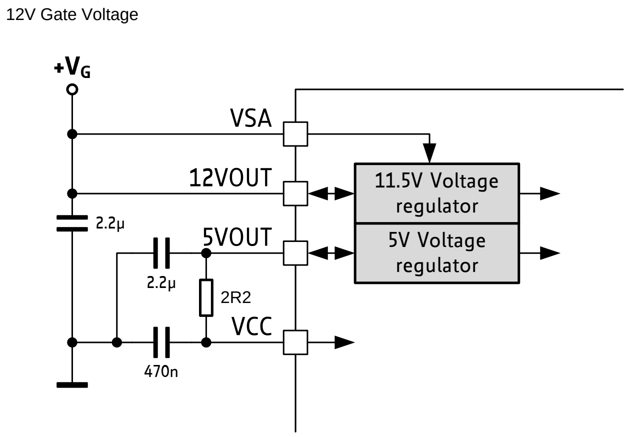


Figure 3.2 External gate voltage supply

Hint

With MOSFETs above 50nC of total gate charge, chopper frequency >40kHz, or at clock frequency >12MHz, it is recommended to use a VSA supply not higher than 40V.

3.3 Choosing MOSFETs and Slope

The selection of power MOSFETs depends on several factors, like package size, on-resistance, voltage rating and supplier. It is not true, that larger, lower RDSon MOSFETs will always be better, as a larger device also has higher capacitances and may add more ringing in trace inductance and power dissipation in the gate drive circuitry. Adapt the MOSFETs to the required motor voltage (adding 5-10V of reserve to the peak supply voltage) and to the desired maximum current, in a way that resistive power dissipation still is low for the thermal capabilities of the chosen MOSFET package. The TMC2160 drives the MOSFET gates with roughly 10V, so normal, 10V specified types are sufficient. Logic level FETs (4.5V specified RDSon) will also work, but may be more critical with regard to bridge cross-conduction due to lower $V_{GS(th)}$.

The gate drive current and MOSFET gate resistors R_G (optional) determine switching behavior and should basically be adapted to the MOSFET gate-drain charge (Miller charge). Figure 3.3 shows the influence of the Miller charge on the switching event. Figure 3.4 additionally shows the switching events in different load situations (load pulling the output up or down), and the required bridge brake-before-make time.
The following table shall serve as a thumb rule for programming the MOSFET driver current (*DRVSTRENGTH* setting) and the selection of gate resistors:

MOSFET MILLER CHARGE VS. *DRVSTRENGTH* AND R_G

Miller Charge [nC] (typ.)	*DRVSTRENGTH* setting	**Value of R_G [Ω]**
<10	0	≤ 15
10…20	0 or 1	≤ 10
20…40	1 or 2	≤ 7.5
40…60	2 or 3	≤ 5
>60	3	≤ 2.7

The TMC2160 provides increased gate-off drive current to avoid bridge cross-conduction induced by high dV/dt. This protection will be less efficient with gate resistors exceeding the values given in the table. Therefore, for larger values of R_G, a parallel diode may be required to ensure keeping the MOSFET safely off during switching events.

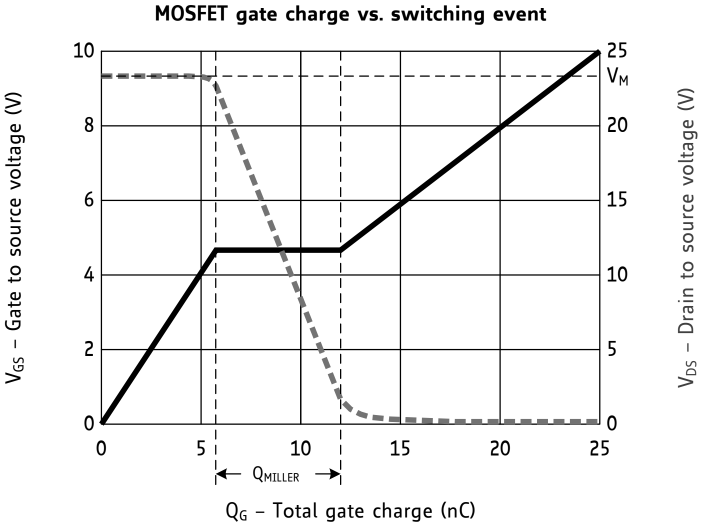


Figure 3.3 Miller charge determines switching slope

Hints
- Choose modern MOSFETs with fast and soft recovery bulk diode and low reverse recovery charge.
- A small, SMD MOSFET package allows compacter routing and reduces parasitic inductance effects.

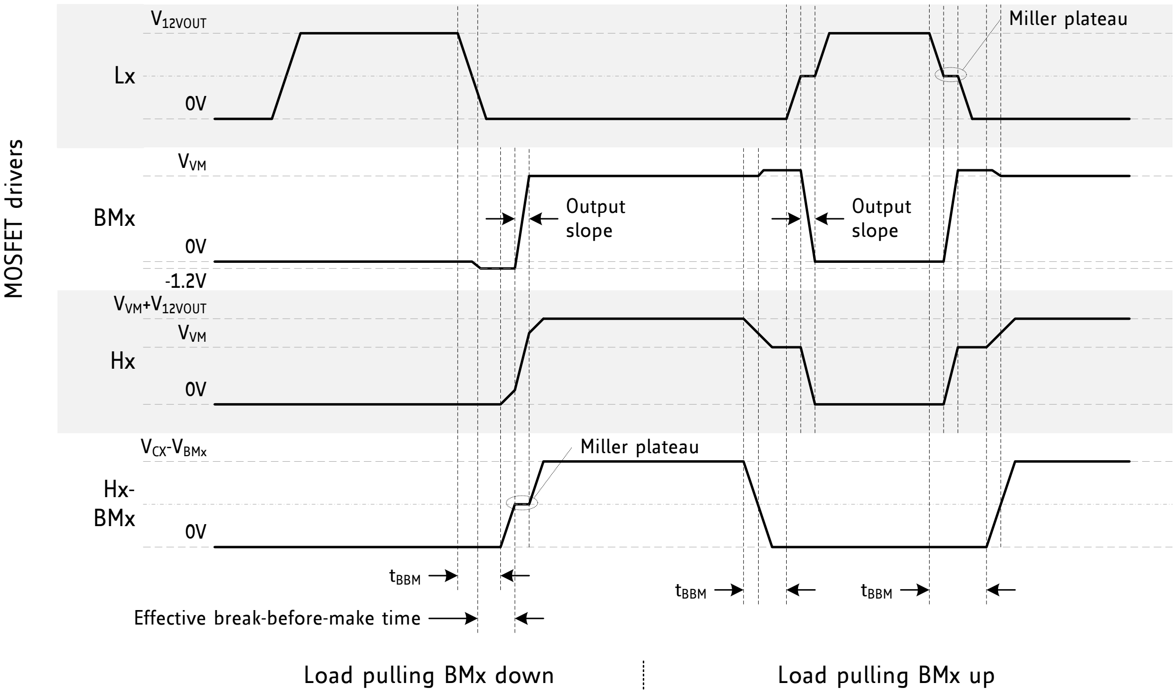


Figure 3.4 Slopes, Miller plateau and blank time

The following *DRV_CONF* parameters allow adapting the driver to the MOSFET bridge:

Parameter	Description	Setting	Comment
BBMTIME	Break-before-make time setting to ensure non-overlapping switching of high-side and low-side MOSFETs. *BBMTIME* allows fine tuning of times in increments shorter than a clock period. For higher times, use BBMCLKS.	0…24	time[ns]≈ 100ns*32/(32-*BBMTIME*) *Ensure ~30% headroom* *Reset Default: 0*
BBMCLKS	Like *BBMTIME*, but in multiple of a clock cycle. The longer setting rules (*BBMTIME* vs. *BBMCLKS*).	0…15	0: off *Reset Default: OTP 4 or 2*
DRV STRENGTH	Selection of gate driver current. Adapts the gate driver current to the gate charge of the external MOSFETs.	0…3	*Reset Default = 0*
FILT_ISENSE	Filter time constant of sense amplifier to suppress ringing and coupling from second coil operation *Hint:* Increase setting if motor chopper noise occurs due to cross-coupling of both coils. *(Reset Default = %00)*	0…3	00: ~100ns (*reset default*) 01: ~200ns 10: ~300ns 11: ~400ns

DRV_CONF Parameters

Use the lowest gate driver strength setting *DRVSTRENGTH* giving favorable switching slopes, before increasing the value of the gate series resistors. A slope time of nominal 40ns to 80ns is sufficient and will normally be covered by the shortest possible Break-Before-Make time setting (*BBMTIME*=0, *BBMCLKS*=0).

In case slower slopes have to be used, e.g., with large MOSFETs, ensure that the break-before-make time (*BBMTIME*, optionally use *BBMCLKS* for times >200ns) sufficiently covers the switching event, in order to avoid bridge cross conduction. The shortest break-before-make time, safely covering the switching event, gives best results. Add roughly 30% of reserve, to cover production stray of MOSFETs and driver.

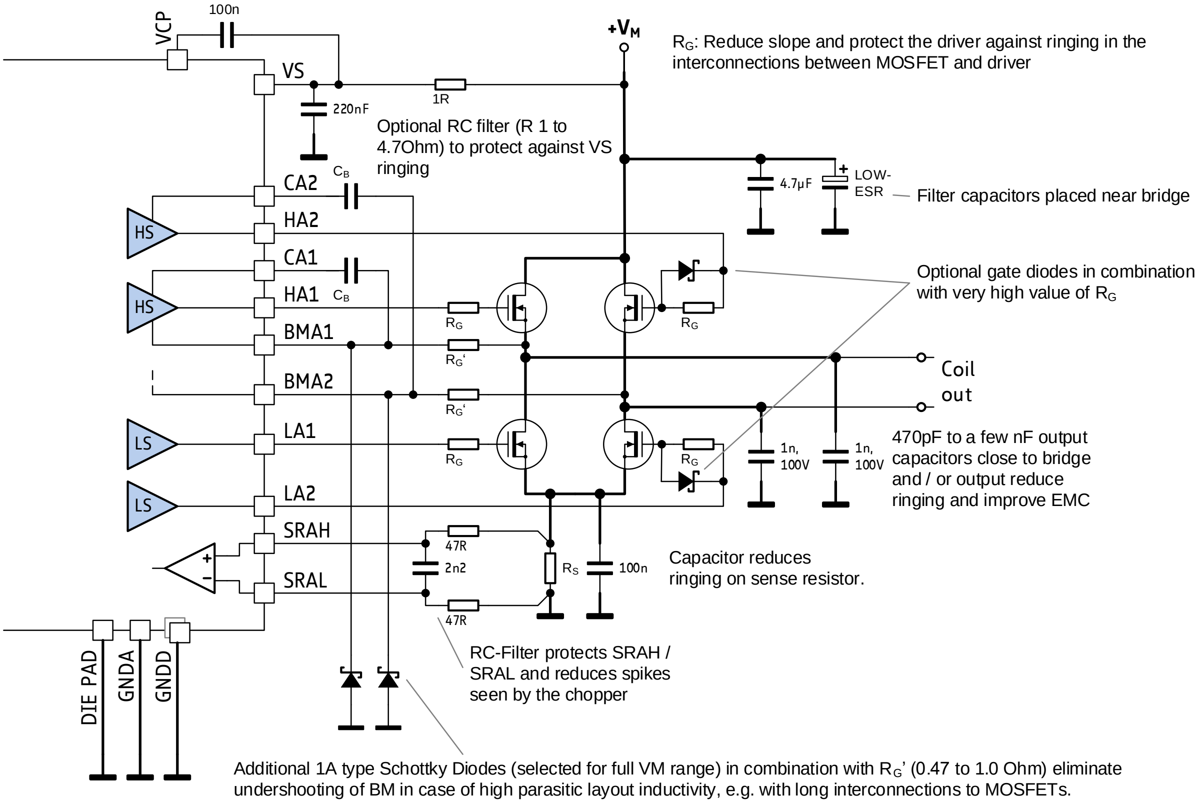
3.4 Tuning the MOSFET Bridge

A clean switching event is favorable to ensure low power dissipation and good EMC behavior. Unsuitable layout or components endanger stable operation of the circuit. Therefore, it is important to understand the effect of parasitic trace inductivity and MOSFET body diode reverse recovery.

Stray inductance in power routing will cause ringing whenever the opposite MOSFET is in diode conduction prior to switching on a low-side or high-side MOSFET. Diode conduction occurs during break-before make time while the load current is inverse to the following bridge polarity. The MOSFET bulk diode has a certain, type specific reverse recovery time and charge. This time typically is in the range of a few 10ns. During reverse recovery time, the bulk diode will cause high current flow across the bridge. This current is taken from the power supply filter capacitors (see thick lines Figure 3.5). Once the diode opens, parasitic inductance tries to keep the current flowing. A high, fast slope results and leads to ringing in parasitic inductivities in the current path (see Figure 3.6). This may lead to bridge voltage undershooting the GND level as well as short pulses on VS and all MOSFET connections. It must be ensured, that the driver IC does not see spikes on its BM pins undershooting GND more than 5V. Severe VS ripple might overload the charge-pump circuitry. Measure the voltage directly at the driver pins to driver GND. The amount of undershooting depends on energy stored in parasitic inductivities from low side drain to low side source and via the sense resistor RS to GND.

When using relatively small MOSFETs, a soft slope control requires a high gate series resistance. This endangers safe MOSFET switch off. Add additional diodes to ensure safe MOSFET off conditions in this case (shown for right MOSFET pair in Figure 3.5).

Figure 3.7 shows performance of the basic circuit after adapting switching slope and adding 1nF bridge output capacitors.



R$_G$: Reduce slope and protect the driver against ringing in the interconnections between MOSFET and driver

Optional RC filter (R 1 to 4.7Ohm) to protect against VS ringing

Filter capacitors placed near bridge

Optional gate diodes in combination with very high value of R$_G$

470pF to a few nF output capacitors close to bridge and / or output reduce ringing and improve EMC

Capacitor reduces ringing on sense resistor.

RC-Filter protects SRAH / SRAL and reduces spikes seen by the chopper

Additional 1A type Schottky Diodes (selected for full VM range) in combination with R$_G$' (0.47 to 1.0 Ohm) eliminate undershooting of BM in case of high parasitic layout inductivity, e.g. with long interconnections to MOSFETs.

Decide use and value of the additional components based on measurements of the actual circuit using the final layout!

Figure 3.5 Bridge protection options for power routing inductivity

ENSURE RELIABLE OPERATION

- Use SMD MOSFETs and short interconnections
- Provide sufficient power filtering capacity close to the bridge and close to VS pin & VCP capacitor
- Tune MOSFET switching slopes (measure switch-on event at MOSFET gate) to be slower than the MOSFET bulk diode reverse recovery time. This will reduce cross conduction.
- Add optional gate resistors close to MOSFET gate and output capacitors to ensure clean switching and reliable operation by minimizing ringing. Figure 3.5 shows the options plus some variations.
- Some MOSFETs eliminate reverse recovery charge by integrating a fast diode from source to drain.

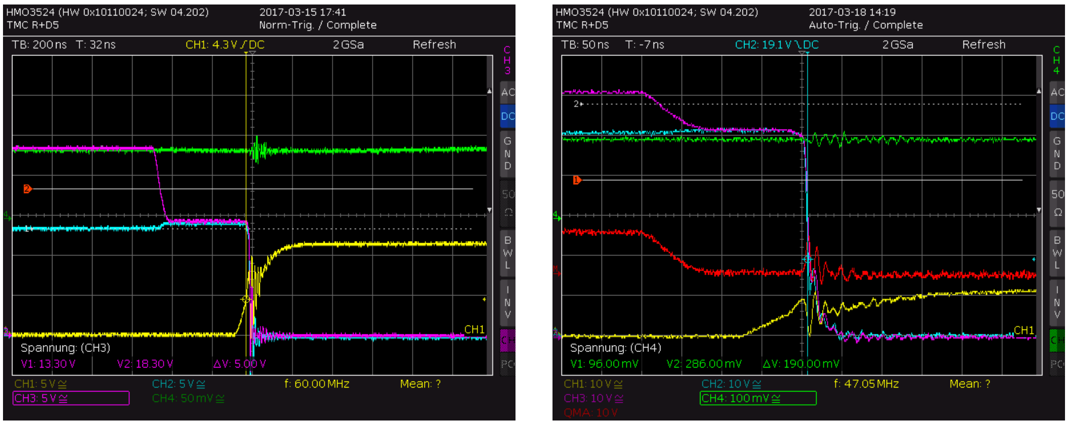


Figure 3.6 Ringing of output (blue) and Gate voltages (Yellow, Cyan) with untuned brige

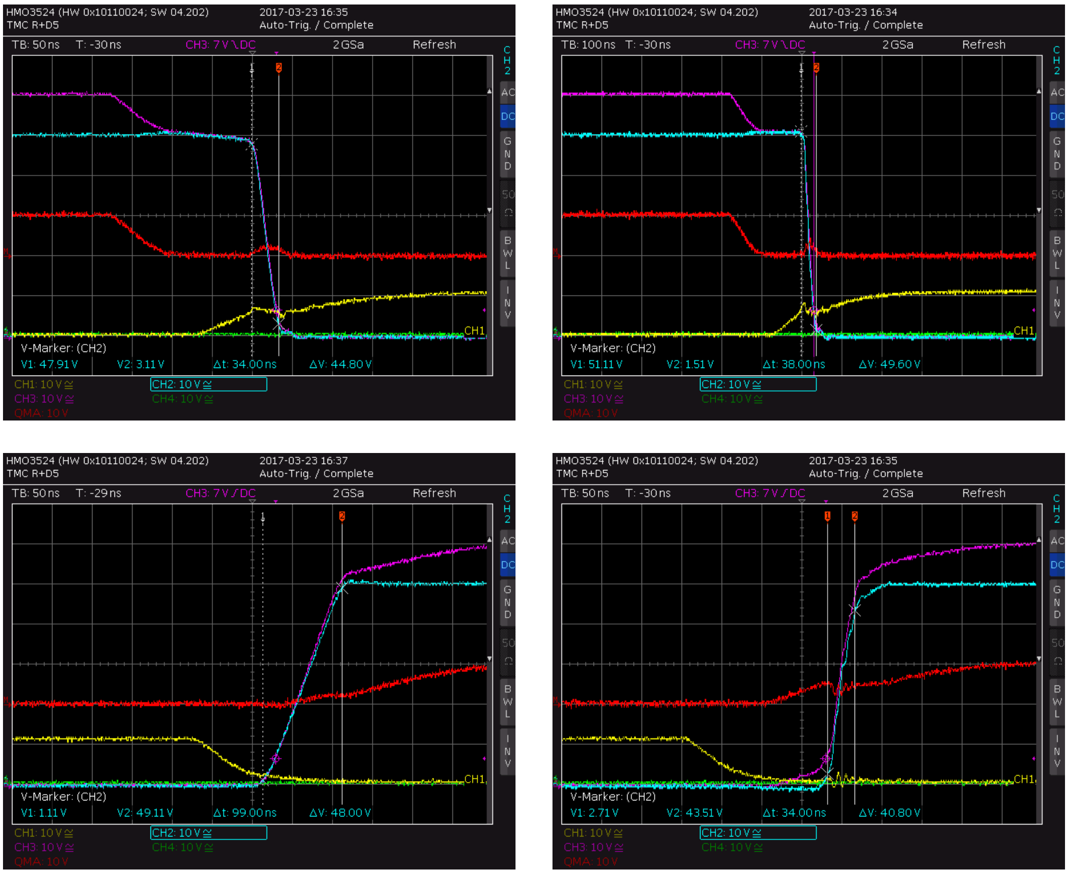


Figure 3.7 Switching event with optimized components (without / after bulk diode conduction)

BRIDGE OPTIMIZATION EXAMPLE

A stepper driver for 6A of motor current has been designed using the MOSFET AOD4126 in the standard schematic.

The MOSFETs have a low gate capacitance and offer roughly 50ns slope time at the lowest driver strength setting. At lowest driver strength setting, switching quality is best (Figure 3.6), but still shows a lot of ringing. Low side gate resistors have been added to slightly increase switching slope time following high-side bulk diode conduction by increasing the effect of Gate-Drain (Miller) charge. High side gate resistors have been added for symmetry. Tests showed, that 1nF output capacitors dramatically reduce ringing of the power bridge following bulk diode conduction (Figure 3.7). Figure 3.8 shows the actual components and values after optimization.

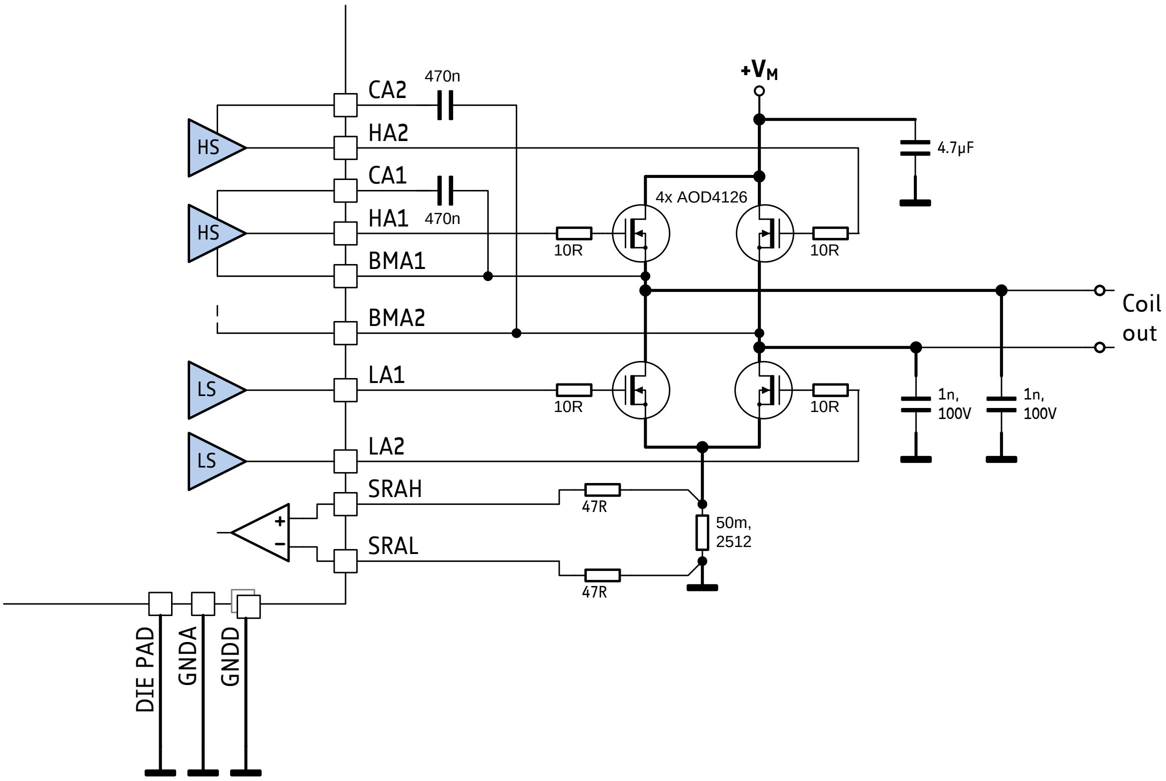


Figure 3.8 Example for bridge with tuned components (see scope shots)

BRIDGE LAYOUT CONSIDERATIONS

- Tune the bridge layout for minimum loop inductivity. A compact layout is best.
- Keep MOSFET gate connections short and straight and avoid loop inductivity between BM and corresponding HS driver pin. Loop inductance is minimized with parallel traces, or adjacent traces on adjacent layers. A wider trace reduces inductivity (don't use minimum trace width).
- Minimize the length of the sense resistor connection to low-side MOSFET source and place the TMC2160 near the sense resistor's GND connection, with its GND connections directly connected to the same GND plane.
- Optimize switching behavior by tuning gate current setting and gate resistors. Add MOSFET bridge output capacitors (470pF to a few nF) to reduce ringing.
- Measure the performance of the bridge by probing BM pins directly at the bridge or at the TMC2160 using a short GND tip on the scope probe rather than a GND cable, if available.

4 SPI Interface

4.1 SPI Datagram Structure

The TMC2160 uses 40-bit SPI™ (Serial Peripheral Interface, SPI is Trademark of Motorola) datagrams for communication with a microcontroller. Microcontrollers which are equipped with hardware SPI are typically able to communicate using integer multiples of 8 bit. The NCS line of the device must be handled in a way, that it stays active (low) for the complete duration of the datagram transmission.

Each datagram sent to the device is composed of an address byte followed by four data bytes. This allows direct 32-bit data word communication with the register set. Each register is accessed via 32 data bits even if it uses less than 32 data bits.

For simplification, each register is specified by a one-byte address:

- For a read access the most significant bit of the address byte is 0.
- For a write access the most significant bit of the address byte is 1.

Most registers are write-only registers, some can be read additionally, and there are also some read only registers.

<table>
<tr><th colspan="40">SPI DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="8">MSB (transmitted first)</td><td colspan="24">40 bit</td><td colspan="8">LSB (transmitted last)</td></tr>
<tr><td colspan="20">39 …</td><td colspan="20">… 0</td></tr>
<tr><td colspan="8">**→ 8 bit address**
← 8 bit SPI status</td><td colspan="32">**← → 32 bit data**</td></tr>
<tr><td colspan="8">39 … 32</td><td colspan="32">31 … 0</td></tr>
<tr><td colspan="8">→ to TMC2160
RW + 7 bit address
← from TMC2160
8 bit SPI status</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td></tr>
<tr><td colspan="8">39 / 38 … 32</td><td colspan="8">31 … 24</td><td colspan="8">23 … 16</td><td colspan="8">15 … 8</td><td colspan="8">7 … 0</td></tr>
<tr><td>W</td><td colspan="7">38…32</td><td colspan="4">31…28</td><td colspan="4">27…24</td><td colspan="4">23…20</td><td colspan="4">19…16</td><td colspan="4">15…12</td><td colspan="4">11…8</td><td colspan="4">7…4</td><td colspan="4">3…0</td></tr>
<tr><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

4.1.1 Selection of Write / Read (WRITE_notREAD)

The read and write selection is controlled by the MSB of the address byte (bit 39 of the SPI datagram). This bit is 0 for read access and 1 for write access. So, the bit named W is a WRITE_notREAD control bit. The active high write bit is the MSB of the address byte. So, 0x80 has to be added to the address for a write access. The SPI interface always delivers data back to the master, independent of the W bit. The data transferred back is the data read from the address, which was transmitted with the *previous* datagram, if the previous access was a read access. If the previous access was a write access, then the data read back mirrors the previously received write data. So, the difference between a read and a write access is that the read access does not transfer data to the addressed register, but it transfers the address only and its 32 data bits are dummies, and, further the following read or write access delivers back the data read from the address transmitted in the preceding read cycle.

A read access request datagram uses dummy write data. Read data is transferred back to the master with the subsequent read or write access. Hence, reading multiple registers can be done in a pipelined fashion.

Whenever data is read from or written to the TMC2160, the MSBs delivered back contain the SPI status, *SPI_STATUS*, a number of eight selected status bits.

Example:

For a read access to the register (*TSTEP*) with the address 0x12, the address byte has to be set to 0x12 in the access preceding the read access. For a write access to the register (*IHOLD_IRUN*), the address byte has to be set to 0x80 + 0x10 = 0x90. For read access, the data bits might have any value (-). So, one can set them to 0.

action	data sent to TMC2160	data received from TMC2160
read *TSTEP*	→ `0x1200000000`	← `0xSS & unused data`
read *TSTEP*	→ `0x1200000000`	← `0xSS & TSTEP`
write *IHOLD_IRUN*:= `0x00011F10`	→ `0x9000ABCDEF`	← `0xSS & TSTEP`
write *IHOLD_IRUN*:= `0x00021807`	→ `0x9000123456`	← `0xSS00011F10`

`*)S:` is a placeholder for the status bits *SPI_STATUS*

4.1.2 SPI Status Bits Transferred with Each Datagram Read Back

New status information becomes latched at the end of each access and is available with the next SPI transfer.

***SPI_STATUS* – status flags transmitted with each SPI access in bits 39 to 32**

Bit	Name	Comment
7	*Unused*	Ignore this bit
6	*Unused*	Ignore this bit
5	*Unused*	Ignore this bit
4	*Unused*	Ignore this bit
3	*standstill*	*DRV_STATUS*[31] – 1: Signals motor stand still
2	*sg2*	*DRV_STATUS*[24] – 1: Signals StallGuard flag active
1	*driver_error*	*GSTAT*[1] – 1: Signals driver 1 driver error (clear *GSTAT* to reset)
0	*reset_flag*	*GSTAT*[0] – 1: Signals, that a reset has occurred (clear *GSTAT* to reset)

4.1.3 Data Alignment

All data are right aligned. Some registers represent unsigned (positive) values, some represent integer values (signed) as two's complement numbers, single bits or groups of bits are represented as single bits respectively as integer groups.

4.2 SPI Signals

The SPI bus on the TMC2160 has four signals:

- SCK – bus clock input
- SDI – serial data input
- SDO – serial data output
- CSN – chip select input (active low)

The slave is enabled for an SPI transaction by a low on the chip select input CSN. Bit transfer is synchronous to the bus clock SCK, with the slave latching the data from SDI on the rising edge of SCK and driving data to SDO following the falling edge. The most significant bit is sent first. A minimum of 40 SCK clock cycles is required for a bus transaction with the TMC2160.

If more than 40 clocks are driven, the additional bits shifted into SDI are shifted out on SDO after a 40-clock delay through an internal shift register. This can be used for daisy chaining multiple chips.

CSN must be low during the whole bus transaction. When CSN goes high, the contents of the internal shift register are latched into the internal control register and recognized as a command from the master to the slave. If more than 40 bits are sent, only the last 40 bits received before the rising edge of CSN are recognized as the command.

4.3 Timing

The SPI interface is synchronized to the internal system clock, which limits the SPI bus clock SCK to half of the system clock frequency. If the system clock is based on the on-chip oscillator, an additional 10% safety margin must be used to ensure reliable data transmission. All SPI inputs as well as the DRV_ENN input are internally filtered to avoid triggering on pulses shorter than 20ns. Figure 4.1 shows the timing parameters of an SPI bus transaction, and the table below specifies their values.

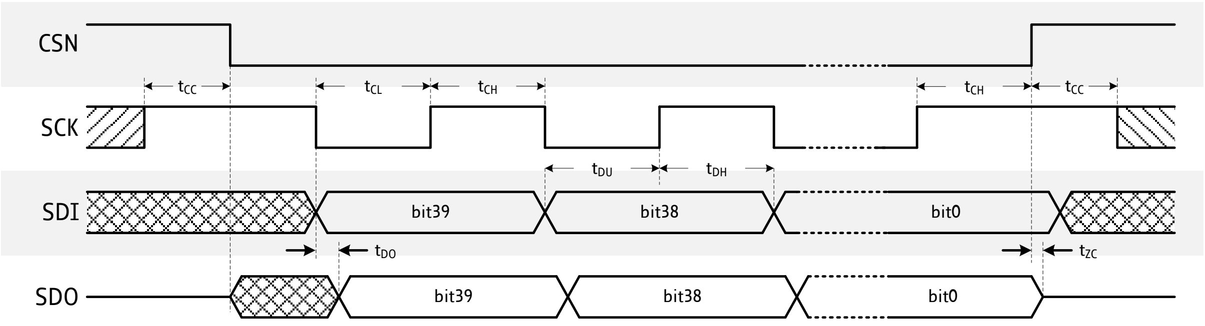


Figure 4.1 SPI timing

Hint
Usually this SPI timing is referred to as SPI MODE 3

SPI interface timing	**AC-Characteristics** clock period: t_{CLK}					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
SCK valid before or after change of CSN	t_{CC}		10			ns
CSN high time	t_{CSH}	*) Min time is for synchronous CLK with SCK high one t_{CH} before CSN high only	$t_{CLK}^{*)}$	$>2t_{CLK}+10$		ns
SCK low time	t_{CL}	*) Min time is for synchronous CLK only	$t_{CLK}^{*)}$	$>t_{CLK}+10$		ns
SCK high time	t_{CH}	*) Min time is for synchronous CLK only	$t_{CLK}^{*)}$	$>t_{CLK}+10$		ns
SCK frequency using internal clock	f_{SCK}	assumes minimum OSC frequency			4	MHz
SCK frequency using external 16MHz clock	f_{SCK}	assumes synchronous CLK			8	MHz
SDI setup time before rising edge of SCK	t_{DU}		10			ns
SDI hold time after rising edge of SCK	t_{DH}		10			ns
Data out valid time after falling SCK clock edge	t_{DO}	no capacitive load on SDO			$t_{FILT}+5$	ns
SDI, SCK and CSN filter delay time	t_{FILT}	rising and falling edge	12	20	30	ns

5 Register Mapping

This chapter gives an overview of the complete register set. Some of the registers bundling a number of single bits are detailed in extra tables. The functional practical application of the settings is detailed in dedicated chapters.

Note
- All registers become reset to 0 upon power up, unless otherwise noted.
- Add 0x80 to the address **Addr** for write accesses!

NOTATION OF HEXADECIMAL AND BINARY NUMBERS	
0x	precedes a hexadecimal number, e.g. 0x04
%	precedes a multi-bit binary number, e.g. %100

NOTATION OF R/W FIELD	
R	Read only
W	Write only
R/W	Read- and writable register
R+WC	Clear by writing "1" bit

OVERVIEW REGISTER MAPPING

REGISTER	**DESCRIPTION**
General Configuration Registers	These registers contain - global configuration - global status flags - interface configuration - and I/O signal configuration
Velocity Dependent Driver Feature Control Register Set	This register set offers registers for - driver current control - setting thresholds for CoolStep operation - setting thresholds for different chopper modes - setting thresholds for DcStep operation
Motor Driver Register Set	This register set offers registers for - setting / reading out microstep table and counter - chopper and driver configuration - CoolStep and StallGuard2 configuration - DcStep configuration - reading out StallGuard2 values and driver error flags

5.1 General Configuration Registers

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)

R/W	Addr	n	*Register*	Bit	*GCONF* – Global configuration flags
RW	0x00	18	*GCONF*	0	*recalibrate* 1: Zero crossing recalibration during driver disable (via DRV_ENN or via *TOFF* setting)
				1	*faststandstill* Timeout for step execution until standstill detection: 1: Short time: 2^18 clocks 0: Normal time: 2^20 clocks
				2	*en_pwm_mode* 1: StealthChop voltage PWM mode enabled (depending on velocity thresholds). Switch from off to on state while in stand-still and at IHOLD= nominal IRUN current, only.
				3	*multistep_filt* 1: Enable step input filtering for StealthChop optimization with external step source (default=1)
				4	*shaft* 1: Inverse motor direction
				5	*diag0_error* 1: Enable DIAG0 active on driver errors: Over temperature (*ot*), short to GND (*s2g*) DIAG0 always shows the reset-status, i.e., is active low during reset condition.
				6	*diag0_otpw* 1: Enable DIAG0 active on driver over temperature prewarning (*otpw*)
				7	*diag0_stall* 1: Enable DIAG0 active on motor stall (set *TCOOLTHRS* before using this feature)
				8	*diag1_stall* 1: Enable DIAG1 active on motor stall (set *TCOOLTHRS* before using this feature)
				9	*diag1_index* 1: Enable DIAG1 active on index position (microstep look up table position 0)
				10	*diag1_onstate* 1: Enable DIAG1 active when chopper is on (for the coil which is in the second half of the fullstep)
				11	*diag1_steps_skipped* 1: Enable output toggle when steps are skipped in DcStep mode (increment of *LOST_STEPS*). Do not enable in conjunction with other DIAG1 options.
				12	*diag0_int_pushpull* 0: DIAG0 is open collector output (active low) 1: Enable DIAG0 push pull output (active high)
				13	*diag1_pushpull* 0: DIAG1 is open collector output (active low) 1: Enable DIAG1 push pull output (active high)
				14	*small_hysteresis* 0: Hysteresis for step frequency comparison is 1/16 1: Hysteresis for step frequency comparison is 1/32

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)

R/W	Addr	n	*Register*	**Description / *bit names***	
				15	*stop_enable* 0: Normal operation 1: Emergency stop: ENCA_DCIN stops the sequencer when tied high (no steps become executed by the sequencer, motor goes to standstill state).
				16	*direct_mode* 0: Normal operation 1: Motor coil currents and polarity directly programmed via serial interface: Register *XDIRECT* (0x2D) specifies signed coil A current (bits 8..0) and coil B current (bits 24..16). In this mode, the current is scaled by *IHOLD* setting. Velocity based current regulation of StealthChop is not available in this mode. The automatic StealthChop current regulation will work only for low stepper motor velocities.
				17	*test_mode* 0: Normal operation 1: Enable analog test output on pin DCO. *IHOLD*[1..0] selects the function of DCO: 0…2: T120, DAC, VDDH *Hint:* Not for user, set to 0 for normal operation!
R+ WC	0x01	3	*GSTAT*	**Bit**	***GSTAT* – Global status flags** (Re-Write with '1' bit to clear respective flags)
				0	*reset* 1: Indicates that the IC has been reset. All registers have been cleared to reset values.
				1	*drv_err* 1: Indicates, that the driver has been shut down due to overtemperature or short circuit detection. Read DRV_STATUS for details. The flag can only be cleared when the temperature is below the limit again.
				2	*uv_cp* 1: Indicates an undervoltage on the charge pump. The driver is disabled during undervoltage. This flag is latched for information.
R	0x04	8 + 8	*IOIN*	**Bit**	***INPUT***
					Reads the state of all input pins available
				0	STEP
				1	DIR
				2	DCEN_CFG4
				3	DCIN_CFG5
				4	DRV_ENN
				5	DCO_CFG6
				6	1
				7	unused
				31.. 24	*VERSION*: 0x30=first version of the IC Identical numbers mean full digital compatibility.

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)

R/W	Addr	n	*Register*	Description / *bit names*	
W	0x06		*OTP_PROG*	**Bit**	***OTP_PROGRAM* – OTP programming** Write access programs OTP memory (one bit at a time), Read access refreshes read data from OTP after a write
				2..0	*OTPBIT* Selection of OTP bit to be programmed to the selected byte location (n=0..7: programs bit n to a logic 1)
				5..4	*OTPBYTE* Set to 00
				15..8	*OTPMAGIC* Set to *0xbd* to enable programming. A programming time of minimum 10ms per bit is recommended (check by reading *OTP_READ*).
R	0x07		*OTP_READ*	**Bit**	***OTP_READ*** (Access to OTP memory result and update) *See separate table!*
				7..0	*OTP0* byte 0 read data
RW	0x08	5	*FACTORY_CONF*	4..0	*FCLKTRIM* (*Reset default: OTP*) 0…31: Lowest to highest clock frequency. Check at charge pump output. The frequency span is not guaranteed, but it is tested, that tuning to 12MHz internal clock is possible. The devices come preset to 12MHz clock frequency by OTP programming. *(Reset Default: OTP)*
W	0x09	19	*SHORT_CONF*	**Bit**	***SHORT_CONF***
				3..0	*S2VS_LEVEL:* Short to VS detector level for lowside FETs. Checks for voltage drop in LS MOSFET and sense resistor. 4 (highest sensitivity) … 15 (lowest sensitivity) *Hint:* Settings from 1 to 3 will trigger during normal operation due to voltage drop on sense resistor. *(Reset Default: OTP 6 or 12)*
				11..8	*S2G_LEVEL:* Short to GND detector level for highside FETs. Checks for voltage drop on high side MOSFET 2 (highest sensitivity) … 15 (lowest sensitivity) *Attention: Settings below 6 not recommended at >52V operation – false detection might result* *(Reset Default: OTP 6 or 12)*
				17..16	*SHORTFILTER*: Spike filtering bandwidth for short detection 0 (lowest, 100ns), 1 (1µs), 2 (2µs) 3 (3µs) *Hint:* A good PCB layout will allow using setting 0. Increase value, if erroneous short detection occurs. *(Reset Default = %01)*
				18	*shortdelay*: Short detection delay 0=750ns: normal, 1=1500ns: high The short detection delay shall cover the bridge switching time. 0 will work for most applications. *(Reset Default = 0)*

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description / bit names**	
				Bit	**DRV_CONF**
W	0x0A	22	*DRV_CONF*	4..0	*BBMTIME:* Break-Before make delay 0=shortest (100ns) … 16 (200ns) … 24=longest (375ns) >24 not recommended, use *BBMCLKS* instead *Hint:* Choose the lowest setting safely covering the switching event to avoid bridge cross-conduction. Add roughly 30% of reserve. *(Reset Default = 0)*
				11..8	*BBMCLKS*: 0..15: Digital BBM time in clock cycles (typ. 83ns). The longer setting rules (*BBMTIME* vs. *BBMCLKS*). *(Reset Default: OTP 4 or 2)*
				17..16	*OTSELECT*: Selection of over temperature level for bridge disable, switch on after cool down to 120°C / OTPW level. 00: 150°C 01: 143°C 10: 136°C (not recommended when VSA > 24V) 11: 120°C (not recommended, no hysteresis) *Hint:* Adapt overtemperature threshold as required to protect the MOSFETs or other components on the PCB. *(Reset Default = %00)*
				19..18	*DRVSTRENGTH*: Selection of gate driver current. Adapts the gate driver current to the gate charge of the external MOSFETs. 00: weak 01: weak+TC (medium above OTPW level) 10: medium 11: strong *Hint:* Choose the lowest setting giving slopes <100ns. *(Reset Default = %00)*
				21..20	*FILT_ISENSE*: Filter time constant of sense amplifier to suppress ringing and coupling from second coil operation 00: low – 100ns 01: – 200ns 10: – 300ns 11: high– 400ns *Hint:* Increase setting if motor chopper noise occurs due to cross-coupling of both coils. *(Reset Default = %00)*

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description / *bit names***	
W	0x0B	8	*GLOBAL SCALER*	7..0	Global scaling of Motor current. This value is multiplied to the current scaling in order to adapt a drive to a certain motor type. This value should be chosen before tuning other settings because it also influences chopper hysteresis. 0: Full Scale (or write 256) 1 … 31: Not allowed for operation 32 … 255: 32/256 … 255/256 of maximum current. *Hint:* Values >128 recommended for best results *(Reset Default = 0)*
R	0x0C	16	*OFFSET_ READ*	15..8	Offset calibration result phase A (signed)
				7..0	Offset calibration result phase B (signed)

5.1.1 *OTP_READ* – OTP configuration memory

The OTP memory holds power up defaults for certain registers. All OTP memory bits are cleared to 0 by default. Programming only can set bits, clearing bits is not possible. Factory tuning of the clock frequency affects *otp0.0* to *otp0.4*. The state of these bits therefore may differ between individual ICs.

0X07: *OTP_READ* – OTP MEMORY MAP

Bit	Name	Function	Comment
7	*otp0.7*	*otp_TBL*	Reset default for *TBL*: 0: *TBL*=%10 (~3µs) 1: *TBL*=%01 (~2µs)
6	*otp0.6*	*otp_BBM*	Reset default for *DRVCONF.BBMCLKS* 0: *BBMCLKS*=4 1: *BBMCLKS*=2
5	*otp0.5*	*otp_S2_LEVEL*	Reset default for *short-detection Levels*: 0: *S2G_LEVEL = S2VS_LEVEL = 6* 1: *S2G_LEVEL = S2VS_LEVEL = 12*
4	*otp0.4*	*OTP_FCLKTRIM*	Reset default for *FCLKTRIM* 0: lowest frequency setting 31: highest frequency setting *Attention: This value is pre-programmed by factory clock trimming to the default clock frequency of 12MHz and differs between individual ICs! It should not be altered.*
3	*otp0.3*		
2	*otp0.2*		
1	*otp0.1*		
0	*otp0.0*		

5.2 Velocity Dependent Driver Feature Control Register Set

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0X10…0X1F)

R/W	Addr	n	*Register*	Bit	Description / *bit names*
W	0x10	5 + 5 + 4	*IHOLD_IRUN*	**Bit**	***IHOLD_IRUN* – Driver current control**
				4..0	*IHOLD* Standstill current (0=1/32…31=32/32) In combination with StealthChop mode, setting *IHOLD*=0 allows to choose freewheeling or coil short circuit for motor stand still.
				12..8	*IRUN* Motor run current (0=1/32…31=32/32) *Hint:* Choose sense resistors in a way, that normal IRUN is 16 to 31 for best microstep performance.
				19..16	*IHOLDDELAY* Controls the number of clock cycles for motor power down after a motion as soon as standstill is detected (*stst*=1) and *TPOWERDOWN* has expired. The smooth transition avoids a motor jerk upon power down. 0: instant power down 1..15: Delay per current reduction step in multiple of 2^18 clocks
W	0x11	8	*TPOWER DOWN*		*TPOWERDOWN* sets the delay time after stand still (*stst*) of the motor to motor current power down. Time range is about 0 to 4 seconds. *Attention: A minimum setting of 2 is required to allow automatic tuning of StealthChop PWM_OFS_AUTO.* *Reset Default = 10* *0…((2^8)-1)* * 2^18 t_{CLK}
R	0x12	20	*TSTEP*		Actual measured time between two 1/256 microsteps derived from the step input frequency in units of 1/fCLK. Measured value is (2^20)-1 in case of overflow or stand still. All TSTEP related thresholds use a hysteresis of 1/16 of the compare value to compensate for jitter in the clock or the step frequency. The flag *small_hysteresis* modifies the hysteresis to a smaller value of 1/32. (*Txxx**15/16)-1 or (*Txxx**31/32)-1 is used as a second compare value for each comparison value. This means, that the lower switching velocity equals the calculated setting, but the upper switching velocity is higher as defined by the hysteresis setting. In DcStep mode *TSTEP* will not show the mean velocity of the motor, but the velocities for each microstep, which may not be stable and thus does not represent the real motor velocity in case it runs slower than the target velocity.
W	0x13	20	*TPWMTHRS*		This is the upper velocity for StealthChop voltage PWM mode. *TSTEP* ≥ *TPWMTHRS* - StealthChop PWM mode is enabled, if configured - DcStep is disabled

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0X10…0X1F)

R/W	Addr	n	*Register*	**Description / *bit names***	
W	0x14	20	*TCOOLTHRS*	This is the lower threshold velocity for switching on smart energy CoolStep and StallGuard feature. (unsigned) Set this parameter to disable CoolStep at low speeds, where it cannot work reliably. The stall output signal becomes enabled when exceeding this velocity. In non-DcStep mode, it becomes disabled again once the velocity falls below this threshold. *TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: - CoolStep is enabled, if configured - StealthChop voltage PWM mode is disabled *TCOOLTHRS* ≥ *TSTEP* - Stall output signal (DIAG0/1) is enabled, if configured	
W	0x15	20	*THIGH*	This velocity setting allows velocity dependent switching into a different chopper mode and fullstepping to maximize torque. (unsigned) The stall detection feature becomes switched off for 2-3 electrical periods whenever passing *THIGH* threshold to compensate for the effect of switching modes. *TSTEP* ≤ *THIGH*: - CoolStep is disabled (motor runs with normal current scale) - StealthChop voltage PWM mode is disabled - If *vhighchm* is set, the chopper switches to *chm*=1 with *TFD*=0 (constant off time with slow decay, only). - If *vhighfs* is set, the motor operates in fullstep mode and the stall detection becomes switched over to DcStep stall detection.	
RW	0x2D	9+9	*XDIRECT*	This register is used in direct coil current mode, only (*direct_mode* = 1). It bypasses the internal sequencer. Specifies signed coil A current (bits 8..0) and coil B current (bits 24..16). In this mode, the current is scaled by *IHOLD* setting. Velocity based current regulation of StealthChop is not available in this mode. The automatic StealthChop current regulation will work only for low stepper motor velocities.	2x -255…+255

Microstep velocity time reference t for velocities: *TSTEP* = f_{CLK} / $f_{STEP256}$. *TSTEP* is related to 1/256 microstep resolution independent of actual resolution set by *MRES*.

5.2.1 DcStep Minimum Velocity Register

DCSTEP MINIMUM VELOCITY REGISTER (0X33)				
R/W	**Addr**	**n**	***Register***	**Description / *bit names***
W	0x33	23	*VDCMIN*	Automatic commutation DcStep minimum velocity. Enable DcStep by DCEN pin. In this mode, the actual position is determined by the sensorless motor commutation and becomes fed back to the external motion controller. In case the motor becomes heavily loaded, *VDCMIN* is used as the minimum step velocity. *Hint:* Also set *DCCTRL* parameters to operate DcStep. (Only bits 22… 8 are used for value and for comparison)

Time reference t for *VDCMIN*: t = 2^24 / f_{CLK}

5.3 Motor Driver Registers

MICROSTEPPING CONTROL REGISTER SET (0X60…0X6B)

R/W	Addr	n	*Register*	Description / *bit names*	Range [Unit]
W	0x60	32	*MSLUT[0]* microstep table entries 0…31	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224*	32x 0 or 1 *reset default= sine wave table*
W	0x61 … 0x67	7 x 32	*MSLUT[1…7]* microstep table entries 32…255		7x 32x 0 or 1 *reset default= sine wave table*
W	0x68	32	***MSLUTSEL***	This register defines four segments within each quarter *MSLUT* wave. Four 2-bit entries determine the meaning of a 0 and a 1 bit in the corresponding segment of *MSLUT*. *See separate table!*	*0<X1<X2<X3* *reset default= sine wave table*
W	0x69	8 + 8	*MSLUTSTART*	bit 7… 0: *START_SIN* bit 23… 16: *START_SIN90* *START_SIN* gives the absolute current at microstep table entry 0. *START_SIN90* gives the absolute current for microstep table entry at positions 256. Start values are transferred to the microstep registers *CUR_A* and *CUR_B*, whenever the reference position *MSCNT*=0 is passed.	*START_SIN* *reset default =0* *START_SIN90* *reset default =247*
R	0x6A	10	*MSCNT*	Microstep counter. Indicates actual position in the microstep table for *CUR_B*. *CUR_A* uses an offset of 256 (2 phase motor). *Hint:* Move to a position where *MSCNT* is zero before re-initializing *MSLUTSTART* or *MSLUT* and *MSLUTSEL*.	0…1023
R	0x6B	9 + 9	*MSCURACT*	bit 8… 0: *CUR_B* (signed): Actual microstep current for motor phase B (sine wave) as read from MSLUT (not scaled by current) bit 24… 16: *CUR_A* (signed): Actual microstep current for motor phase A (co-sine wave) as read from *MSLUT* (not scaled by current)	+/-0...255

DRIVER REGISTER SET (0X6C…0X7F)

R/W	Addr	n	Register	Description / bit names	Range [Unit]
RW	0x6C	32	**CHOPCONF**	chopper and driver configuration *See separate table!*	*reset default=* *0x10410150*
W	0x6D	25	**COOLCONF**	CoolStep smart current control register and StallGuard2 configuration *See separate table!*	
W	0x6E	24	*DCCTRL*	DcStep (*DC*) automatic commutation configuration register (enable via pin DCEN or via *VDCMIN*): bit 9… 0: *DC_TIME*: Upper PWM on time limit for commutation (*DC_TIME* * $1/f_{CLK}$). Set slightly above effective blank time *TBL*. bit 23… 16: *DC_SG*: Max. PWM on time for step loss detection using DcStep StallGuard2 in DcStep mode. (*DC_SG* * $16/f_{CLK}$) Set slightly higher than *DC_TIME*/16 0=disable *Hint:* Using a higher microstep resolution or interpolated operation, DcStep delivers a better StallGuard signal. DC_SG is also available above VHIGH if vhighfs is activated. For best result also set vhighchm.	
R	0x6F	32	**DRV_ STATUS**	StallGuard2 value and driver error flags *See separate table!*	
W	0x70	32	**PWMCONF**	Voltage PWM mode chopper configuration *See separate table!*	*reset default=* *0xC40C001E*
R	0x71	9+8	*PWM_SCALE*	Results of StealthChop amplitude regulator. These values can be used to monitor automatic PWM amplitude scaling (255=max. voltage).	
				bit 7… 0 *PWM_SCALE_SUM*: Actual PWM duty cycle. This value is used for scaling the values *CUR_A* and *CUR_B* read from the sine wave table.	0…255
				bit 24… 16 *PWM_SCALE_AUTO*: 9 Bit signed offset added to the calculated PWM duty cycle. This is the result of the automatic amplitude regulation based on current measurement.	signed -255…+255
R	0x72	8+8	*PWM_AUTO*	These automatically generated values can be read out in order to determine a default / power up setting for *PWM_GRAD* and *PWM_OFS*.	
				bit 7… 0 *PWM_OFS_AUTO*: Automatically determined offset value	0…255

DRIVER REGISTER SET (0X6C…0X7F)						
R/W	**Addr**	**n**	***Register***	**Description / *bit names***		**Range [Unit]**
				bit 23… 16	*PWM_GRAD_AUTO*: Automatically determined gradient value	0…255
R	0x73	20	*LOST_STEPS*	Number of input steps skipped due to higher load in DcStep operation, if step input does not stop when DC_OUT is low. This counter wraps around after 2^20 steps. Counts up or down depending on direction.		

MICROSTEP TABLE CALCULATION FOR A SINE WAVE EQUIVALENT TO THE POWER ON DEFAULT

$$round\left(248 * sin\left(2 * PI * \frac{i}{1024} + \frac{PI}{1024}\right)\right) - 1$$

- *i*: [0… 255] is the table index
- The amplitude of the wave is 248. The resulting maximum positive value is 247 and the maximum negative value is -248.
- The round function rounds values from 0.5 to 1.4999 to 1

5.3.1 *MSLUTSEL* – Look up Table Segmentation Definition

0x68: *MSLUTSEL* – Look up table segmentation definition			
Bit	**Name**	**Function**	**Comment**
31	*X3*	LUT segment 3 start	The sine wave look-up table can be divided into up to four segments using an individual step width control entry *Wx*. The segment borders are selected by *X1*, *X2* and *X3*. Segment 0 goes from 0 to *X1*-1. Segment 1 goes from *X1* to *X2*-1. Segment 2 goes from *X2* to *X3*-1. Segment 3 goes from *X3* to 255. For defined response the values shall satisfy: 0<*X1*<*X2*<*X3*
30			
29			
28			
27			
26			
25			
24			
23	*X2*	LUT segment 2 start	
22			
21			
20			
19			
18			
17			
16			
15	*X1*	LUT segment 1 start	
14			
13			
12			
11			
10			
9			
8			
7	*W3*	LUT width select from *ofs(X3)* to *ofs255*	Width control bit coding *W0…W3*: %00: MSLUT entry 0, 1 select: -1, +0 %01: MSLUT entry 0, 1 select: +0, +1 %10: MSLUT entry 0, 1 select: +1, +2 %11: MSLUT entry 0, 1 select: +2, +3
6			
5	*W2*	LUT width select from *ofs(X2)* to *ofs(X3-1)*	
4			
3	*W1*	LUT width select from *ofs(X1)* to *ofs(X2-1)*	
2			
1	*W0*	LUT width select from *ofs00* to *ofs(X1-1)*	
0			

5.3.2 *CHOPCONF* – Chopper Configuration

0X6C: *CHOPCONF* – CHOPPER CONFIGURATION

Bit	Name	Function	Comment
31	*diss2vs*	short to supply protection disable	0: Short to VS protection is on 1: Short to VS protection is disabled
30	*diss2g*	short to GND protection disable	0: Short to GND protection is on 1: Short to GND protection is disabled
29	*dedge*	enable double edge step pulses	1: Enable step impulse at each step edge to reduce step frequency requirement.
28	*intpol*	interpolation to 256 microsteps	1: The actual microstep resolution (*MRES*) becomes extrapolated to 256 microsteps for smoothest motor operation (useful for STEP/DIR operation, only)
27	*mres3*	*MRES* micro step resolution	%0000: Native 256 microstep setting. Normally use this setting with the internal motion controller.
26	*mres2*		
25	*mres1*		
24	*mres0*		%0001 … %1000: 128, 64, 32, 16, 8, 4, 2, FULLSTEP Reduced microstep resolution esp. for STEP/DIR operation. The resolution gives the number of microstep entries per sine quarter wave. The driver automatically uses microstep positions which result in a symmetrical wave, when choosing a lower microstep resolution. step width=2^*MRES* [microsteps]
23	*tpfd3*	*TPFD* passive fast decay time	*TPFD* allows dampening of motor mid-range resonances. Passive fast decay time setting controls duration of the fast decay phase inserted after bridge polarity change N_{CLK}= 128**TPFD* %0000: Disable %0001 … %1111: 1 … 15
22	*tpfd2*		
21	*tpdf1*		
20	*tpfd0*		
19	*vhighchm*	high velocity chopper mode	This bit enables switching to *chm*=1 and *fd*=0, when *VHIGH* is exceeded. This way, a higher velocity can be achieved. Can be combined with *vhighfs*=1. If set, the *TOFF* setting automatically becomes doubled during high velocity operation in order to avoid doubling of the chopper frequency.
18	*vhighfs*	high velocity fullstep selection	This bit enables switching to fullstep, when *VHIGH* is exceeded. Switching takes place only at 45° position. The fullstep target current uses the current value from the microstep table at the 45° position.
17	-	reserved	reserved, set to 0
16	*tbl1*	*TBL* blank time select	%00 … %11: Set comparator blank time to 16, 24, 36 or 54 clocks *Hint*: %01 or %10 is recommended for most applications *(Reset Default: OTP %01 or %10)*
15	*tbl0*		
14	*chm*	chopper mode	0 Standard mode (SpreadCycle) 1 Constant off time with fast decay time. Fast decay time is also terminated when the negative nominal current is reached. Fast decay is after on time.
13	-	reserved	Reserved, set to 0
12	*disfdcc*	fast decay mode	*chm*=1: *disfdcc*=1 disables current comparator usage for termination of the fast decay cycle

11	*fd3*	*TFD [3]*	*chm*=1: MSB of fast decay time setting *TFD*	
10	*hend3*	*HEND* hysteresis low value *OFFSET* sine wave offset	*chm*=0	%0000 … %1111: Hysteresis is -3, -2, -1, 0, 1, …, 12 (1/512 of this setting adds to current setting) This is the hysteresis value which becomes used for the hysteresis chopper.
9	*hend2*			
8	*hend1*			
7	*hend0*		*chm*=1	%0000 … %1111: Offset is -3, -2, -1, 0, 1, …, 12 This is the sine wave offset and 1/512 of the value becomes added to the absolute value of each sine wave entry.
6	*hstrt2*	*HSTRT* hysteresis start value added to *HEND*	*chm*=0	%000 … %111: Add 1, 2, …, 8 to hysteresis low value *HEND* (1/512 of this setting adds to current setting) *Attention: Effective HEND+HSTRT ≤ 16.* *Hint:* Hysteresis decrement is done each 16 clocks
5	*hstrt1*			
4	*hstrt0*			
		TFD [2..0] fast decay time setting	*chm*=1	Fast decay time setting (MSB: *fd3*): %0000 … %1111: Fast decay time setting *TFD* with $N_{CLK}= 32*TFD$ (%0000: slow decay only)
3	*toff3*	*TOFF* off time and driver enable	Off time setting controls duration of slow decay phase $N_{CLK}= 24 + 32*TOFF$ %0000: Driver disable, all bridges off %0001: 1 – use only with *TBL* ≥ 2 %0010 … %1111: 2 … 15	
2	*toff2*			
1	*toff1*			
0	*toff0*			

5.3.3 *COOLCONF* – Smart Energy Control CoolStep and StallGuard2

0X6D: *COOLCONF* – SMART ENERGY CONTROL COOLSTEP AND STALLGUARD2

Bit	Name	Function	Comment
…	-	reserved	set to 0
24	*sfilt*	StallGuard2 filter enable	0 Standard mode, high time resolution for StallGuard2 1 Filtered mode, StallGuard2 signal updated for each four fullsteps (resp. six fullsteps for 3 phase motor) only to compensate for motor pole tolerances
23	-	reserved	set to 0
22	*sgt6*	StallGuard2 threshold value	This signed value controls StallGuard2 level for stall output and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value working with most motors. -64 to +63: A higher value makes StallGuard2 less sensitive and requires more torque to indicate a stall.
21	*sgt5*		
20	*sgt4*		
19	*sgt3*		
18	*sgt2*		
17	*sgt1*		
16	*sgt0*		
15	*seimin*	minimum current for smart current control	0: 1/2 of current setting (*IRUN*) 1: 1/4 of current setting (*IRUN*)
14	*sedn1*	current down step speed	%00: For each 32 StallGuard2 values decrease by one %01: For each 8 StallGuard2 values decrease by one %10: For each 2 StallGuard2 values decrease by one %11: For each StallGuard2 value decrease by one
13	*sedn0*		
12	-	reserved	set to 0
11	*semax3*	StallGuard2 hysteresis value for smart current control	If the StallGuard2 result is equal to or above (*SEMIN*+*SEMAX*+1)*32, the motor current becomes decreased to save energy. %0000 … %1111: 0 … 15
10	*semax2*		
9	*semax1*		
8	*semax0*		
7	-	reserved	set to 0
6	*seup1*	current up step width	Current increment steps per measured StallGuard2 value %00 … %11: 1, 2, 4, 8
5	*seup0*		
4	-	reserved	set to 0
3	*semin3*	minimum StallGuard2 value for smart current control and smart current enable	If the StallGuard2 result falls below *SEMIN**32, the motor current becomes increased to reduce motor load angle. %0000: smart current control CoolStep off %0001 … %1111: 1 … 15
2	*semin2*		
1	*semin1*		
0	*semin0*		

5.3.4 *PWMCONF* – Voltage PWM Mode StealthChop

0x70: *PWMCONF* – VOLTAGE MODE PWM STEALTHCHOP

Bit	Name	Function	Comment
31	*PWM_LIM*	PWM automatic scale amplitude limit when switching on	Limit for *PWM_SCALE_AUTO* when switching back from SpreadCycle to StealthChop. This value defines the upper limit for bits 7 to 4 of the automatic current control when switching back. It can be set to reduce the current jerk during mode change back to StealthChop. It does not limit *PWM_GRAD* or *PWM_GRAD_AUTO* offset. (Default = 12)
30			
29			
28			
27	*PWM_REG*	Regulation loop gradient	User defined maximum PWM amplitude change per half wave when using *pwm_autoscale*=1. (1…15): 1: 0.5 increments (slowest regulation) 2: 1 increment 3: 1.5 increments 4: 2 increments *(Reset default)*) ... 8: 4 increments ... 15: 7.5 increments (fastest regulation)
26			
25			
24			
23	-	reserved	set to 0
22	-	reserved	set to 0
21	*freewheel1*	Allows different standstill modes	Stand still option when motor current setting is zero (*I_HOLD*=0). %00: Normal operation %01: Freewheeling %10: Coil shorted using LS drivers %11: Coil shorted using HS drivers
20	*freewheel0*		
19	*pwm_ autograd*	PWM automatic gradient adaptation	0 Fixed value for *PWM_GRAD* (*PWM_GRAD_AUTO* = *PWM_GRAD*)
			1 Automatic tuning (only with *pwm_autoscale*=1) *(Reset default)* *PWM_GRAD_AUTO* is initialized with *PWM_GRAD* while *pwm_autograd*=0 and becomes optimized automatically during motion. Preconditions 1. *PWM_OFS_AUTO* has been automatically initialized. This requires standstill at *IRUN* for >130ms to a) detect standstill b) wait > 128 chopper cycles at *IRUN* and c) regulate *PWM_OFS_AUTO* so that -1 < *PWM_SCALE_AUTO* < 1 2. Motor running and *PWM_SCALE_SUM* < 255 and 1.5 * *PWM_OFS_AUTO* * (*IRUN*+1)/32 < *PWM_SCALE_SUM* < 4* *PWM_OFS_AUTO* * (*IRUN*+1)/32. Time required for tuning *PWM_GRAD_AUTO* About 8 fullsteps per change of +/-1. Also enables use of reduced chopper frequency for tuning *PWM_OFS_AUTO*.

0x70: *PWMCONF* – VOLTAGE MODE PWM STEALTHCHOP

Bit	Name	Function	Comment
18	*pwm_ autoscale*	PWM automatic amplitude scaling	0: User defined feed forward PWM amplitude. The current settings *IRUN* and *IHOLD* are not enforced by regulation, but scale the PWM amplitude, only! The resulting PWM amplitude (limited to 0…255) is: *PWM_OFS * ((CS_ACTUAL+1) / 32)* + *PWM_GRAD ** 256 / *TSTEP*
			1: Enable automatic current control *(Reset default)*
17	*pwm_freq1*	PWM frequency selection	%00: $f_{PWM}=2/1024\ f_{CLK}$ *(Reset default)* %01: $f_{PWM}=2/683\ f_{CLK}$ %10: $f_{PWM}=2/512\ f_{CLK}$ %11: $f_{PWM}=2/410\ f_{CLK}$
16	*pwm_freq0*		
15	*PWM_ GRAD*	User defined amplitude gradient	Velocity dependent gradient for PWM amplitude: *PWM_GRAD ** 256 / *TSTEP* This value is added to *PWM_OFS* to compensate for the velocity-dependent motor back-EMF. Use *PWM_GRAD* as initial value for automatic scaling to speed up the automatic tuning process. To do this, set *PWM_GRAD* to the determined, application specific value, with *pwm_autoscale*=0. Only afterwards, set *pwm_autoscale*=1. Enable StealthChop when finished. *Hint:* After initial tuning, the required initial value can be read out from *PWM_GRAD_AUTO.*
14			
13			
12			
11			
10			
9			
8			
7	*PWM_ OFS*	User defined amplitude (offset)	User defined PWM amplitude offset (0-255) related to full motor current (*CS_ACTUAL*=31) in stand still. (*Reset default*=30) Use *PWM_OFS* as initial value for automatic scaling to speed up the automatic tuning process. To do this, set *PWM_OFS* to the determined, application specific value, with *pwm_autoscale*=0. Only afterwards, set *pwm_autoscale*=1. Enable StealthChop when finished. *PWM_OFS* = 0 will disable scaling down motor current below a motor specific lower measurement threshold. This setting should only be used under certain conditions, i.e., when the power supply voltage can vary up and down by a factor of two or more. It prevents the motor going out of regulation, but it also prevents power down below the regulation limit. *PWM_OFS* > 0 allows automatic scaling to low PWM duty cycles even below the lower regulation threshold. This allows low (standstill) current settings based on the actual (hold) current scale (register *IHOLD_IRUN*).
6			
5			
4			
3			
2			
1			
0			

5.3.5 *DRV_STATUS* – StallGuard2 Value and Driver Error Flags

0X6F: DRV_STATUS – STALLGUARD2 VALUE AND DRIVER ERROR FLAGS

Bit	Name	Function	Comment
31	*stst*	standstill indicator	This flag indicates motor stand still in each operation mode. This occurs 2^20 clocks after the last step pulse.
30	*olb*	open load indicator phase B	1: Open load detected on phase A or B. *Hint:* This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion, only.
29	*ola*	open load indicator phase A	
28	*s2gb*	short to ground indicator phase B	1: Short to GND detected on phase A or B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (*TOFF*=0) or by the DRV_ENN input.
27	*s2ga*	short to ground indicator phase A	
26	*otpw*	overtemperature pre-warning flag	1: Overtemperature pre-warning threshold is exceeded. The overtemperature pre-warning flag is common for both bridges.
25	*ot*	overtemperature flag	1: Overtemperature limit has been reached. Drivers become disabled until *otpw* is also cleared due to cooling down of the IC. The overtemperature flag is common for both bridges.
24	*StallGuard*	StallGuard2 status	1: Motor stall detected (*SG_RESULT*=0) or DcStep stall in DcStep mode.
23	-	reserved	Ignore these bits
22			
21			
20	*CS ACTUAL*	actual motor current / smart energy current	Actual current control scaling, for monitoring smart energy current scaling controlled via settings in register *COOLCONF*, or for monitoring the function of the automatic current scaling.
19			
18			
17			
16			
15	*fsactive*	full step active indicator	1: Indicates that the driver has switched to fullstep as defined by chopper mode settings and velocity thresholds.
14	*stealth*	StealthChop indicator	1: Driver operates in StealthChop mode
13	*s2vsb*	short to supply indicator phase B	1: Short to supply detected on phase A or B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (*TOFF*=0) or by the DRV_ENN input. Sense resistor voltage drop is included in the measurement!
12	*s2vsa*	short to supply indicator phase A	
11	-	reserved	Ignore this bit
10	-	reserved	Ignore this bit
9	*SG_RESULT*	StallGuard2 result respectively PWM on time for coil A in standstill for motor temperature detection	Mechanical load measurement: The StallGuard2 result gives a means to measure mechanical motor load. A higher value means lower mechanical load. A value of 0 signals highest load. With optimum *SGT* setting, this is an indicator for a motor stall. The stall detection compares *SG_RESULT* to 0 to detect a stall. *SG_RESULT* is used as a base for CoolStep operation, by comparing it to a programmable upper and a lower limit. It is not applicable in StealthChop mode. StallGuard2 works best with microstep operation or DcStep. Temperature measurement: In standstill, no StallGuard2 result can be obtained. *SG_RESULT* shows the chopper on-time for motor coil A instead. Move the motor to a determined microstep position at a certain current setting to get a rough estimation of motor temperature by a reading the chopper on-time. As the motor heats up, its coil resistance rises and the chopper on-time increases.
8			
7			
6			
5			
4			
3			
2			
1			
0			

6 StealthChop™

StealthChop is an extremely quiet mode of operation for stepper motors. It is based on a voltage mode PWM. In case of standstill and at low velocities, the motor is absolutely noiseless. Thus, StealthChop operated stepper motor applications are very suitable for indoor or home use. The motor operates free of vibration at low velocities. With StealthChop, the motor current is applied by driving a certain effective voltage into the coil, using a voltage mode PWM. With the enhanced StealthChop2, the driver automatically adapts to the application for best performance. No more configurations are required. Optional configuration allows for tuning the setting in special cases, or for storing initial values for the automatic adaptation algorithm. For high velocity drives SpreadCycle should be considered in combination with StealthChop.

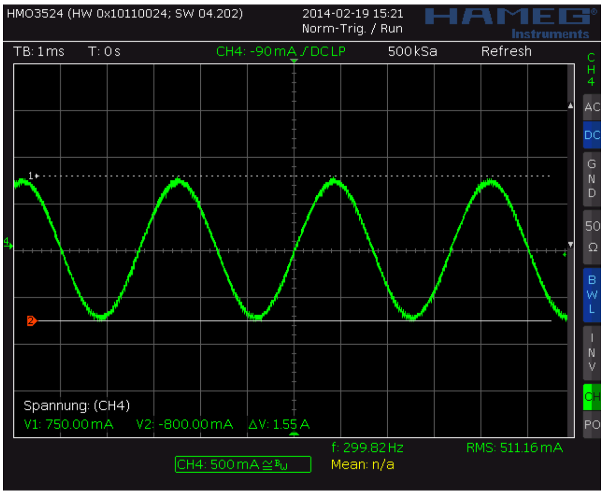


Figure 6.1 Motor coil sine wave current with StealthChop (measured with current probe)

6.1 Automatic Tuning

StealthChop2 integrates an automatic tuning procedure (AT), which adapts the most important operating parameters to the motor automatically. This way, StealthChop2 allows high motor dynamics and supports powering down the motor to very low currents. Just two steps have to be respected by the motion controller for best results: Start with the motor in standstill but powered with nominal run current (AT#1). Move the motor at a medium velocity, e.g., as part of a homing procedure (AT#2). Figure 6.2 shows the tuning procedure.

Border conditions for AT#1 and AT#2 are shown in the following table:

AUTOMATIC TUNING TIMING AND BORDER CONDITIONS

Step	**Parameter**	**Conditions**	**Required Duration**
AT#1	*PWM_ OFS_AUTO*	- Motor in standstill and actual current scale (*CS*) is identical to run current (*IRUN*). - If standstill reduction is enabled, an initial step pulse switches the drive back to run current or set *IHOLD* to *IRUN*. - Pin VS at operating level. *Attention: Driver may reduce chopper frequency during AT#1. Use reduced standstill current IHOLD<IRUN to prevent extended periods of time at lower chopper frequency*	≤ 2^20+2*2^18 t_{CLK}, ≤ 130ms (with internal clock)
AT#2	*PWM_ GRAD_AUTO*	- Move motor at a velocity, where a significant amount of back EMF is generated and where the full run current can be reached. Conditions: - 1.5**PWM_OFS_AUTO**(*IRUN*+1)/32 < *PWM_SCALE_SUM* < 4**PWM_OFS_AUTO**(*IRUN*+1)/32 - *PWM_SCALE_SUM* < 255. *Hint:* A typical range is 60-300 RPM.	8 fullsteps are required for a change of +/-1. For a typical motor with *PWM_GRAD_AUTO* optimum at 50 or less, up to 400 fullsteps are required when starting from default value 0.

Determine best conditions for automatic tuning with the evaluation board.
Monitor *PWM_SCALE_AUTO* going down to zero during the constant velocity phase in AT#2 tuning. This indicates a successful tuning.

Attention:
Operating in StealthChop without proper tuning can lead to high motor currents during a deceleration ramp, especially with low resistive motors and fast deceleration settings. Follow the automatic tuning process and check optimum tuning conditions using the evaluation board. It is recommended to use an initial value for settings *PWM_OFS* and *PWM_GRAD* determined per motor type.
Protect the power stage and supply by additionally tuning the overcurrent protection.

Known Limitations for non-A-version, only:
Successful completion of AT#1 tuning phase is not safely detected by the TMC2160. It will require multiple motor start / stop events to safely detect completion.
Successful determination is mandatory for AT#2: Tuning of *PWM_GRAD* will not start when AT#1 has not completed.
Successful completion of AT#1 and AT#2 only can be checked by monitoring *PWM_SCALE_AUTO* approaching 0 during AT#2 motion.

Solution a):
Complete automatic tuning phase AT#1 process, by using a slow-motion sequence which leads to standstill detection in between of each two steps. Use a velocity of 8 (6 Hz) or lower and execute minimum 10 steps during AT#1 phase.

Solution b):
Store initial parameters for *PWM_GRAD_AUTO* for the application. Therefore, use the motor and operating conditions determined for the application and do a complete automatic tuning sequence (refer to *a)*). Store the resulting *PWM_GRAD_AUTO* value and use it for initialization of *PWM_GRAD*. With this, tuning of AT#2 phase is not mandatory in the application and can be skipped. Automatic tuning will further optimize settings during operation. Combine with a) if desired.

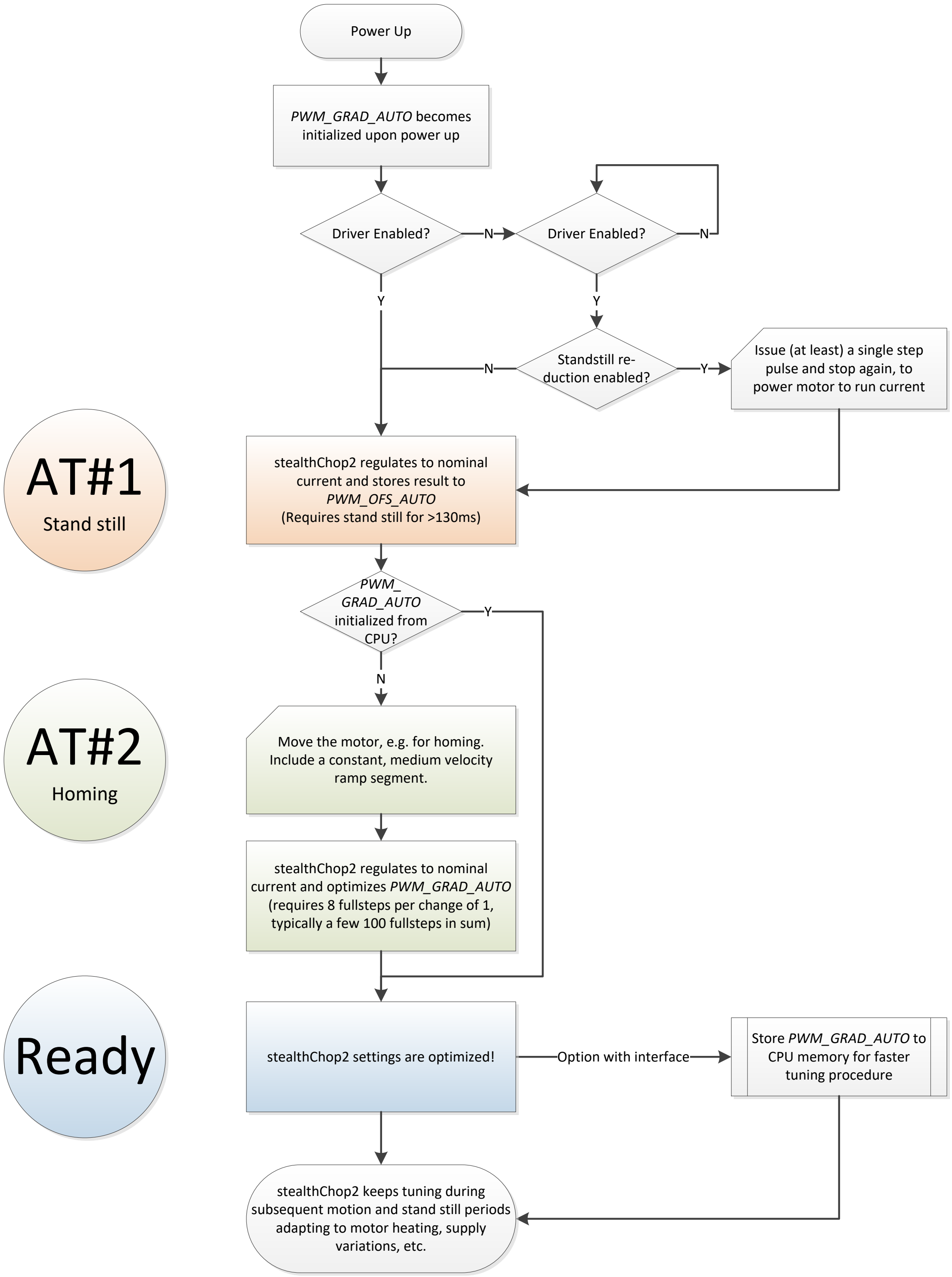


Figure 6.2 StealthChop2 automatic tuning procedure

Attention
Modifying *GLOBALSCALER* or VS voltage invalidates the result of the automatic tuning process. However, automatic tuning adapts to changed conditions whenever AT#1 or AT#2 conditions are fulfilled. Modifying VS is no problem with sinking supply voltage, i.e., due to the battery running low, as the regulator corrects by increasing the PWM value. However, with significantly increasing supply voltage, motor current rises, as the lower regulator limit is given by the result of the last AT#1 phase. Take this into account, when experimenting with a lab supply and modifying supply voltage.

6.2 StealthChop Options

To match the motor current to a certain level, the effective PWM voltage becomes scaled depending on the actual motor velocity. Several additional factors influence the required voltage level to drive the motor at the target current: The motor resistance, its back EMF (i.e., directly proportional to its velocity) as well as the actual level of the supply voltage. Two modes of PWM regulation are provided: The automatic tuning mode (AT) using current feedback (*pwm_autoscale* = 1, *pwm_autograd* = 1) and a feed forward velocity-controlled mode (*pwm_autoscale* = 0). The feed forward velocity-controlled mode will not react to a change of the supply voltage or to events like a motor stall, but it provides very stable amplitude. It does not use nor require any means of current measurement. This is perfect when motor type and supply voltage are well known. Therefore, we recommend the automatic mode, unless current regulation is not satisfying in the given operating conditions.

> It is recommended to use application specific initial tuning parameters, fitting the motor type and supply voltage. Additionally, operate in automatic tuning mode (*pwm_autoscale*=1) to respond to parameter change, e.g., due to motor heat-up or change of supply voltage.
>
> *Hint:* To reduce amplitude jitter, use pre-determined *PWM_GRAD* and set *pwm_autograd = 0*.
>
> Non-automatic mode (*pwm_autoscale*=0) should be considered only with well-known motor and operating conditions. In this case, careful programming via the interface is required. The operating parameters *PWM_GRAD* and *PWM_OFS* can be determined in automatic tuning mode initially.

The StealthChop PWM frequency can be chosen in four steps to adapt the frequency divider to the frequency of the clock source. A setting in the range of 20-50kHz is good for most applications. It balances low current ripple and good higher velocity performance vs. dynamic power dissipation.

CHOICE OF PWM FREQUENCY FOR STEALTHCHOP

Clock frequency f_{CLK}	**PWM_FREQ=%00** f_{PWM}=2/1024 f_{CLK}	**PWM_FREQ=%01** f_{PWM}=2/683 f_{CLK}	**PWM_FREQ=%10** f_{PWM}=2/512 f_{CLK}	**PWM_FREQ=%11** f_{PWM}=2/410 f_{CLK}
18MHz	35.2kHz	52.7kHz	70.3kHz	87.8kHz
16MHz	31.3kHz	46.9kHz	62.5kHz	78.0kHz
12MHz (internal)	23.4kHz	35.1kHz	46.9kHz	58.5kHz
10MHz	19.5kHz	29.3kHz	39.1kHz	48.8kHz
8MHz	15.6kHz	23.4kHz	31.2kHz	39.0kHz

Table 6.1 Choice of PWM frequency – green / light green: recommended

6.3 StealthChop Current Regulator

In StealthChop voltage PWM mode, the autoscaling function (*pwm_autoscale* = 1, *pwm_autograd* = 1) regulates the motor current to the desired current setting. Automatic scaling is used as part of the automatic tuning process (AT), and for subsequent tracking of changes within the motor parameters. The driver measures the motor current during the chopper on time and uses a proportional regulator to regulate *PWM_SCALE_AUTO* in order match the motor current to the target current. *PWM_REG* is the proportionality coefficient for this regulator. Basically, the proportionality coefficient should be as small as possible to get a stable and soft regulation behavior, but it must be large enough to allow the driver to quickly react to changes caused by variation of parameters (e.g., change of mechanical load). During initial tuning step AT#2, *PWM_REG* also compensates for the change of motor velocity. Therefore, a high acceleration during AT#2 will require a higher setting of *PWM_REG*. With careful selection of homing velocity and acceleration, a minimum setting of the regulation gradient often is sufficient (*PWM_REG*=1). *PWM_REG* setting should be optimized for the fastest required acceleration and deceleration ramp (compare Figure 6.3 and Figure 6.4). The quality of the setting *PWM_REG* in phase AT#2 and the finished automatic tuning procedure (or non-automatic settings for *PWM_OFS* and *PWM_GRAD*) can be examined when monitoring motor current during an acceleration phase Figure 6.5.

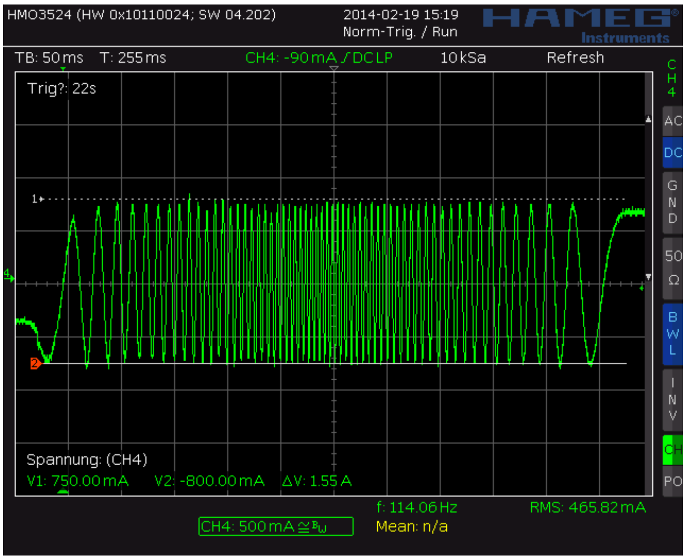


Figure 6.3 Scope shot: good setting for PWM_REG

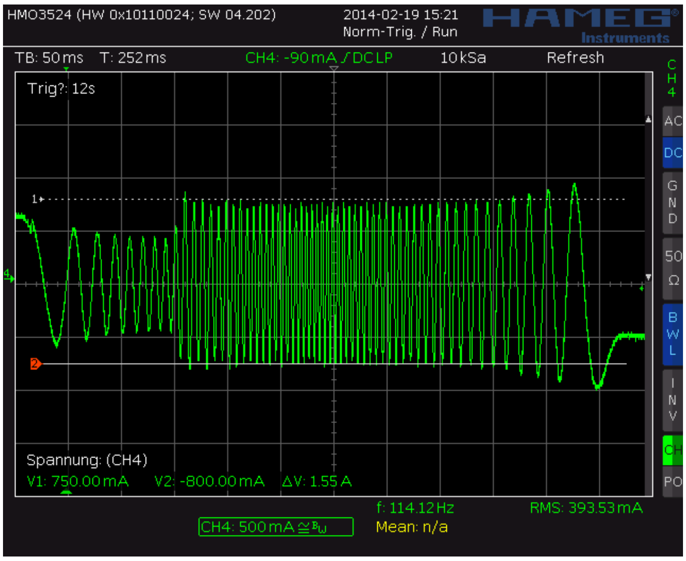


Figure 6.4 Scope shot: too small setting for PWM_REG during AT#2

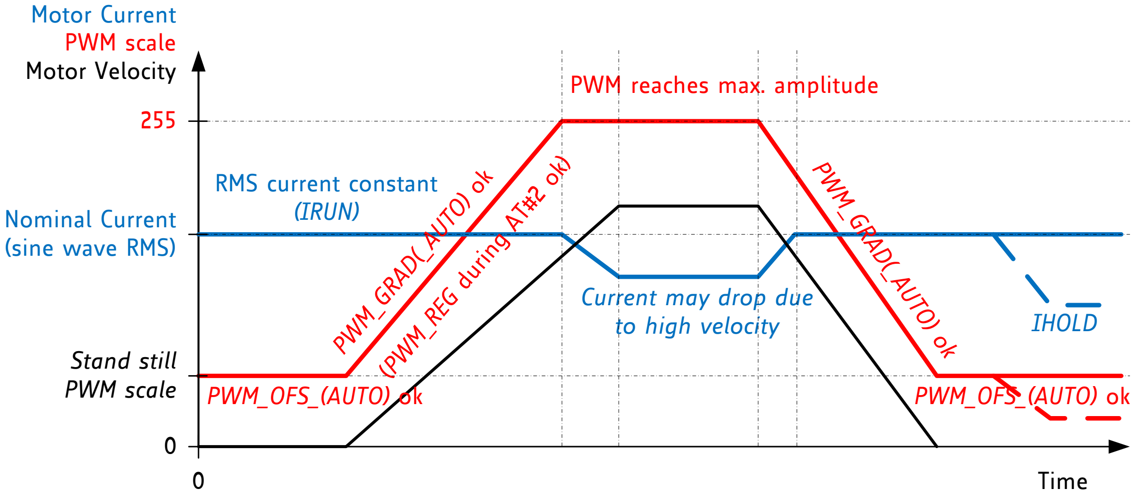


Figure 6.5 Successfully determined PWM_GRAD(_AUTO) and PWM_OFS(_AUTO)

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 18.

6.3.1 Lower Current Limit

The StealthChop current regulator imposes a lower limit for motor current regulation. As the coil current can be measured in the shunt resistor during chopper on phase only, a minimum chopper duty cycle allowing coil current regulation is given by the blank time as set by *TBL* and by the chopper frequency setting. Therefore, the motor specific minimum coil current in StealthChop autoscaling mode rises with the supply voltage and with the chopper frequency. A lower blanking time allows a lower current limit. It is important for the correct determination of *PWM_OFS_AUTO*, that in AT#1 the run current set by the sense resistor, *GLOBALSCALER* and *IRUN* is well within the regulation range. Lower currents (e.g., for standstill power down) are automatically realized based on *PWM_OFS_AUTO* and *PWM_GRAD_AUTO* respectively based on *PWM_OFS* and *PWM_GRAD* with non-automatic current scaling. The freewheeling option allows going to zero motor current.

Lower motor coil current limit for StealthChop2 automatic tuning:

$$I_{Lower\ Limit} = t_{BLANK} * f_{PWM} * \frac{V_M}{R_{COIL}}$$

With V_M the motor supply voltage and R_{COIL} the motor coil resistance.
$I_{Lower\ Limit}$ can be treated as a thumb value for the minimum nominal *IRUN* motor current setting. In case the lower current limit is not sufficient to reach the desired setting, the driver will retry with a lower chopper frequency in step AT#1, only.
f_{PWM} is the chopper frequency as determined by setting *PWM_FREQ*. In AT#1, the driver tries a lower, (roughly half frequency), in case it cannot reach the current. The frequency will remain active in standstill, while currentscale *CS=IRUN.* With automatic standstill reduction, this is a short moment.

EXAMPLE:

A motor has a coil resistance of 5Ω, the supply voltage is 24V. With *TBL*=%01 and *PWM_FREQ*=%00, t_{BLANK} is 24 clock cycles, f_{PWM} is 2/(1024 clock cycles):

$$I_{Lower\ Limit} = 24\ t_{CLK} * \frac{2}{1024\ t_{CLK}} * \frac{24V}{5\Omega} = \frac{24}{512} * \frac{24V}{5\Omega} = 225mA$$

This means, the motor target current for automatic tuning must be 225mA or more, taking into account all relevant settings. This lower current limit also applies for modification of the motor current via the *GLOBALSCALER*.

Attention
For automatic tuning, a lower coil current limit applies.
IRUN ≥ 8: Current settings for *IRUN* below 8 do not work with automatic tuning.
$I_{LOWER\ LIMIT}$: The motor current in automatic tuning phase AT#1 must exceed this lower limit. Calculate $I_{LOWER\ LIMIT}$ or measure it using a current probe. Setting the motor run-current or hold-current below the lower current limit during operation by modifying *IRUN* and *IHOLD* is possible after successful automatic tuning.

The lower current limit also limits the capability of the driver to respond to changes of *GLOBALSCALER*.

6.4 Velocity Based Scaling

Velocity based scaling scales the StealthChop amplitude based on the time between each two steps (*TSTEP*) measured in clock cycles. This concept basically does not require a current measurement, because no regulation loop is necessary. A pure velocity-based scaling is available via programming, only, when setting *pwm_autoscale* = 0. The basic idea is to have a linear approximation of the voltage required to drive the target current into the motor. The stepper motor has a certain coil resistance and thus needs a certain voltage amplitude to yield a target current based on the basic formula I=U/R. With R being the coil resistance, U the supply voltage scaled by the PWM value, the current I results. The initial value for *PWM_OFS* can be calculated:

$$PWM\_OFS = \frac{374 * R_{COIL} * I_{COIL}}{V_M}$$

With V_M the motor supply voltage and I_{COIL} the target RMS current

The effective PWM voltage U_{PWM} (1/SQRT(2) x peak value) results considering the 8 bit resolution and 248 sine wave peak for the actual PWM amplitude shown as *PWM_SCALE*:

$$U_{PWM} = V_M * \frac{PWM\_SCALE}{256} * \frac{248}{256} * \frac{1}{\sqrt{2}} = V_M * \frac{PWM\_SCALE}{374}$$

With rising motor velocity, the motor generates an increasing back EMF voltage. The back EMF voltage is proportional to the motor velocity. It reduces the PWM voltage effective at the coil resistance and thus current decreases. The TMC2160 provides a second velocity dependent factor (*PWM_GRAD*) to compensate for this. The overall effective PWM amplitude (*PWM_SCALE_SUM*) in this mode automatically is calculated in dependence of the microstep frequency as:

$$PWM\_SCALE\_SUM = PWM\_OFS + PWM\_GRAD * 256 * \frac{f_{STEP}}{f_{CLK}}$$

With f_{STEP} being the microstep frequency for 256 microstep resolution equivalent
and f_{CLK} the clock frequency supplied to the driver or the actual internal frequency

As a first approximation, the back EMF subtracts from the supply voltage and thus the effective current amplitude decreases. This way, a first approximation for *PWM_GRAD* setting can be calculated:

$$PWM\_GRAD = C_{BEMF}\left[\frac{V}{\frac{rad}{s}}\right] * 2\pi * \frac{f_{CLK} * 1.46}{V_M * MSPR}$$

C_{BEMF} is the back EMF constant of the motor in Volts per radian/second.
MSPR is the number of microsteps per rotation assuming a 256 microstep resolution, e.g., 51200 = 256µsteps multiplied by 200 fullsteps for a 1.8° motor.

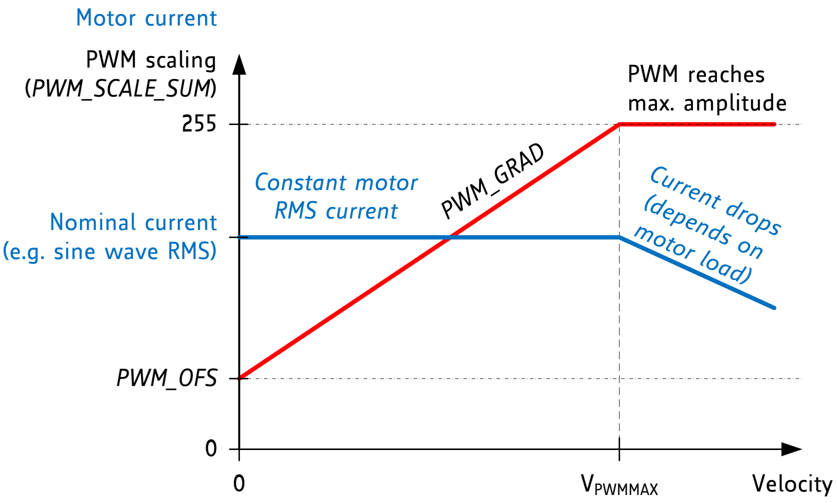


Figure 6.6 Velocity based PWM scaling (pwm_autoscale=0)

Hint
The values for *PWM_OFS* and *PWM_GRAD* can easily be optimized by tracing the motor current with a current probe on the oscilloscope. Alternatively, automatic tuning determines these values, and they can be read out from *PWM_OFS_AUTO* and *PWM_GRAD_AUTO*.

UNDERSTANDING THE BACK EMF CONSTANT OF A MOTOR

The back EMF constant is the voltage a motor generates when turned with a certain velocity. Often motor datasheets do not specify this value, as it can be deducted from motor torque and coil current rating. Within SI units, the numeric value of the back EMF constant C_{BEMF} has the same numeric value as the numeric value of the torque constant. For example, a motor with a torque constant of 1 Nm/A would have a C_{BEMF} of 1V/rad/s. Turning such a motor with 1 rps (1 rps = 1 revolution per second = 6.28 rad/s) generates a back EMF voltage of 6.28V. Thus, the back EMF constant can be calculated as:

$$C_{BEMF}\left[\frac{V}{rad/s}\right] = \frac{HoldingTorque[Nm]}{2 * I_{COILNOM}[A]}$$

$I_{COILNOM}$ is the motor's rated phase current for the specified holding torque
HoldingTorque is the motor specific holding torque, i.e., the torque reached at $I_{COILNOM}$ on both coils. The torque unit is [Nm] where 1Nm = 100Ncm = 1000mNm.
The voltage is valid as RMS voltage per coil, thus the nominal current is multiplied by 2 in this formula, since the nominal current assumes a full step position, with two coils operating.

6.5 Combine StealthChop and SpreadCycle

For applications requiring high velocity motion, SpreadCycle may bring more stable operation in the upper velocity range. To combine no-noise operation with highest dynamic performance, the TMC2160 allows combining StealthChop and SpreadCycle based on a velocity threshold (Figure 6.7). With this, StealthChop is only active at low velocities.

Hint
Operate the motor within your application when exploring StealthChop. Motor performance often is better with a mechanical load, because it prevents the motor from stalling due mechanical oscillations which can occur without load.

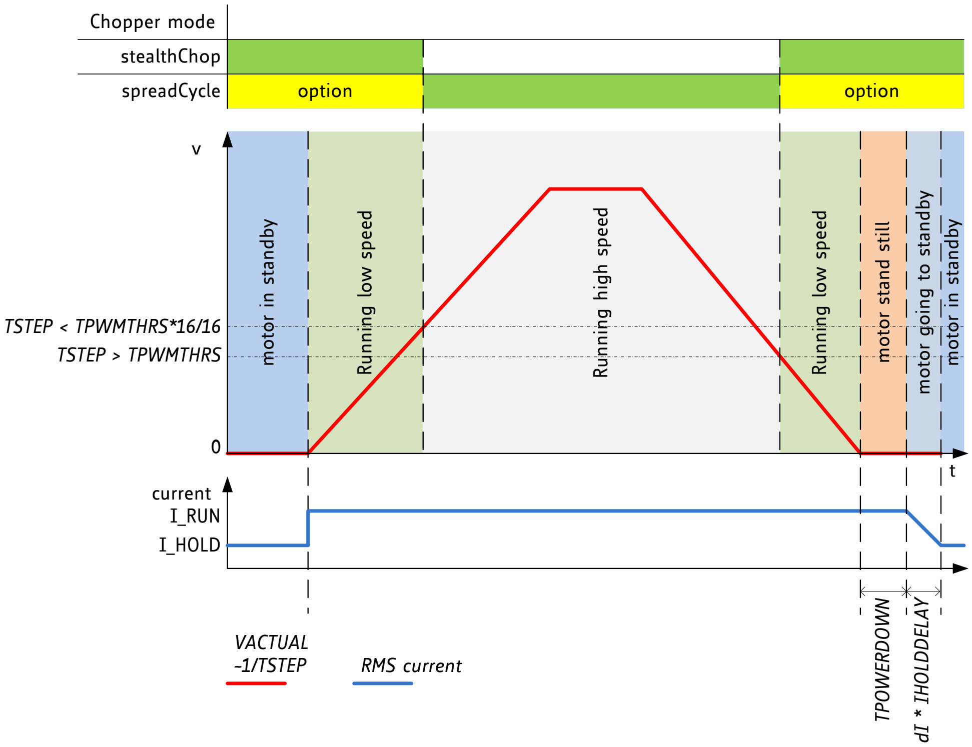


Figure 6.7 TPWMTHRS for optional switching to SpreadCycle

As a first step, both chopper principles should be parameterized and optimized individually. In a next step, a transfer velocity has to be fixed. For example, StealthChop operation is used for precise low speed positioning, while SpreadCycle shall be used for highly dynamic motion. *TPWMTHRS* determines the transition velocity. Read out *TSTEP* when moving at the desired velocity and program the resulting value to *TPWMTHRS*. Use a low transfer velocity to avoid a jerk at the switching point.

A jerk occurs when switching at higher velocities, because the back-EMF of the motor (which rises with the velocity) causes a phase shift of up to 90° between motor voltage and motor current. So when switching at higher velocities between voltage PWM and current PWM mode, this jerk will occur with increased intensity. A high jerk may even produce a temporary overcurrent condition (depending on the motor coil resistance). At low velocities (e.g., 1 to a few 10 RPM), it can be completely neglected for most motors. Therefore, consider the switching jerk when choosing *TPWMTHRS*. Set *TPWMTHRS* zero if you want to work with StealthChop only.

When enabling the StealthChop mode the first time using automatic current regulation, the motor must be at stand still to allow a proper current regulation. When the drive switches to StealthChop at a higher velocity, StealthChop logic stores the last current regulation setting until the motor returns to a lower velocity again. This way, the regulation has a known starting point when returning to a lower velocity, where StealthChop becomes re-enabled. Therefore, neither the velocity threshold nor the supply voltage must be considerably changed during the phase while the chopper is switched to a different mode, because otherwise the motor might lose steps, or the instantaneous current might be too high or too low.

A motor stall or a sudden change in the motor velocity may lead to the driver detecting a short circuit or to a state of automatic current regulation, from which it cannot recover. Clear the error flags and restart the motor from zero velocity to recover from this situation.

Hint
Start the motor from standstill when switching on StealthChop the first time and keep it stopped for at least 128 chopper periods to allow StealthChop to do initial standstill current control.

6.6 Flags in StealthChop

As StealthChop uses voltage mode driving, status flags based on current measurement respond slower, respectively the driver reacts delayed to sudden changes of back EMF, like on a motor stall.

Attention
A motor stall, or abrupt stop of the motion during operation in StealthChop can lead to a overcurrent condition. Depending on the previous motor velocity, and on the coil resistance of the motor, it significantly increases motor current for a time of several 10ms. With low velocities, where the back EMF is just a fraction of the supply voltage, there is no danger of triggering the short detection.

Hint
Tune low side driver overcurrent detection to safely trigger upon motor stall, when using StealthChop. This will avoid high peak current draw from the power supply.

6.6.1 Open Load Flags

In StealthChop, status information is different from the cycle-by-cycle regulated SpreadCycle mode. OLA and OLB show if the current regulation sees that the nominal current is reached on both coils.

- Flickering OLA or OLB can result from asymmetries in the sense resistors or the motor coils.
- An interrupted motor coil leads to a continuously active open load flag for the coil.
- One or both flags go active, if the current regulation did not succeed in scaling up to the full target current within the last few fullsteps (e.g., when no motor is attached, or a high velocity causes a back-EMF exceeding the supply voltage).

If desired, do an on-demand open load test using the SpreadCycle chopper, as it delivers the safest result. With StealthChop, *PWM_SCALE_SUM* can be checked to detect the correct coil resistance.

Attention
In standstill with StealthChop automatic scaling, an interrupted coil connection on one coil can lead to the current regulator increasing the PWM duty cycle up to its limit. This behavior results from the regulator measuring the coil with the higher target current, only. Thus, it may lead to overcurrent on the other coil. To prevent this situation, check for an open load situation (using SpreadCycle) prior to operation in StealthChop.

6.6.2 PWM_SCALE_SUM Informs about the Motor State

Information about the motor state is available with automatic scaling by reading out *PWM_SCALE_SUM*. As this parameter reflects the actual voltage required to drive the target current into the motor, it depends on several factors: motor load, coil resistance, supply voltage, and current setting. Therefore, an evaluation of the *PWM_SCALE_SUM* value allows checking the motor operation point. When reaching the limit (255), the current regulator cannot sustain the full motor current, e.g., due to a drop in supply volage.

6.7 Freewheeling and Passive Braking

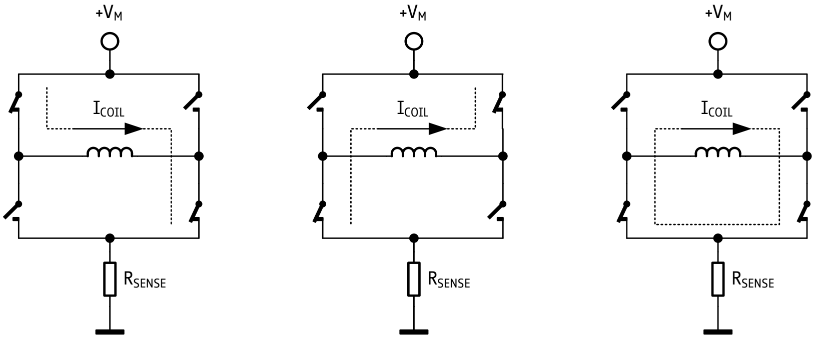
StealthChop provides different options for motor standstill. Enable these options by setting *IHOLD* to zero and choosing the desired *FREEWHEEL* setting. The desired option becomes enabled after the delay specified by *TPOWERDOWN* and *IHOLDDELAY*. Current regulation becomes frozen once the motor target current is at zero to ensure a quick startup. With the freewheeling options, both freewheeling and passive braking can be realized. Passive braking is an energy efficient eddy current motor braking with no active current driven into the coils. However, passive braking will allow slow turning of the motor when a continuous torque is applied.

PARAMETERS RELATED TO STEALTHCHOP

Parameter	Description	Setting	Comment
en_spread_ cycle	General disable for use of StealthChop (register *GCONF*). The input SPREAD is XORed to this flag.	1	Do not use StealthChop
		0	StealthChop enabled
TPWMTHRS	Specifies the upper velocity for operation in StealthChop. Entry the *TSTEP* reading (time between two microsteps) when operating at the desired threshold velocity.	0 … 1048575	StealthChop is disabled if *TSTEP* falls *TPWMTHRS*
PWM_LIM	Limiting value for limiting the current jerk when switching from SpreadCycle to StealthChop. Reduce the value to yield a lower current jerk.	0 … 15	Upper four bits of 8 bit amplitude limit (Default=12)
pwm_ autoscale	Enable automatic current scaling using current measurement. If off, use forward controlled velocity-based mode.	0	Forward controlled mode
		1	Automatic scaling with current regulator
pwm_ autograd	Enable automatic tuning of *PWM_GRAD_AUTO*	0	disable, use *PWM_GRAD* from register instead
		1	enable
PWM_FREQ	PWM frequency selection. Use the lowest setting giving good results. The frequency measured at each of the chopper outputs is half of the effective chopper frequency f_{PWM}.	0	$f_{PWM}=2/1024\ f_{CLK}$
		1	$f_{PWM}=2/683\ f_{CLK}$
		2	$f_{PWM}=2/512\ f_{CLK}$
		3	$f_{PWM}=2/410\ f_{CLK}$
PWM_REG	User defined PWM amplitude regulation loop P-coefficient. A higher value leads to a higher adaptation speed when *pwm_autoscale*=1.	1 … 15	Results in 0.5 to 7.5 steps for *PWM_SCALE_AUTO* regulator per fullstep
PWM_OFS	User defined PWM amplitude (offset) for velocity-based scaling and initialization value for automatic tuning of *PWM_OFFS_AUTO*.	0 … 255	*PWM_OFS*=0 disables linear current scaling based on current setting
PWM_GRAD	User defined PWM amplitude (gradient) for velocity-based scaling and initialization value for automatic tuning of *PWM_GRAD_AUTO*.	0 … 255	
FREEWHEEL	Stand still option when motor current setting is zero (*I_HOLD*=0). Only available with StealthChop enabled. The freewheeling option makes the motor easy movable, while both coil short options realize a passive brake.	0	Normal operation
		1	Freewheeling
		2	Coil short via LS drivers
		3	Coil short cia HS drivers
PWM_SCALE _AUTO	Read back of the actual StealthChop voltage PWM scaling correction as determined by the current regulator. Shall regulate close to 0 during tuning.	-255 … 255	(read only) Scaling value becomes frozen when operating in SpreadCycle
PWM_GRAD _AUTO *PWM_OFS _AUTO*	Allow monitoring of the automatic tuning and determination of initial values for *PWM_OFS* and *PWM_GRAD*.	0 … 255	(read only)
TOFF	General enable for the motor driver, the actual value does not influence StealthChop	0	Driver off
		1 … 15	Driver enabled
TBL	Comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. Choose a setting of 1 or 2 for typical applications. For higher capacitive loads, 3 may be required. Lower settings allow StealthChop to regulate down to lower coil current values.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	36 t_{CLK}
		3	54 t_{CLK}

7 SpreadCycle and Classic Chopper

While StealthChop is a voltage mode PWM controlled chopper, SpreadCycle is a cycle-by-cycle current control. Therefore, it can react extremely fast to changes in motor velocity or motor load. The currents through both motor coils are controlled using choppers. The choppers work independently of each other. In Figure 7.1 the different chopper phases are shown.



On Phase:
current flows in
direction of target
current

Fast Decay Phase:
current flows in
opposite direction
of target current

Slow Decay Phase:
current re-circulation

Figure 7.1 Chopper phases

Although the current could be regulated using only on phases and fast decay phases, insertion of the slow decay phase is important to reduce electrical losses and current ripple in the motor. The duration of the slow decay phase is specified in a control parameter and sets an upper limit on the chopper frequency. The current comparator can measure coil current during phases when the current flows through the sense resistor, but not during the slow decay phase, so the slow decay phase is terminated by a timer. The on phase is terminated by the comparator when the current through the coil reaches the target current. The fast decay phase may be terminated by either the comparator or another timer.

When the coil current is switched, spikes at the sense resistors occur due to charging and discharging parasitic capacitances. During this time, typically one or two microseconds, the current cannot be measured. Blanking is the time when the input to the comparator is masked to block these spikes.

There are two cycle-by-cycle chopper modes available: a new high-performance chopper algorithm called SpreadCycle and a proven constant off-time chopper mode. The constant off-time mode cycles through three phases: on, fast decay, and slow decay. The SpreadCycle mode cycles through four phases: on, slow decay, fast decay, and a second slow decay.

The chopper frequency is an important parameter for a chopped motor driver. A too low frequency might generate audible noise. A higher frequency reduces current ripple in the motor, but with a too high frequency magnetic losses may rise. Also, power dissipation in the driver rises with increasing frequency due to the increased influence of switching slopes causing dynamic dissipation. Therefore, a compromise needs to be found. Most motors are optimally working in a frequency range of 16 kHz to 30 kHz. The chopper frequency is influenced by a number of parameter settings as well as by the motor inductivity and supply voltage.

Hint

A chopper frequency in the range of 16 kHz to 30 kHz gives a good result for most motors when using SpreadCycle. A higher frequency leads to increased switching losses.

Three parameters are used for controlling both chopper modes:

Parameter	Description	Setting	Comment
TOFF	Sets the slow decay time (*off time*). This setting also limits the maximum chopper frequency. For operation with StealthChop, this parameter is not used, but it is required to enable the motor. In case of operation with StealthChop only, any setting is OK. Setting this parameter to zero completely disables all driver transistors and the motor can free-wheel.	0	chopper off
		1…15	off time setting N_{CLK}= 24 + 32**TOFF* (1 will work with minimum blank time of 24 clocks)
TBL	Selects the comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. For most applications, a setting of 1 or 2 is good. For highly capacitive loads, e.g., when filter networks are used, a setting of 2 or 3 will be required.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	36 t_{CLK}
		3	54 t_{CLK}
chm	Selection of the *chopper mode*	0	SpreadCycle
		1	classic const. off time
TPFD	Adds passive fast decay time after bridge polarity change. Starting from 0, increase value, in case the motor suffers from mid-range resonances.	0…15	Fast decay time in multiple of 128 clocks (128 clocks are roughly 10µs)

7.1 SpreadCycle Chopper

The SpreadCycle (patented) chopper algorithm is a precise and simple to use chopper mode which automatically determines the optimum length for the fast-decay phase. The SpreadCycle will provide superior microstepping quality even with default settings. Several parameters are available to optimize the chopper to the application.

Each chopper cycle is comprised of an on phase, a slow decay phase, a fast decay phase and a second slow decay phase (see Figure 7.3). The two slow decay phases and the two blank times per chopper cycle put an upper limit to the chopper frequency. The slow decay phases typically make up for about 30%-70% of the chopper cycle in standstill and are important for low motor and driver power dissipation.

Calculation of a starting value for the slow decay time *TOFF*:

EXAMPLE:

Target Chopper frequency: 25kHz.
Assumption: Two slow decay cycles make up for 50% of overall chopper cycle time

$$t_{OFF} = \frac{1}{25kHz} * \frac{50}{100} * \frac{1}{2} = 10\mu s$$

For the *TOFF* setting this means:

$$TOFF = (t_{OFF} * f_{CLK} - 24)/32$$

With 12 MHz clock this gives a setting of TOFF=3, i.e. 3.
With 16 MHz clock this gives a setting of TOFF=4.25, i.e. 4 or 5.

The hysteresis start setting forces the driver to introduce a minimum amount of current ripple into the motor coils. The current ripple must be higher than the current ripple which is caused by resistive losses in the motor in order to give best microstepping results. This will allow the chopper to precisely regulate the current both for rising and for falling target current. The time required to introduce the current ripple into the motor coil also reduces the chopper frequency. Therefore, a higher hysteresis setting will lead to a lower chopper frequency. The motor inductance limits the ability of the chopper to follow a changing motor current. Further the duration of the on phase and the fast decay must be longer than the blanking time because the current comparator is disabled during blanking.

It is easiest to find the best setting by starting from a low hysteresis setting (e.g., *HSTRT*=0, *HEND*=0) and increasing *HSTRT*, until the motor runs smoothly at low velocity settings. This can best be checked when measuring the motor current either with a current probe or by probing the sense resistor voltages (see Figure 7.2). Checking the sine wave shape near zero transition will show a small ledge between both half waves in case the hysteresis setting is too small. At medium velocities (i.e., 100 to 400 fullsteps per second), a too low hysteresis setting will lead to increased humming and vibration of the motor.

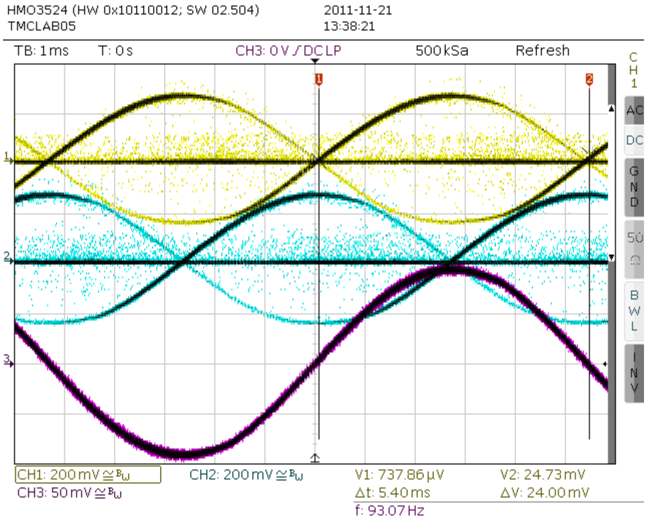


Figure 7.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)

A too high hysteresis setting will lead to reduced chopper frequency and increased chopper noise but will not yield any benefit for the wave shape.

Quick Start For a quick start, see the Quick Configuration Guide in chapter 18. For detail procedure see Application Note AN001 - *Parameterization of SpreadCycle*

As experiments show, the setting is quite independent of the motor, because higher current motors typically also have a lower coil resistance. Therefore, choosing a low to medium default value for the hysteresis (for example, effective hysteresis = 4) normally fits most applications. The setting can be optimized by experimenting with the motor: A too low setting will result in reduced microstep accuracy, while a too high setting will lead to more chopper noise and motor power dissipation. When measuring the sense resistor voltage in motor standstill at a medium coil current with an oscilloscope, a too low setting shows a fast decay phase not longer than the blanking time. When the fast decay time becomes slightly longer than the blanking time, the setting is optimum. You can reduce the off-time setting, if this is hard to reach.

The hysteresis principle could in some cases lead to the chopper frequency becoming too low, e.g. when the coil resistance is high when compared to the supply voltage. This is avoided by splitting the hysteresis setting into a start setting (*HSTRT+HEND*) and an end setting (*HEND*). An automatic hysteresis decrementer (HDEC) interpolates between both settings, by decrementing the hysteresis value stepwise each 16 system clocks. At the beginning of each chopper cycle, the hysteresis begins with a value which is the sum of the start and the end values (*HSTRT+HEND*), and decrements during the cycle, until either the chopper cycle ends, or the hysteresis end value (*HEND*) is reached. This way, the chopper frequency is stabilized at high amplitudes and low supply voltage situations, if the frequency gets too low. This avoids the frequency reaching the audible range.

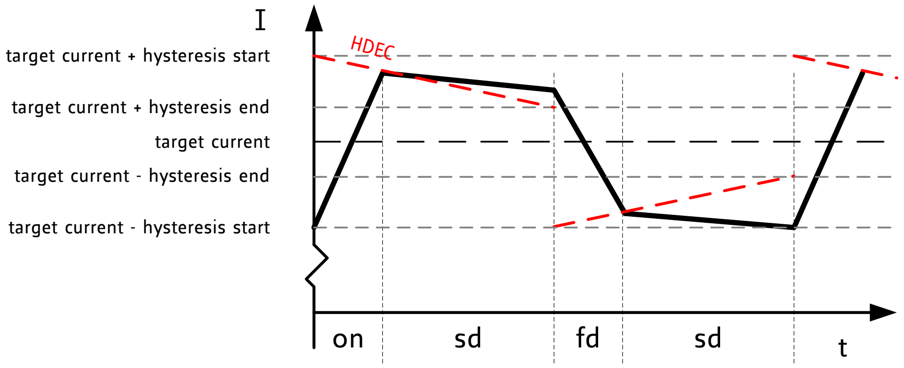


Figure 7.3 SpreadCycle chopper scheme showing coil current during a chopper cycle

Two parameters control SpreadCycle mode:

Parameter	**Description**	**Setting**	**Comment**
HSTRT	*Hysteresis start* setting. This value is an offset from the hysteresis end value *HEND*.	0…7	*HSTRT*=1…8 This value adds to HEND.
HEND	*Hysteresis end* setting. Sets the hysteresis end value after a number of decrements. The sum *HSTRT+HEND* must be ≤16. At a current setting of max. 30 (amplitude reduced to 240), the sum is not limited.	0…2	-3…-1: negative HEND
		3	0: zero HEND
		4…15	1…12: positive HEND

With HSTRT=0 and HEND=0, the hysteresis is 0 (off).

EXAMPLE:

A hysteresis of 4 has been chosen. You might decide to not use hysteresis decrement. In this case set:

HEND=6 (sets an effective end value of 6-3=3)
HSTRT=0 (sets minimum hysteresis, i.e. 1: 3+1=4)

In order to take advantage of the variable hysteresis, we can set most of the value to the HSTRT, i.e. 4, and the remaining 1 to hysteresis end. The resulting configuration register values are as follows:

HEND=0 (sets an effective end value of -3)
HSTRT=6 (sets an effective start value of hysteresis end +7: 7-3=4)

Hint
Highest motor velocities sometimes benefit from setting TOFF to 2 or 3 and a short TBL of 2 or 1.

7.2 Classic Constant Off Time Chopper

The classic constant off time chopper is an alternative to SpreadCycle. Perfectly tuned, it also gives good results. Also, the classic constant off time chopper (automatically) is used in combination with fullstepping in DcStep operation.

The classic constant off-time chopper uses a fixed-time fast decay following each on phase. While the duration of the on phase is determined by the chopper comparator, the fast decay time needs to be long enough for the driver to follow the falling slope of the sine wave, but it should not be so long that it causes excess motor current ripple and power dissipation. This can be tuned using an oscilloscope or evaluating motor smoothness at different velocities. A good starting value is a fast decay time setting similar to the slow decay time setting.

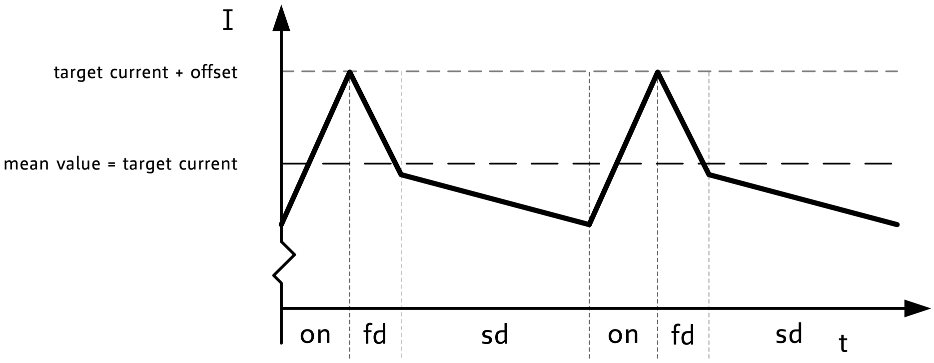
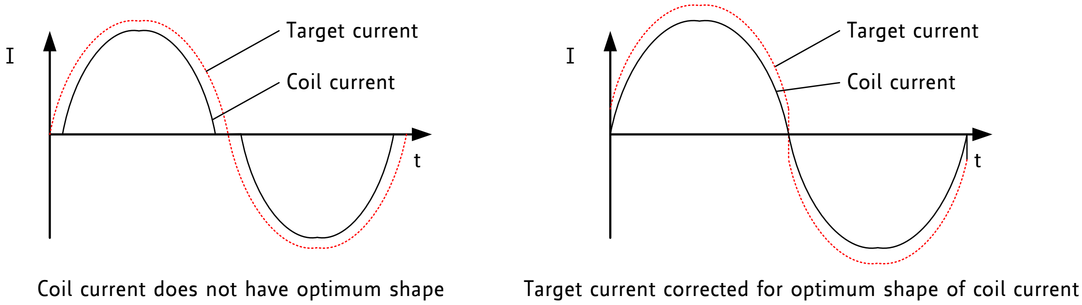


Figure 7.4 Classic const. off time chopper with offset showing coil current

After tuning the fast decay time, the offset should be tuned for a smooth zero crossing. This is necessary because the fast decay phase makes the absolute value of the motor current lower than the target current (see Figure 7.5). If the zero offset is too low, the motor stands still for a short moment during current zero crossing. If it is set too high, it makes a larger microstep. Typically, a positive offset setting is required for smoothest operation.



Coil current does not have optimum shape

Target current corrected for optimum shape of coil current

Figure 7.5 Zero crossing with classic chopper and correction using sine wave offset

Three parameters control constant off-time mode:

Parameter	**Description**	**Setting**	**Comment**
TFD (*fd3* & *HSTRT*)	*Fast decay time* setting. With CHM=1, these bits control the portion of fast decay for each chopper cycle.	0	slow decay only
		1…15	duration of fast decay phase
OFFSET (*HEND*)	*Sine wave offset*. With CHM=1, these bits control the sine wave offset. A positive offset corrects for zero crossing error.	0…2	negative offset: -3…-1
		3	no offset: 0
		4…15	positive offset 1…12
disfdcc	Selects usage of the *current comparator* for termination of the *fast decay* cycle. If current comparator is enabled, it terminates the fast decay cycle in case the current reaches a higher negative value than the actual positive value.	0	enable comparator termination of fast decay cycle
		1	end by time only

8 Selecting Sense Resistors

The TMC2160 provides several means to set the motor current: Sense resistors, *GLOBALSCALER* and currentscale *CS.* To adapt a drive to the motor, choose a sense-resistor value fitting or slightly exceeding the maximum desired current at 100% settings of the scalers. Fine-tune the current to the specific motor via the 8-bit *GLOBALSCALER*. Situation specific motor current adaptation is done by 5-bit scalers (actual scale can be read via *CS*), controlled by CoolStep, run- and hold current (*IRUN*, *IHOLD*). This makes the *CS* control compatible to other TRINAMIC ICs.

Set the desired maximum motor current by selecting an appropriate value for the sense resistor. The following table shows the RMS current values which are reached using standard resistors.

CHOICE OF R_{SENSE} AND RESULTING MAX. MOTOR CURRENT WITH *GLOBALSCALER*=0 (RESP. VALUE 256)		
R_{SENSE} [Ω]	***RMS current [A] (CS=31)***	***Sine wave peak current [A] (CS=31)***
0.22	1.1	1.5
0.15	1.6	2.2
0.12	2.0	2.8
0.10	2.3	3.3
0.075	3.1	4.4
0.066	3.5	5.0
0.050	4.7	6.6
0.033	7.1	10.0
0.022	10.6	15.0

Sense resistors should be carefully selected. The full motor current flows through the sense resistors. Due to chopper operation the sense resistors see pulsed current from the MOSFET bridges. Therefore, a low-inductance type such as film or composition resistors is required to prevent voltage spikes causing ringing on the sense voltage inputs leading to unstable measurement results. Also, a low-inductance, low-resistance PCB layout is essential. A massive ground plane is best. Please also refer to layout considerations in chapter 25.

The sense resistor sets the upper current which can be set by software settings *IRUN*, *IHOLD* and *GLOBALSCALER*. Choose the sense resistor value so that the maximum desired current (or slightly more) flows at the maximum current setting (*GLOBALSCALER* = 256 (0) and *IRUN* = 31).

CALCULATION OF RMS CURRENT

$$I_{RMS} = \frac{GLOBALSCALER}{256} * \frac{CS+1}{32} * \frac{V_{FS}}{R_{SENSE}} * \frac{1}{\sqrt{2}}$$

The momentary motor current is calculated by:

$$I_{MOT} = \frac{GLOBALSCALER}{256} * \frac{CUR_{A/B}}{248} * \frac{CS+1}{32} * \frac{V_{FS}}{R_{SENSE}}$$

GLOBALSCALER is the global current scaler. A setting of 0 is treated as full scale (256).
CS is the current scale setting as set by the *IHOLD* and *IRUN* and CoolStep.
V_{FS} is the full-scale voltage (please refer to electrical characteristics, V_{SRT}).
$CUR_{A/B}$ is the actual value from the internal sine wave table.
248 is the amplitude of the internal sine wave table.

The sense resistor needs to be able to conduct the peak motor coil current in motor standstill conditions unless standby power is reduced. Under normal conditions, the sense resistor conducts less than the coil RMS current, because no current flows through the sense resistor during the slow decay phases.

CALCULATION OF PEAK SENSE RESISTOR POWER DISSIPATION

$$P_{RSMAX} = I_{COIL}{}^{2} * R_{SENSE}$$

Hint

For best precision of current setting, it is advised to measure and fine tune the current in the application. Choose the sense resistors to the next value covering the desired motor current. Set *IRUN* to 31 corresponding 100% of the desired motor current and fine-tune motor current using *GLOBALSCALER*.

Attention

Be sure to use a symmetrical sense resistor layout and short and straight sense resistor traces of identical length. Well matching sense resistors ensure best performance.
A compact layout with massive ground plane is best to avoid parasitic resistance effects.

Parameter	Description	Setting	Comment
IRUN	Current scale when motor is running. Scales coil current values as taken from the internal sine wave table. For high precision motor operation, work with a current scaling factor in the range 16 to 31, because scaling down the current values reduces the effective microstep resolution by making microsteps coarser. This setting also controls the maximum current value set by CoolStep.	0 … 31	scaling factor 1/32, 2/32, … 32/32
IHOLD	Identical to *IRUN*, but for motor in stand still.		
IHOLD DELAY	Allows smooth current reduction from run current to hold current. *IHOLDDELAY* controls the number of clock cycles for motor power down after *TZEROWAIT* in increments of 2^18 clocks: 0=instant power down, 1..15: Current reduction delay per current step in multiple of 2^18 clocks. *Example:* When using *IRUN*=31 and *IHOLD*=16, 15 current steps are required for hold current reduction. A *IHOLDDELAY* setting of 4 thus results in a power down time of 4*15*2^18 clock cycles, i.e., roughly one second at 16MHz.	0	instant *IHOLD*
		1 … 15	$1*2^{18}$ … $15*2^{18}$ clocks per current decrement
GLOBAL SCALER	Allows fine control of the motor current range setting	0 … 255	scales in 1/256 steps 0=full scale

9 Velocity Based Mode Control

The TMC2160 allows the configuration of different chopper modes and modes of operation for optimum motor control. Depending on the motor load, the different modes can be optimized for lowest noise & high precision, highest dynamics, or maximum torque at highest velocity. Some of the features like CoolStep or StallGuard2 are useful in a limited velocity range. A number of velocity thresholds allow combining the different modes of operation within an application requiring a wide velocity range.

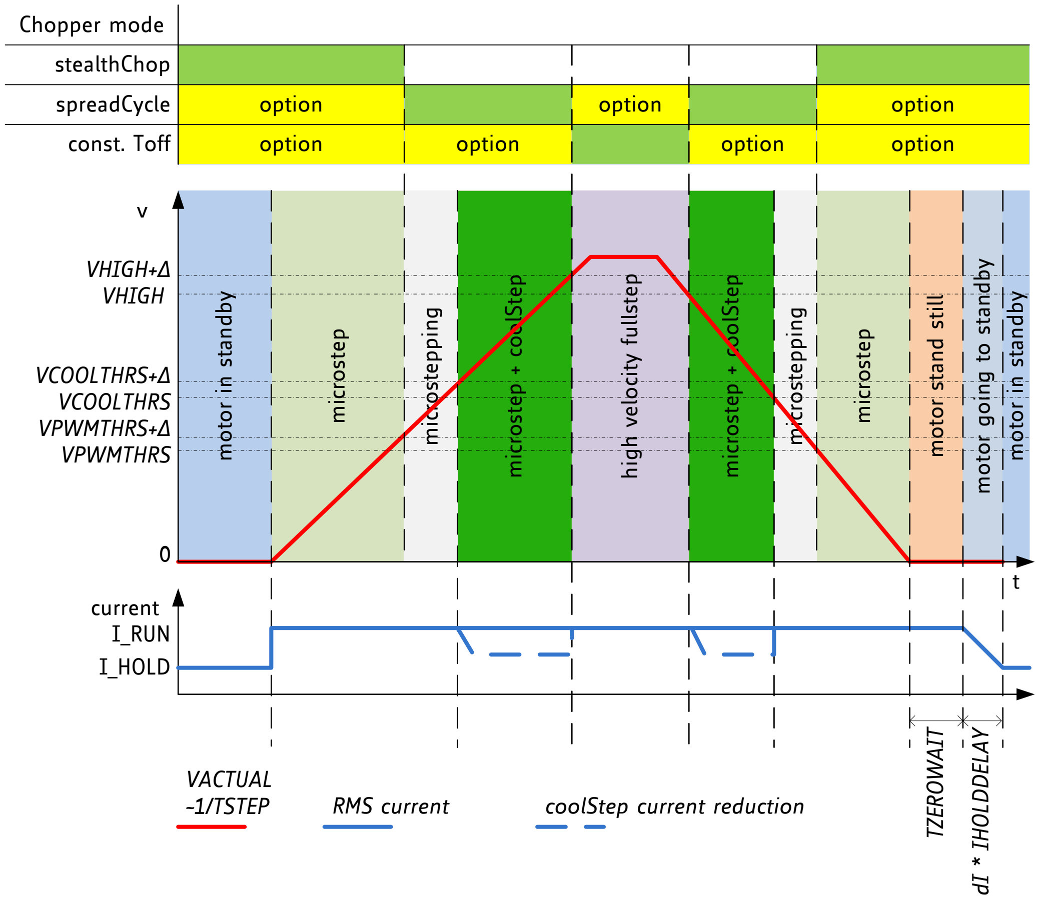


Figure 9.1 Choice of velocity dependent modes

Figure 9.1 shows all available thresholds and the required ordering. VPWMTHRS, VHIGH and VCOOLTHRS are determined by the settings *TPWMTHRS*, *THIGH* and *TCOOLTHRS.* The velocity is described by the time interval *TSTEP* between each two step pulses. This allows determination of the velocity when an external step source is used. *TSTEP* always becomes normalized to 256 microstepping. This way, the thresholds do not have to be adapted when the microstep resolution is changed. The thresholds represent the same motor velocity, independent of the microstep settings. *TSTEP* becomes compared to these threshold values. A hysteresis of 1/16 *TSTEP* resp. 1/32 *TSTEP* is applied to avoid continuous toggling of the comparison results when a jitter in the *TSTEP* measurement occurs. The upper switching velocity is higher by 1/16, resp. 1/32 of the value set as threshold. The motor current can be programmed to a run and a hold level, dependent on the standstill flag *stst*.

Using automatic velocity thresholds allows tuning the application for different velocity ranges. Features like CoolStep will integrate completely transparently in your setup. This way, once parameterized, they do not require any activation or deactivation via software.

Parameter	Description	Setting	Comment
stst	This flag indicates motor stand still in each operation mode. This occurs 2^20 clocks after the last step pulse.	0/1	Status bit, read only
TPOWER DOWN	This is the delay time after stand still (*stst*) of the motor to motor current power down. Time range is about 0 to 4 seconds. Setting 0 is no delay, 1 a minimum delay. Further increment is in discrete steps of 2^18 clock cycles.	*0…255*	Time in multiples of 2^18 t_{CLK} Set at minimum to 2 to allow automatic tuning of *PWM_OFS_AUTO*
TSTEP	Actual measured time between two 1/256 microsteps derived from the step input frequency in units of 1/fCLK. Measured value is (2^20)-1 in case of overflow or stand still.	0… 1048575	Status register, read only. Actual measured step time in multiple of t_{CLK}
TPWMTHRS	*TSTEP* ≥ *TPWMTHRS* - StealthChop PWM mode is enabled, if configured - DcStep is disabled	0… 1048575	Setting to control the upper velocity threshold for operation in StealthChop
TCOOLTHRS	*TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: - CoolStep is enabled, if configured - StealthChop voltage PWM mode is disabled *TCOOLTHRS* ≥ *TSTEP* - Stall output signal is enabled, if configured	0… 1048575	Setting to control the lower velocity threshold for operation with CoolStep and stallGuard
THIGH	*TSTEP* ≤ *THIGH*: - CoolStep is disabled (motor runs with normal current scale) - StealthChop voltage PWM mode is disabled - If *vhighchm* is set, the chopper switches to *chm*=1 with *TFD*=0 (constant off time with slow decay, only). - If *vhighfs* is set, the motor operates in fullstep mode, and the stall detection becomes switched over to DcStep stall detection.	0… 1048575	Setting to control the upper threshold for operation with CoolStep and stallGuard as well as optional high velocity step mode
small_ hysteresis	Hysteresis for step frequency comparison based on *TSTEP* (lower velocity threshold) and (*TSTEP**15/16)-1 respectively (*TSTEP**31/32)-1 (upper velocity threshold)	0	Hysteresis is 1/16
		1	Hysteresis is 1/32
vhighfs	This bit enables switching to fullstep, when *VHIGH* is exceeded. Switching takes place only at 45° position. The fullstep target current uses the current value from the microstep table at the 45° position.	0	No switch to fullstep
		1	Fullstep at high velocities
vhighchm	This bit enables switching to *chm*=1 and *fd*=0, when *VHIGH* is exceeded. This way, a higher velocity can be achieved. Can be combined with *vhighfs*=1. If set, the *TOFF* setting automatically becomes doubled during high velocity operation to avoid doubling of the chopper frequency.	0	No change of chopper mode
		1	Classic const. Toff chopper at high velocities
en_pwm_ mode	StealthChop voltage PWM enable flag (depending on velocity thresholds). Switch from off to on state while in stand still, only.	0	No StealthChop
		1	StealthChop below VPWMTHRS

10 Diagnostics and Protection

The TMC2160 supplies a complete set of diagnostic and protection capabilities, like short circuit protection and undervoltage detection. Open load detection allows testing if a motor coil connection is interrupted. See the *DRV_STATUS* table for details.

10.1 Temperature Sensors

The driver integrates a four-level temperature sensor (120°C pre-warning and selectable 136°C / 143°C / 150°C thermal shutdown) for diagnostics and for protection of the IC and the power MOSFETs and adjacent components against excess heat. Choose the overtemperature level to safely cover error conditions like missing heat convection. Heat is mainly generated by the power MOSFETs, and, at increased voltage, by the internal voltage regulators. For many applications, already the overtemperature pre-warning will indicate an abnormal operation situation and can be used to initiate user warning or power reduction measures like motor current reduction. The thermal shutdown is just an emergency measure and temperature rising to the shutdown level should be prevented by design.

After triggering the overtemperature sensor (*ot* flag), the driver remains switched off until the system temperature falls below the pre-warning level (*otpw*) to avoid continuous heating to the shutdown level.

10.2 Short Protection

The TMC2160 protects the MOSFET power stages against a short circuit or overload condition by monitoring the voltage drop in the high-side MOSFETs, as well as the voltage drop in sense resistor and low-side MOSFETs (Figure 10.1). A programmable short detection delay (*shortdelay*) allows adjusting the detector to work with very slow switching slopes. Additionally, the short detector allows filtering of the signal. This helps to prevent spurious triggering caused by effects of PCB layout, or long, adjacent motor cables (*SHORTFILTER*). All control bits are available via register *SHORT_CONF*. Additionally, the short detection is protected against single events, e.g., caused by ESD discharges, by retrying three times before switching off the motor continuously.

Parameter	**Description**	**Setting**	**Comment**
S2VS_LEVEL	Short or overcurrent detector level for lowside FETs. Checks for voltage drop in LS MOSFET and sense resistor. *Hint:* 6 to 8 recommended, down to 4 at low current scale	4…15	4 (highest sensitivity) … 15 (lowest sensitivity) *(Reset Default:* *OTP 6 or 12)*
S2G_LEVEL	*S2G_LEVEL*: Short to GND detector level for highside FETs. Checks for voltage drop on high side MOSFET. *Hint:* 6 to 14 recommended (minimum 12 if the bridge supply voltage can exceed 52V)	*2…15*	2 (highest sensitivity) … 15 (lowest sensitivity) *(Reset Default:* *OTP 6 or 12)*
SHORT_ FILTER	Spike filtering bandwidth for short detection *Hint:* A good PCB layout will allow using setting 0. Increase value if erroneous short detection occurs.	0…3	0 (lowest, 100ns), 1 (1µs) (*Reset Default*), 2 (2µs), 3 (3µs)
shortdelay	*shortdelay*: Short detection delay The short detection delay shall cover the bridge switching time. 0 will work for most applications.	0/1	0=750ns: normal, 1=1500ns: high
CHOPCONF. diss2vs	Allows to disable short to VS protection.	0/1	Leave detection enabled for normal use (0).
CHOPCONF. diss2g	Allows to disable short to GND protection.	0/1	Leave detection enabled for normal use (0).

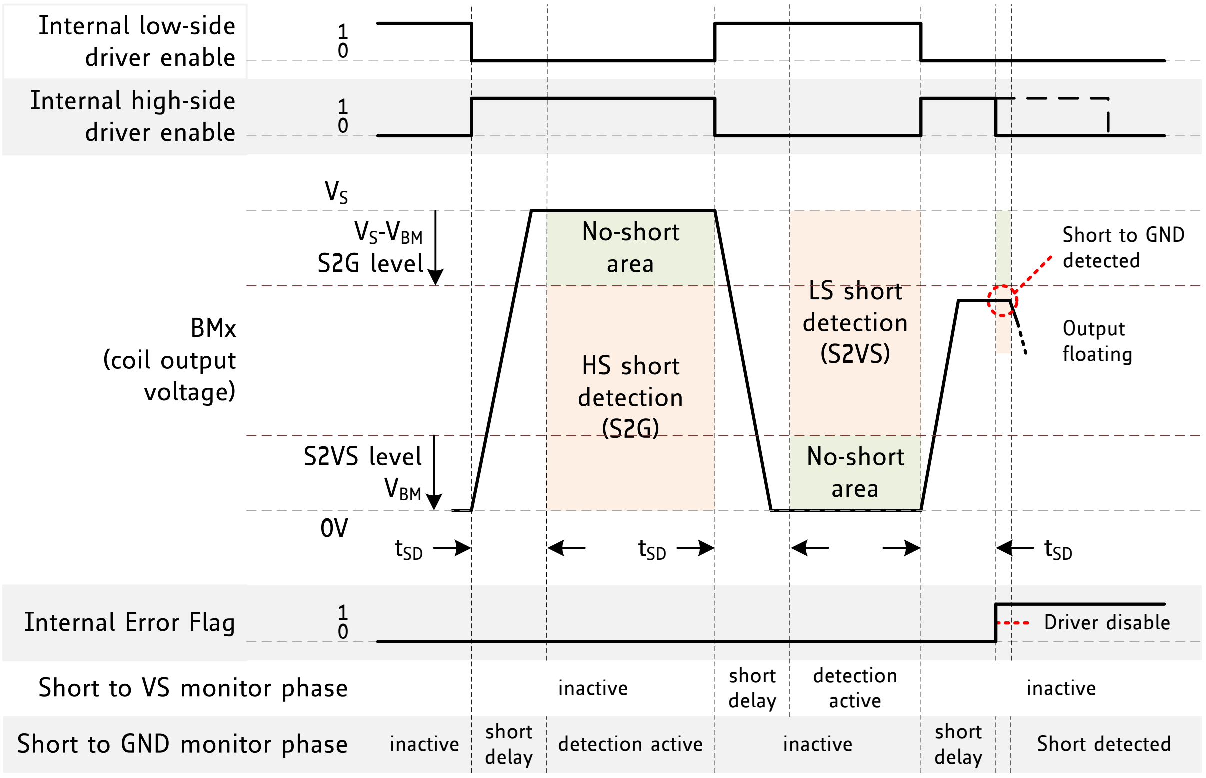


Figure 10.1 Short detection

As the low-side short detection includes the sense resistor, it can be set to a high sensitivity and provides good precision of current detection. This way, it will safely cover most overcurrent conditions, i.e., when the motor stalls, or is abruptly stopped in StealthChop mode.

> *Hint*
> Once a short condition is safely detected, the corresponding driver bridge (A or B) becomes switched off, and the *s2ga* or *s2gb* flag, respectively *s2vsa* or *s2vsb* becomes set.
> To restart the motor, disable and re-enable the driver.

> *Attention*
> Short protection cannot protect the system and the power stages for all possible short events, as a short event is rather undefined, and a complex network of external components may be involved. Therefore, short circuits should basically be avoided.

> *Hint*
> Set low-side short protection (S2VS) to sensitively detect an overcurrent condition (at 150 to 200% of nominal peak current). Especially with low resistive motors an overcurrent can easily be triggered by false settings, or motor stall when using StealthChop. Therefore, a sensitive short to VS setting will protect the power stage.

> *Attention*
> High-side short detection (S2G) sensitivity may increase at voltages above 52V. Therefore, a higher setting is required if motor supply voltage can overshoot up to 55V. We recommend a setting of 12 to 15 in this case. For fine tuning of overcurrent detection, trim the S2VS detector threshold.
> High-side short detection may falsely trigger if motor supply voltage overshoots 55V.

10.3 Open Load Diagnostics

Interrupted cables are a common cause for systems failing, e.g., when connectors are not firmly plugged. The TMC2160 detects open load conditions by checking if it can reach the desired motor coil current. This way, also undervoltage conditions, high motor velocity settings or short and overtemperature conditions may cause triggering of the open load flag, and inform the user, that motor torque may suffer. In motor stand still, open load cannot be measured, as the coils might eventually have zero current.

Open load detection is provided for system debugging.
To safely detect an interrupted coil connection, operate in SpreadCyle, and check the open load flags following a motion of minimum four times the selected microstep resolution into a single direction using low or nominal motor velocity operation, only. However, the *ola* and *olb* flags have just informative character and do not cause any action of the driver.

11 StallGuard2 Load Measurement

StallGuard2 provides an accurate measurement of the load on the motor. It can be used for stall detection as well as other uses at loads below those which stall the motor, such as CoolStep load-adaptive current reduction. The StallGuard2 measurement value changes linearly over a wide range of load, velocity, and current settings, as shown in Figure 11.1. At maximum motor load, the value goes to zero or near to zero. This corresponds to a load angle of 90° between the magnetic field of the coils and magnets in the rotor. This also is the most energy-efficient point of operation for the motor.

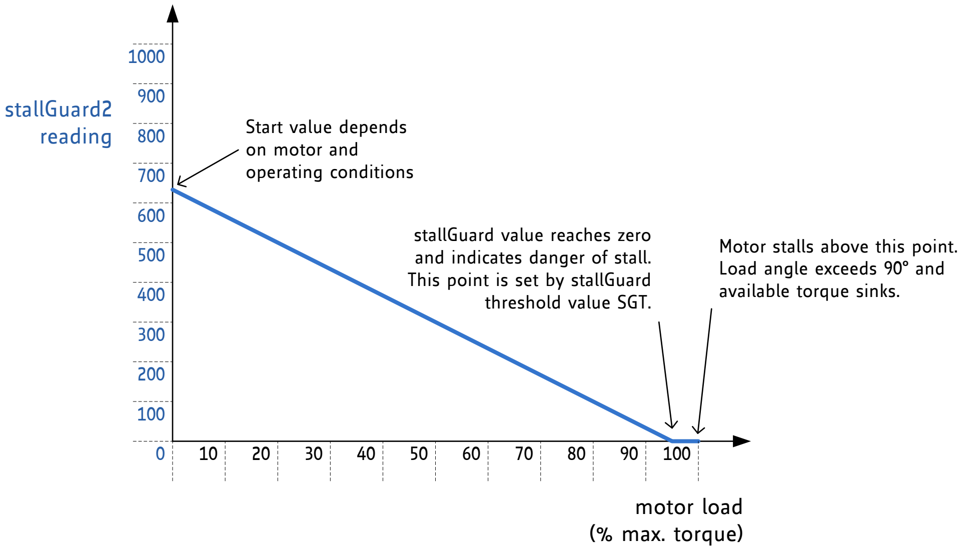


Figure 11.1 Function principle of StallGuard2

Parameter	Description	Setting	Comment
SGT	This signed value controls the StallGuard2 threshold level for stall detection and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value working with most motors. A higher value makes StallGuard2 less sensitive and requires more torque to indicate a stall.	0	indifferent value
		+1… +63	less sensitivity
		-1… -64	higher sensitivity
sfilt	Enables the StallGuard2 filter for more precision of the measurement. If set, reduces the measurement frequency to one measurement per electrical period of the motor (4 fullsteps).	0	standard mode
		1	filtered mode
Status word	**Description**	**Range**	**Comment**
SG_RESULT	This is the *StallGuard2 result*. A higher reading indicates less mechanical load. A lower reading indicates a higher load and thus a higher load angle. Tune the *SGT* setting to show a *SG_RESULT* reading of roughly 0 to 100 at maximum load before motor stall.	0… 1023	0: highest load low value: high load high value: less load

Hint
In order to use StallGuard2 and CoolStep, the StallGuard2 sensitivity should first be tuned using the SGT setting!

11.1 Tuning StallGuard2 Threshold SGT

The StallGuard2 value *SG_RESULT* is affected by motor-specific characteristics and application-specific demands on load and velocity. Therefore, the easiest way to tune the StallGuard2 threshold *SGT* for a specific motor type and operating conditions is interactive tuning in the actual application.

INITIAL PROCEDURE FOR TUNING STALLGUARD SGT

1. Operate the motor at the normal operation velocity for your application and monitor *SG_RESULT*.
2. Apply slowly increasing mechanical load to the motor. If the motor stalls before *SG_RESULT* reaches zero, decrease *SGT*. If *SG_RESULT* reaches zero before the motor stalls, increase *SGT*. A good *SGT* starting value is zero. *SGT* is signed, so it can have negative or positive values.
3. Set *TCOOLTHRS* to a value above *TSTEP* and monitor the StallGuard output signal (configure DIAG0 or DIAG1 to output stall detection). Stop the motor when a pulse is seen on the respective output. Make sure, that the motor is safely stopped whenever it is stalled. Increase *SGT* if the motor becomes stopped before a stall occurs.
4. The optimum setting is reached when *SG_RESULT* is between 0 and roughly 100 at increasing load shortly before the motor stalls, and *SG_RESULT* increases by 100 or more without load. *SGT* in most cases can be tuned for a certain motion velocity or a velocity range. Make sure, that the setting works reliable in a certain range (e.g. 80% to 120% of desired velocity) and also under extreme motor conditions (lowest and highest applicable temperature).

OPTIONAL PROCEDURE ALLOWING AUTOMATIC TUNING OF SGT

The basic idea behind the SGT setting is a factor, which compensates the StallGuard measurement for resistive losses inside the motor. At standstill and very low velocities, resistive losses are the main factor for the balance of energy in the motor, because mechanical power is zero or near to zero. This way, SGT can be set to an optimum at near zero velocity. This algorithm is especially useful for tuning SGT within the application to give the best result independent of environment conditions, motor stray, etc.

1. Operate the motor at low velocity < 10 RPM (i.e., a few to a few fullsteps per second) and target operation current and supply voltage. In this velocity range, there is not much dependence of *SG_RESULT* on the motor load, because the motor does not generate significant back EMF. Therefore, mechanical load will not make a big difference on the result.
2. Switch on *sfilt*. Now increase *SGT* starting from 0 to a value, where *SG_RESULT* starts rising. With a high *SGT*, *SG_RESULT* will rise up to the maximum value. Reduce again to the highest value, where *SG_RESULT* stays at 0. Now the *SGT* value is set as sensibly as possible. When you see *SG_RESULT* increasing at higher velocities, there will be useful stall detection.

The upper velocity for the stall detection with this setting is determined by the velocity, where the motor back EMF approaches the supply voltage, and the motor current starts dropping when further increasing velocity.

SG_RESULT goes to zero when the motor stalls and the stall output becomes activated. The external motion controller should react to a single pulse by stopping the motor, if desired. Set *TCOOLTHRS* to match the lower velocity threshold where StallGuard delivers a good result.

The power supply voltage also affects *SG_RESULT*, so tighter voltage regulation results in more accurate values. StallGuard measurement has a high resolution, and there are a few ways to enhance its accuracy, as described in the following sections.

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 18.
For detail procedure see Application Note AN002 - *Parameterization of StallGuard2 & CoolStep*

11.1.1 Variable Velocity Limits *TCOOLTHRS* and *THIGH*

The *SGT* setting chosen as a result of the previously described *SGT* tuning can be used for a certain velocity range. Outside this range, a stall may not be detected safely, and CoolStep might not give the optimum result.

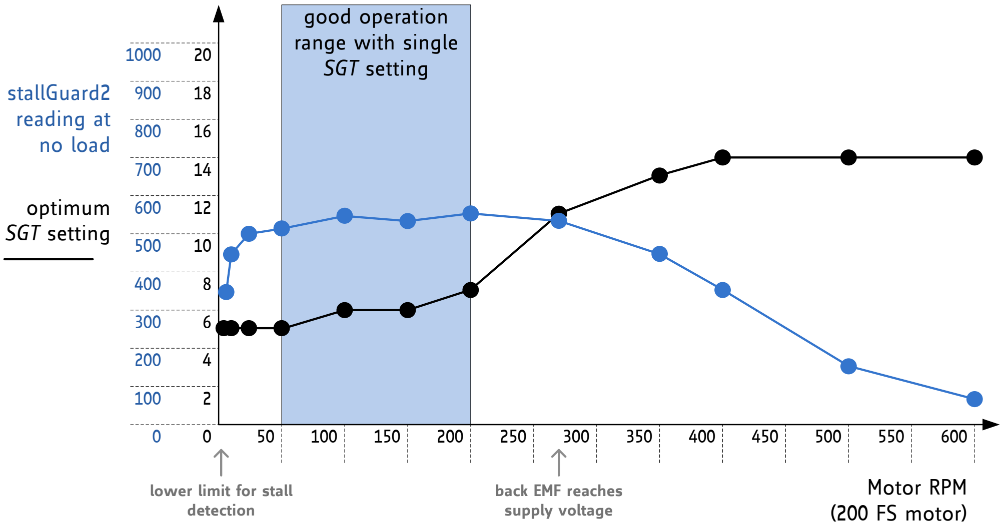


Figure 11.2 Example: optimum SGT setting and StallGuard2 reading with an example motor

In many applications, operation at or near a single operation point is used most of the time and a single setting is sufficient. The driver provides a lower and an upper velocity threshold to match this. The stall detection is disabled outside the determined operation point, e.g. during acceleration phases preceding a sensorless homing procedure when setting *TCOOLTHRS* to a matching value. An upper limit can be specified by *THIGH*.

In some applications, a velocity dependent tuning of the *SGT* value can be expedient, using a small number of support points and linear interpolation.

11.1.2 Small Motors with High Torque Ripple and Resonance

Motors with a high detent torque show an increased variation of the StallGuard2 measurement value SG with varying motor currents, especially at low currents. For these motors, the current dependency should be checked for best result.

11.1.3 Temperature Dependence of Motor Coil Resistance

Motors working over a wide temperature range may require temperature correction, because motor coil resistance increases with rising temperature. This can be corrected as a linear reduction of *SGT* at increasing temperature, as motor efficiency is reduced.

11.1.4 Accuracy and Reproducibility of StallGuard2 Measurement

In a production environment, it may be desirable to use a fixed *SGT* value within an application for one motor type. Most of the unit-to-unit variation in StallGuard2 measurements results from manufacturing tolerances in motor construction. The measurement error of StallGuard2 – provided that all other parameters remain stable – can be as low as:

$$stallGuard\ measurement\ error = \pm max(1, |SGT|)$$

11.2 StallGuard2 Update Rate and Filter

The StallGuard2 measurement value *SG_RESULT* is updated with each full step of the motor. This is enough to safely detect a stall because a stall always means the loss of four full steps. In a practical application, especially when using CoolStep, a more precise measurement might be more important than an update for each fullstep because the mechanical load never changes instantaneously from one step to the next. For these applications, the *sfilt* bit enables a filtering function over four load measurements. The filter should always be enabled when high-precision measurement is required. It compensates for variations in motor construction, for example due to misalignment of the phase A to phase B magnets. The filter should be disabled when rapid response to increasing load is required and for best results of sensorless homing using StallGuard.

11.3 Detecting a Motor Stall

For best stall detection, work without StallGuard filtering (*sfilt*=0). To safely detect a motor stall the stall threshold must be determined using a specific *SGT* setting. Therefore, the maximum load needs to be determined, which the motor can drive without stalling. At the same time, monitor the *SG_RESULT* value at this load, e.g., some value within the range 0 to 100. The stall threshold should be a value safely within the operating limits, to allow for parameter stray. The response at an *SGT* setting at or near 0 gives some idea on the quality of the signal: Check the *SG* value without load and with maximum load. They should show a difference of at least 100 or a few 100, which shall be large compared to the offset. If you set the *SGT* value in a way, that a reading of 0 occurs at maximum motor load, the stall can be automatically detected by the motion controller to issue a motor stop. In the moment of the step resulting in a step loss, the lowest reading will be visible. After the step loss, the motor will vibrate and show a higher *SG_RESULT* reading.

11.4 Homing with StallGuard

The homing of a linear drive requires moving the motor into the direction of a hard stop. As StallGuard needs a certain velocity to work (as set by *TCOOLTHRS*), make sure that the start point is far enough away from the hard stop to provide the distance required for the acceleration phase. After setting up *SGT* and the ramp generator registers, start a motion into the direction of the hard stop and activate the stop on stall function of your controller. Best results are yielded at 30% to 70% of nominal motor current and typically 1 to 5 RPS (motors smaller than NEMA17 may require higher velocities).

11.5 Limits of StallGuard2 Operation

StallGuard2 does not operate reliably at extreme motor velocities: Very low motor velocities (for many motors, less than one revolution per second) generate a low back EMF and make the measurement unstable and dependent on environment conditions (temperature, etc.). The automatic tuning procedure described above will compensate for this. Other conditions will also lead to extreme settings of *SGT* and poor response of the measurement value *SG_RESULT* to the motor load.

Very high motor velocities, in which the full sinusoidal current is not driven into the motor coils also leads to poor response. These velocities are typically characterized by the motor back EMF reaching the supply voltage.

12 CoolStep Operation

CoolStep is an automatic smart energy optimization for stepper motors based on the motor mechanical load, making them "green".

12.1 User Benefits

Energy efficiency – consumption decreased up to 75%
Motor generates less heat – improved mechanical precision
Less cooling infrastructure – for motor and driver
Cheaper motor – does the job!

CoolStep allows substantial energy savings, especially for motors which see varying loads or operate at a high duty cycle. Because a stepper motor application needs to work with a torque reserve of 30% to 50%, even a constant-load application allows significant energy savings because CoolStep automatically enables torque reserve when required. Reducing power consumption keeps the system cooler, increases motor life, and allows reducing cost in the power supply and cooling components.

Reducing motor current by half results in reducing power by a factor of four.

12.2 Setting up for CoolStep

CoolStep is controlled by several parameters, but two are critical for understanding how it works:

Parameter	Description	Range	Comment
SEMIN	4-bit unsigned integer that sets a *lower threshold*. If *SG* goes below this threshold, CoolStep increases the current to both coils. The 4-bit *SEMIN* value is scaled by 32 to cover the lower half of the range of the 10-bit *SG* value. (The name of this parameter is derived from smartEnergy, which is an earlier name for CoolStep.)	0	disable CoolStep
		1…15	threshold is *SEMIN**32
SEMAX	4-bit unsigned integer that controls an *upper threshold*. If SG is sampled equal to or above this threshold enough times, CoolStep decreases the current to both coils. The upper threshold is (*SEMIN* + *SEMAX* + 1)*32.	0…15	threshold is (*SEMIN*+*SEMAX*+1)*32

Figure 12.1 shows the operating regions of CoolStep:

- The black line represents the *SG* measurement value.
- The blue line represents the mechanical load applied to the motor.
- The red line represents the current into the motor coils.

When the load increases, *SG_RESULT* falls below *SEMIN*, and CoolStep increases the current. When the load decreases, *SG_RESULT* rises above (*SEMIN* + *SEMAX* + 1) * 32, and the current is reduced.

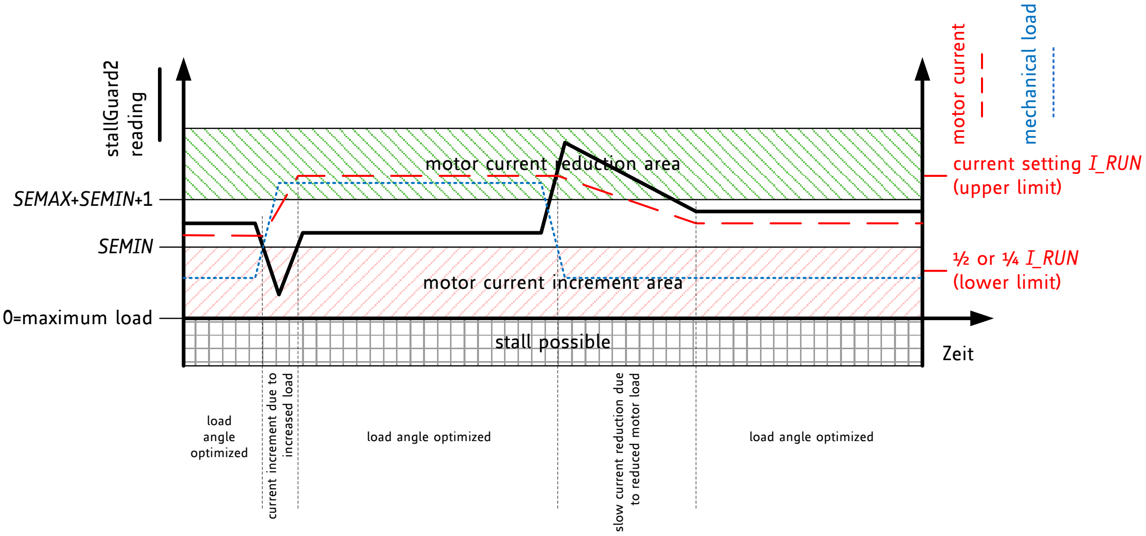


Figure 12.1 CoolStep adapts motor current to the load

Five more parameters control CoolStep and one status value is returned:

Parameter	**Description**	**Range**	**Comment**
SEUP	Sets the *current increment step*. The current becomes incremented for each measured StallGuard2 value below the lower threshold.	0…3	step width is 1, 2, 4, 8
SEDN	Sets the number of StallGuard2 readings above the upper threshold necessary for each *current decrement* of the motor current.	0…3	number of StallGuard2 measurements per decrement: 32, 8, 2, 1
SEIMIN	Sets the *lower motor current limit* for CoolStep operation by scaling the *IRUN* current setting.	0 1	0: 1/2 of IRUN 1: 1/4 of IRUN
TCOOL THRS	Lower velocity threshold for switching on CoolStep and stall output. Below this velocity CoolStep becomes disabled (not used in STEP/DIR mode). Adapt to the lower limit of the velocity range where StallGuard2 gives a stable result. *Hint:* May be adapted to disable CoolStep during acceleration and deceleration phase by setting identical to *VMAX*.	1… 2^20-1	Specifies lower CoolStep velocity by comparing the threshold value to *TSTEP*
THIGH	Upper velocity threshold value for CoolStep and stall output signal. Above this velocity CoolStep becomes disabled. Adapt to the velocity range where StallGuard2 gives a stable result.	1… 2^20-1	Also controls additional functions like switching to fullstepping.
Status word	**Description**	**Range**	**Comment**
CSACTUAL	This status value provides the *actual motor current scale* as controlled by CoolStep. The value goes up to the *IRUN* value and down to the portion of *IRUN* as specified by *SEIMIN*.	0…31	1/32, 2/32, … 32/32

12.3 Tuning CoolStep

Before tuning CoolStep, first tune the StallGuard2 threshold level *SGT*, which affects the range of the load measurement value *SG_RESULT*. CoolStep uses *SG_RESULT* to operate the motor near the optimum load angle of +90°.

The current increment speed is specified in *SEUP*, and the current decrement speed is specified in *SEDN*. They can be tuned separately because they are triggered by different events that may need different responses. The encodings for these parameters allow the coil currents to be increased much more quickly than decreased, because crossing the lower threshold is a more serious event that may require a faster response. If the response is too slow, the motor may stall. In contrast, a slow response to crossing the upper threshold does not risk anything more serious than missing an opportunity to save power.

CoolStep operates between limits controlled by the current scale parameter *IRUN* and the *seimin* bit.

12.3.1 Response Time

For fast response to increasing motor load, use a high current increment step *SEUP*. If the motor load changes slowly, a lower current increment step can be used to avoid motor oscillations. If the filter controlled by *sfilt* is enabled, the measurement rate and regulation speed are cut by a factor of four.

Hint

The most common and most beneficial use is to adapt CoolStep for operation at the typical system target operation velocity and to set the velocity thresholds according. As acceleration and decelerations normally shall be quick, they will require the full motor current, while they have only a small contribution to overall power consumption due to their short duration.

12.3.2 Low Velocity and Standby Operation

Because CoolStep is not able to measure the motor load in standstill and at very low RPM, a lower velocity threshold is provided for enabling CoolStep. It should be set to an application specific default value. Below this threshold the normal current setting via *IRUN* respectively *IHOLD* is valid. An upper threshold is provided by the *VHIGH* setting. Both thresholds can be set as a result of the StallGuard2 tuning process.

13 STEP/DIR Interface

The STEP and DIR inputs provide a simple, standard interface compatible with many existing motion controllers. The MicroPlyer STEP pulse interpolator brings the smooth motor operation of high-resolution microstepping to applications originally designed for coarser stepping. In case an external step source is used, the complete integrated motion controller can be switched off.

13.1 Timing

Figure 13.1 shows the timing parameters for the STEP and DIR signals, and the table below gives their specifications. When the *dedge* mode bit in the *CHOPCONF* register is set, both edges of STEP are active. If *dedge* is cleared, only rising edges are active. STEP and DIR are sampled and synchronized to the system clock. An internal analog filter removes glitches on the signals, such as those caused by long PCB traces. If the signal source is far from the chip, and especially if the signals are carried on cables, the signals should be filtered or differentially transmitted.

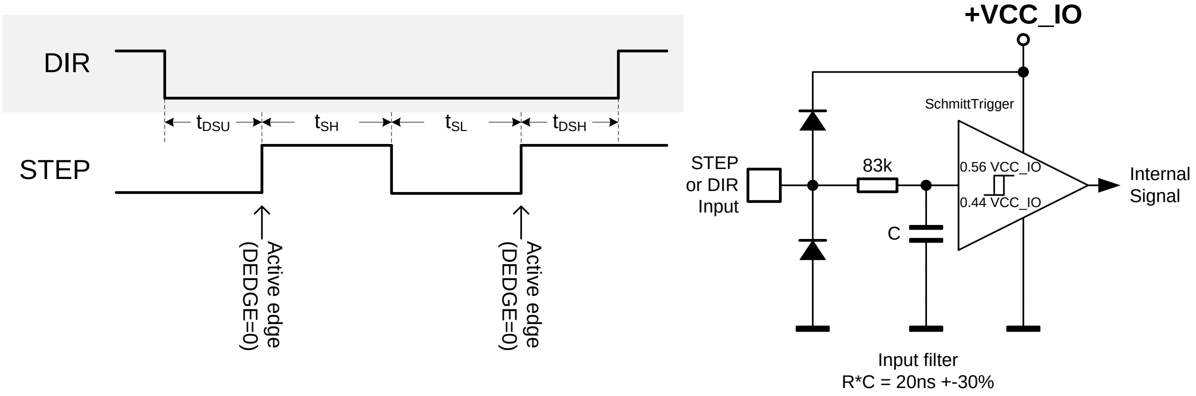


Figure 13.1 STEP and DIR timing, Input pin filter

STEP and DIR interface timing	**AC-Characteristics** clock period is t_{CLK}					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
step frequency (at maximum microstep resolution)	f_{STEP}	*dedge*=0			½ f_{CLK}	
		dedge=1			¼ f_{CLK}	
fullstep frequency	f_{FS}				f_{CLK}/512	
STEP input low time *)	t_{SL}		max(t_{FILTSD}, t_{CLK}+20)	100		ns
STEP input high time *)	t_{SH}		max(t_{FILTSD}, t_{CLK}+20)	100		ns
DIR to STEP setup time	t_{DSU}		20			ns
DIR after STEP hold time	t_{DSH}		20			ns
STEP and DIR spike filtering time *)	t_{FILTSD}	rising and falling edge	13	20	30	ns
STEP and DIR sampling relative to rising CLK input	$t_{SDCLKHI}$	before rising edge of CLK input		t_{FILTSD}		ns

*) These values are valid with full input logic level swing, only. Asymmetric logic levels will increase filtering delay t_{FILTSD}, due to an internal input RC filter.

13.2 Changing Resolution

The TMC2160 includes an internal microstep table with 1024 sine wave entries to generate sinusoidal motor coil currents. These 1024 entries correspond to one electrical revolution or four fullsteps. The microstep resolution setting determines the step width taken within the table. Depending on the DIR input, the microstep counter is increased (DIR=0) or decreased (DIR=1) with each STEP pulse by the step width. The microstep resolution determines the increment respectively the decrement. At maximum resolution, the sequencer advances one step for each step pulse. At half resolution, it advances two steps. Increment is up to 256 steps for fullstepping. The sequencer has special provision to allow seamless switching between different microstep rates at any time. When switching to a lower microstep resolution, it calculates the nearest step within the target resolution and reads the current vector at that position. This behavior especially is important for low resolutions like fullstep and halfstep because any failure in the step sequence would lead to asymmetrical run when comparing a motor running clockwise and counterclockwise.

EXAMPLES:

Fullstep:	Cycles through table positions: 128, 384, 640 and 896 (45°, 135°, 225° and 315° electrical position, both coils on at identical current). The coil current in each position corresponds to the RMS-Value (0.71 * amplitude). Step size is 256 (90° electrical)
Half step:	The first table position is 64 (22.5° electrical), Step size is 128 (45° steps)
Quarter step:	The first table position is 32 (90°/8=11.25° electrical), Step size is 64 (22.5° steps)

This way equidistant steps result, and they are identical in both rotation directions. Some older drivers also use zero current (table entry 0, 0°) as well as full current (90°) within the step tables. This kind of stepping is avoided because it provides less torque and has a worse power dissipation in driver and motor.

Step position	table position	current coil A	current coil B
Half step 0	64	38.3%	92.4%
Full step 0	128	70.7%	70.7%
Half step 1	192	92.4%	38.3%
Half step 2	320	92.4%	-38.3%
Full step 1	384	70.7%	-70.7%
Half step 3	448	38.3%	-92.4%
Half step 4	576	-38.3%	-92.4%
Full step 2	640	-70.7%	-70.7%
Half step 5	704	-92.4%	-38.3%
Half step 6	832	-92.4%	38.3%
Full step 3	896	-70.7%	70.7%
Half step 7	960	-38.3%	92.4%

13.3 MicroPlyer and Stand Still Detection

For each active edge on STEP, MicroPlyer produces microsteps at 256x resolution, as shown in Figure 13.2. It interpolates the time in between of two step impulses at the step input based on the last step interval. This way, from 2 microsteps (128 microstep to 256 microstep interpolation) up to 256 microsteps (full step input to 256 microsteps) are driven for a single step pulse.

Enable MicroPlyer by setting the *intpol* bit in the *CHOPCONF* register.
GCONF.faststandstill allows reduction of standstill detection time to 2^18 clocks (~20ms)

The step rate for the interpolated 2 to 256 microsteps is determined by measuring the time interval of the previous step period and dividing it into up to 256 equal parts. The maximum time between two microsteps corresponds to 2^{20} (roughly one million system clock cycles), for an even distribution of 256 microsteps. At 12 MHz system clock frequency, this results in a minimum step input frequency of 12 Hz for MicroPlyer operation (50 Hz with *faststandstill* = 1). A lower step rate causes the *STST* bit to be set, which indicates a standstill event. At that frequency, microsteps occur at a rate of (system clock frequency)/2^{16} ~ 256 Hz. When a stand still is detected, the driver automatically switches the motor to holding current *IHOLD*.

Hint
MicroPlyer only works perfectly with a stable STEP frequency. Do not use the *dedge* option if the STEP signal does not have a 50% duty cycle.

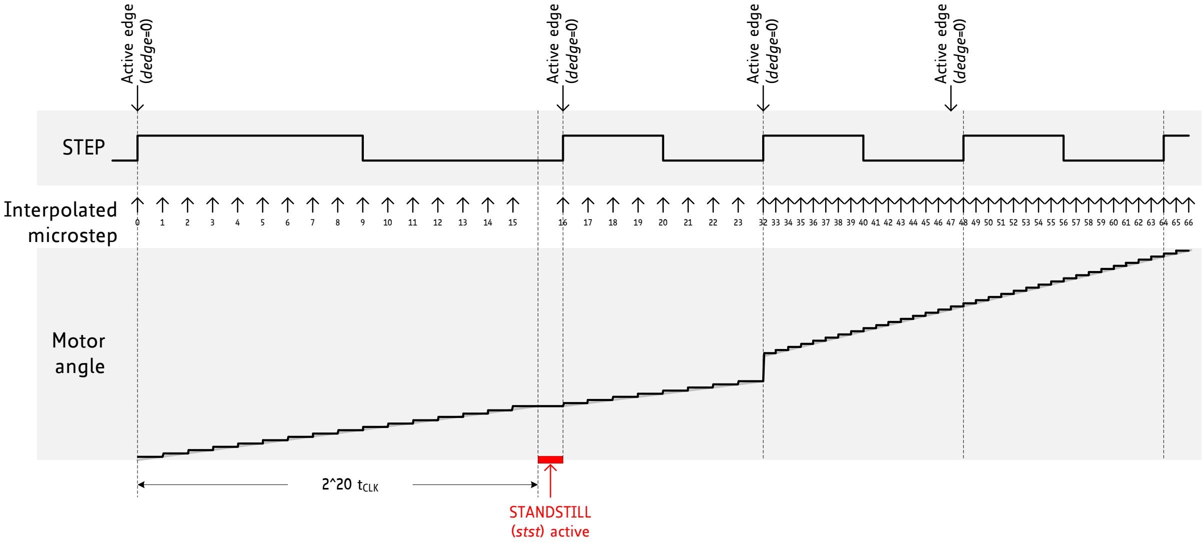


Figure 13.2 MicroPlyer microstep interpolation with rising STEP frequency (Example: 16 to 256)

In Figure 13.2, the first STEP cycle is long enough to set the standstill bit *stst*. This bit is cleared on the next STEP active edge. Then, the external STEP frequency increases. After one cycle at the higher rate MicroPlyer adapts the interpolated microstep rate to the higher frequency. During the last cycle at the slower rate, MicroPlyer did not generate all 16 microsteps, so there is a small jump in motor angle between the first and second cycles at the higher rate. With the flag *GCONF.faststandstill* enabled, standstill detection is after 2^18 clocks (rather than 2^20 clocks) without step pulse. This allows faster current reduction for energy saving in drives with short stand still times.

14 DIAG Outputs

Operation with an external motion controller often requires quick reaction to certain states of the stepper motor driver. Therefore, the DIAG outputs supply a configurable set of different real time information complementing the STEP/DIR interface.

Both, the information available at DIAG0 and DIAG1 can be selected as well as the type of output (low active open drain – default setting, or high active push-pull). In order to determine a reset of the driver, DIAG0 always shows a power-on reset condition by pulling low during a reset condition. Figure 14.1 shows the available signals and control bits.

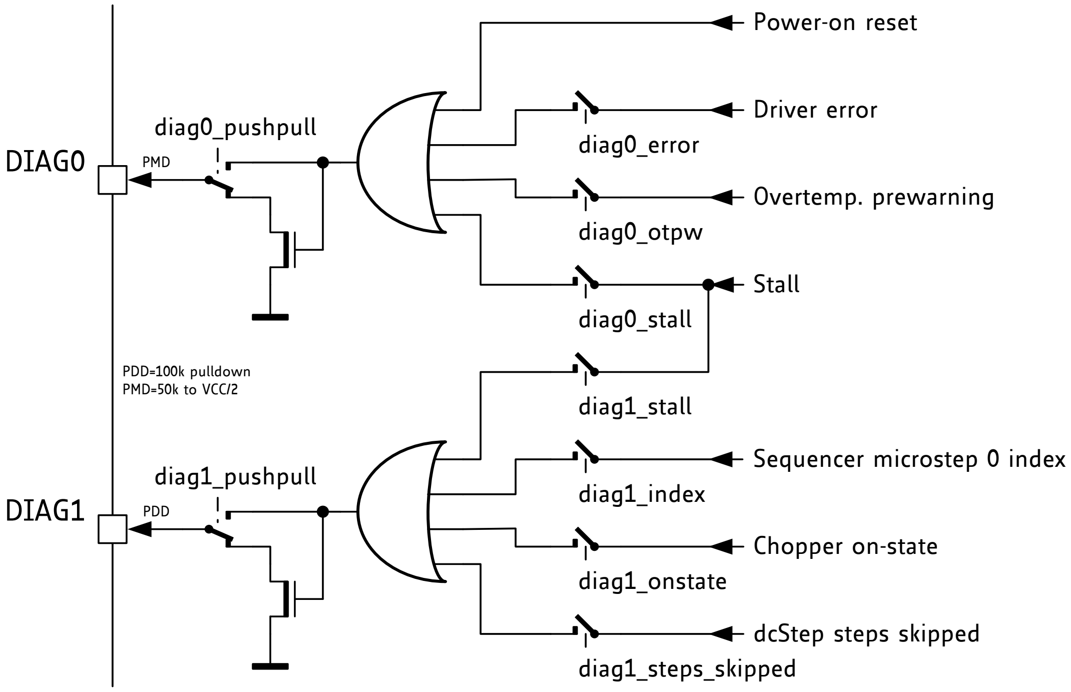


Figure 14.1 DIAG outputs

The stall output signal allows StallGuard2 to be handled by the external motion controller like a stop switch. The index output signals the microstep counter zero position, to allow the application to reference the drive to a certain current pattern. Chopper on-state shows the on-state of both coil choppers (alternating) when working in SpreadCycle or constant off time to determine the duty cycle. The DcStep skipped information is an alternative way to find out when DcStep runs with a velocity below the step velocity. It toggles with each step not taken by the sequencer.

Attention
The duration of the index pulse corresponds to the duration of the microstep. When working without interpolation at less than 256 microsteps, the index time goes down to two CLK clock cycles.

15 DcStep

DcStep is an automatic commutation mode for the stepper motor. It allows the stepper to run with its target velocity as commanded by the Step signal, as long as it can cope with the load. In case the motor becomes overloaded, it slows down to a velocity, where the motor can still drive the load. This way, the stepper motor never stalls and can drive heavy loads as fast as possible. Its higher torque available at lower velocity, plus dynamic torque from its flywheel mass allows compensating for mechanical torque peaks. In case the motor becomes completely blocked, the stall flag becomes set.

15.1 User Benefits

Motor	–	never loses steps
Application	–	works as fast as possible
Acceleration	–	automatically as high as possible
Energy efficiency	–	highest at speed limit
Cheaper motor	–	does the job!

15.2 Designing-In DcStep

In a classical application, the operation area is limited by the maximum torque required at maximum application velocity. A safety margin of up to 50% torque is required, to compensate for unforeseen load peaks, torque loss due to resonance and aging of mechanical components. DcStep allows using up to the full available motor torque. Even higher short time dynamic loads can be overcome using motor and application flywheel mass without the danger of a motor stall. With DcStep the nominal application load can be extended to a higher torque only limited by the safety margin near the holding torque area (which is the highest torque the motor can provide). Additionally, maximum application velocity can be increased up to the actually reachable motor velocity.

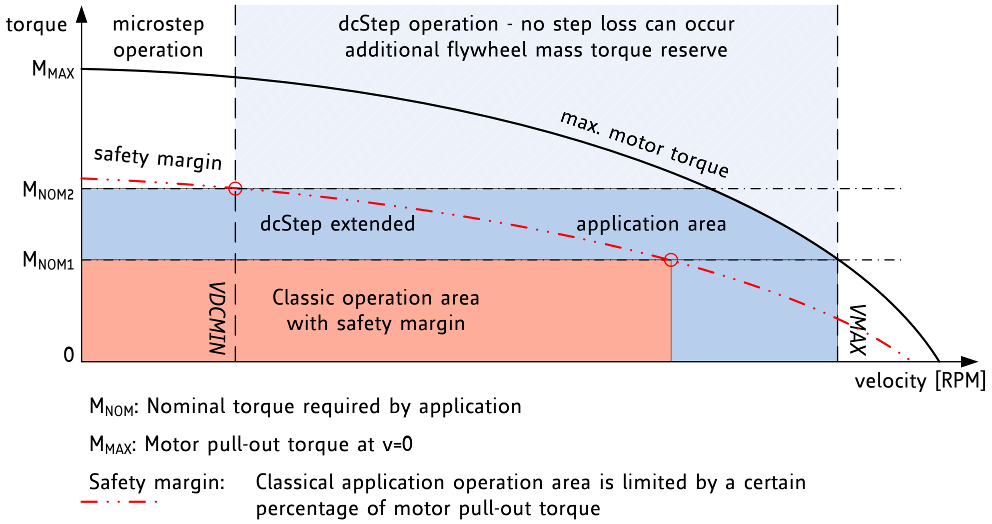


Figure 15.1 DcStep extended application operation area

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 18.
For detail configuration procedure see Application Note AN003 - *DcStep*

15.3 Stall Detection in DcStep Mode

While DcStep is able to decelerate the motor upon overload, it cannot avoid a stall in every operation situation. Once the motor is blocked, or it becomes decelerated below a motor dependent minimum velocity where the motor operation cannot safely be detected any more, the motor may stall and loose steps. A StallGuard2 load value also is available during DcStep operation. The range of values is limited to 0 to 255, in certain situations up to 511 will be read out. To enable stallGuard, also set *TCOOLTHRS* corresponding to a velocity slightly above *VDCMIN* or up to *VMAX*.

Stall detection in this mode may trigger falsely due to resonances when flywheel loads are loosely coupled to the motor axis.

Parameter	Description	Range	Comment
vhighfs *&* *vhighchm*	These chopper configuration flags in *CHOPCONF* need to be set for DcStep operation. As soon as *VDCMIN* becomes exceeded, the chopper becomes switched to fullstepping.	0 / 1	set to 1 for DcStep
TOFF	DcStep often benefits from an increased off time value in *CHOPCONF*. Settings >2 should be preferred.	2… 15	Settings 8…15 do not make any difference to setting 8 for DcStep operation.
VDCMIN	This is the lower threshold for DcStep operation when using internal ramp generator. Below this threshold, the motor operates in normal microstep mode. In DcStep operation, the motor operates at minimum *VDCMIN*, even when it is completely blocked. Tune together with *DC_TIME* setting. Activation of StealthChop also disables DcStep.	0… 2^22	0: Disable DcStep Set to the lower velocity limit for DcStep operation.
DC_TIME	This setting controls the reference pulse width for DcStep load measurement. It must be optimized for robust operation with maximum motor torque. A higher value allows higher torque and higher velocity, a lower value allows operation down to a lower velocity as set by *VDCMIN*. Check best setting under nominal operation conditions, and re-check under extreme operating conditions (e.g., lowest operation supply voltage, highest motor temperature, and highest supply voltage, lowest motor temperature).	0… 1023	Lower limit for the setting is: t_{BLANK} (as defined by *TBL*) in clock cycles + *n* with *n* in the range 1 to 100 (for a typical motor)
DC_SG	This setting controls stall detection in DcStep mode. Increase for higher sensitivity. A stall can be used as an error condition by issuing a hard stop for the motor. Stop the motor upon an impulse on the stall output (configure DIAG0 or DIAG1 to signal a stall). This way the motor will be stopped once it stalls.	0… 255	Set slightly higher than *DC_TIME* / 16

15.4 DcStep with STEP/DIR Interface

The TMC2160 provides two ways to use DcStep when interfaced to an external motion controller. The first way gives direct control of the DcStep step execution to the external motion controller, which must react to motor overload and is allowed to override a blocked motor situation. The second way assumes that the external motion controller cannot directly react to DcStep signals. The TMC2160 automatically reduces the motor velocity or stops the motor upon overload. To allow the motion controller to react to the reduced real motor velocity in this mode, the counter *LOST_STEPS* gives the number of steps which have been commanded, but not taken by the motor controller. The motion controller can later on read out *LOST_STEPS* and drive any missing number of steps. In case of a blocked motor, it tries moving it with the minimum velocity as programmed by *VDCMIN*.

Enabling DcStep automatically sets the chopper to constant TOFF mode with slow decay only. This way, no re-configuration is required when switching from microstepping mode to DcStep and back.

DcStep operation is controlled by three pins in STEP and DIR mode:

- DCEN – Forces the driver to DcStep operation if high. A velocity-based activation of DcStep is controlled by *TPWMTHRS* when using StealthChop operation for low velocity settings. In this case, DcStep is disabled while in StealthChop mode, i.e. at velocities below the StealthChop switching velocity.
- DCO – Informs the motion controller when motor is not ready to take a new step (low level). The motion controller shall react by delaying the next step until DCO becomes high. The sequencer can buffer up to the effective number of microsteps per fullstep to allow the motion controller to react to assertion of DCO. In case the motor is blocked this wait-situation can be terminated after a timeout by providing a long > 1024 clock STEP input, or via the internal *VDCMIN* setting.
- DCIN – Commands the driver to wait with step execution and to disable DCO. This input can be used for synchronization of multiple drivers operating with DcStep.

15.4.1 Using *LOST_STEPS* for DcStep Operation

This is the simplest possibility to integrate DcStep with an external motion controller: The external motion controller enables DcStep using DCEN or the internal velocity threshold. The TMC2160 tries to follow the steps. In case it needs to slow down the motor, it counts the difference between incoming steps on the STEP signal and steps going to the motor. The motion controller can read out the difference and compensate for the difference after the motion or on a cyclic basis. Figure 15.2 shows the principle (simplified).

In case the motor driver needs to postpone steps due to detection of a mechanical overload in DcStep, and the motion controller does not react to this by pausing the step generation, *LOST_STEPS* becomes incremented or decremented (depending on the direction set by DIR) with each step which is not taken. This way, the number of lost steps can be read out and executed later on or be appended to the motion. As the driver needs to slow down the motor while the overload situation persists, the application will benefit from a high microstepping resolution, because it allows more seamless acceleration or deceleration in DcStep operation. In case the application is completely blocked, *VDCMIN* sets a lower limit to the step execution. If the motor velocity falls below this limit, however an unknown number of steps is lost, and the motor position is not exactly known any more. DCIN allows for step synchronization of two drivers: it stops the execution of steps if low and sets DCO low.

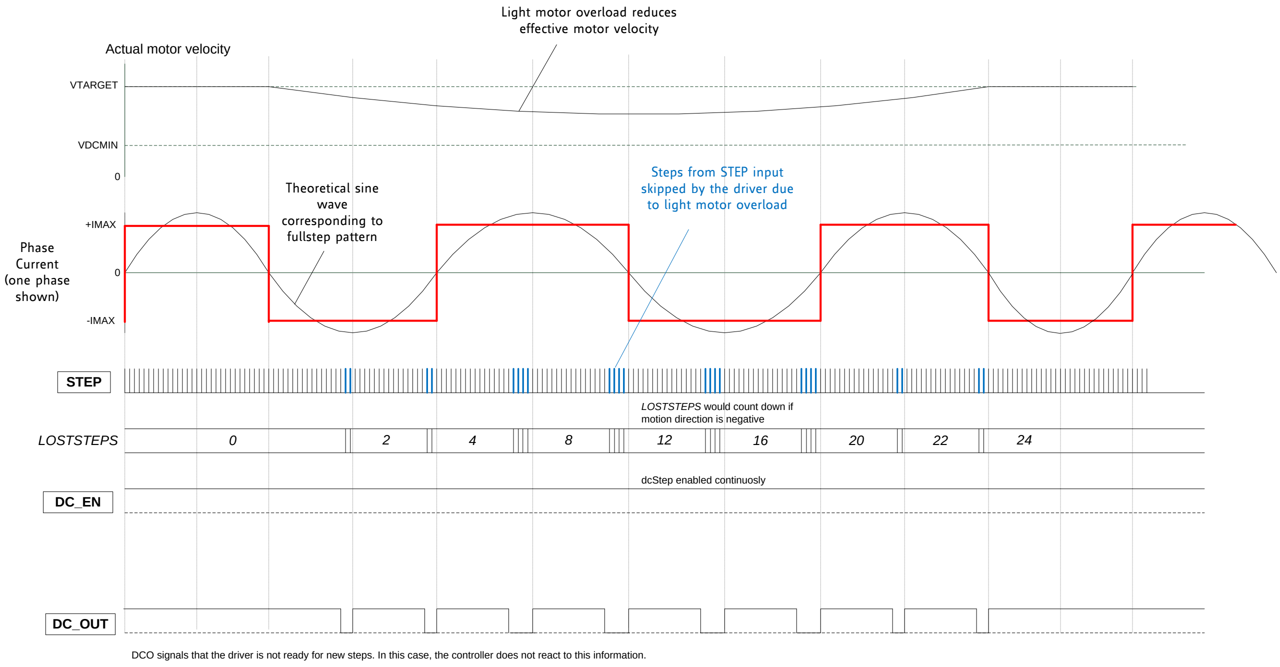


Figure 15.2 Motor moving slower than STEP input due to light overload. LOSTSTEPS incremented

15.4.2 DCO Interface to Motion Controller

In STEP/DIR mode, DCEN enables DcStep. It is up to the external motion controller to enable DcStep either, once a minimum step velocity is exceeded within the motion ramp, or to use the automatic threshold *VDCMIN* for DcStep enable.

The STEP/DIR interface works in microstep resolution, even if the internal step execution is based on fullstep. This way, no switching to a different mode of operation is required within the motion controller. The DcStep output DCO signals if the motor is ready for the next step based on the DcStep measurement of the motor. If the motor has not yet mechanically taken the last step, this step cannot be executed, and the driver stops automatically before execution of the next fullstep. This situation is signaled by DCO. The external motion controller shall stop step generation if DCOUT is low and wait until it becomes high again. Figure 15.4 shows this principle. The driver buffers steps during the waiting period up to the number of microstep setting minus one. In case, DCOUT does not go high within the lower step limit time e.g., due to a severe motor overload, a step can be enforced: override the stop status by a long STEP pulse with min. 1024 system clocks length. When using internal clock, a pulse length of minimum 125µs is recommended.

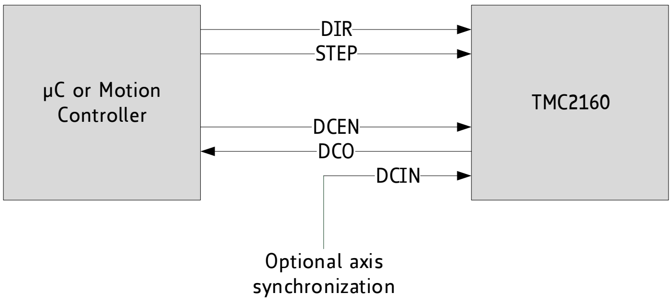
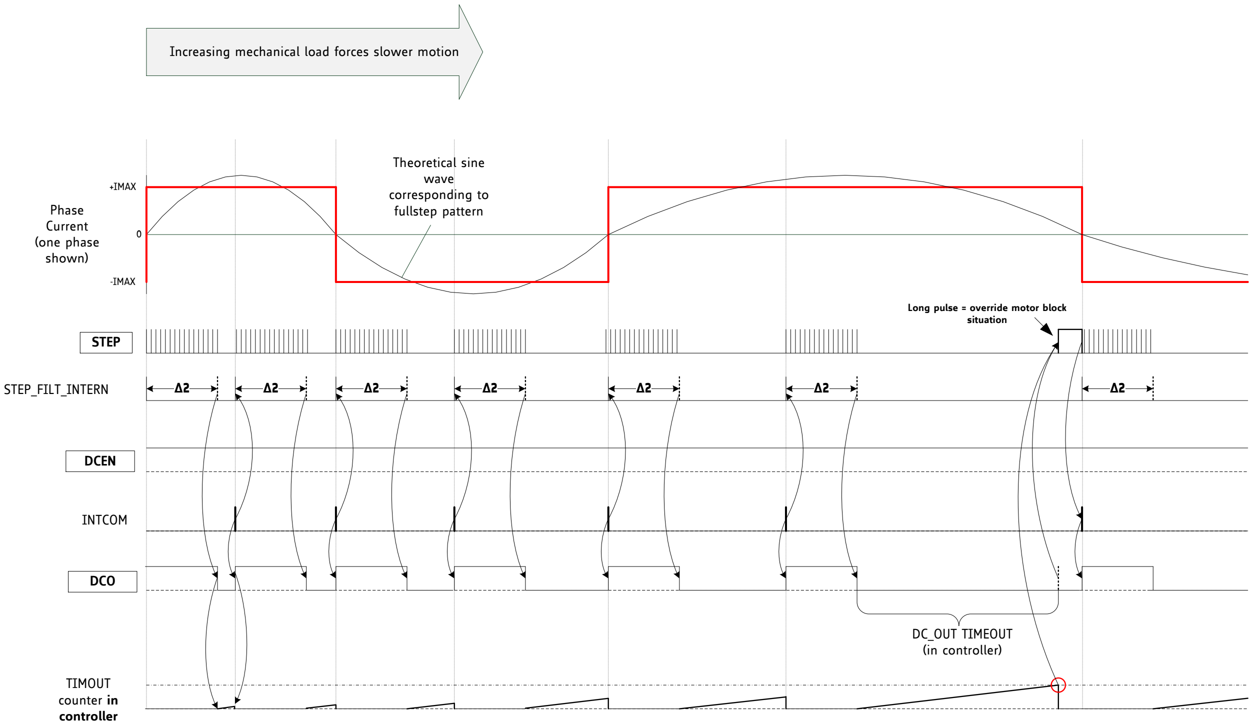


Figure 15.3 Full signal interconnection for DcStep



Δ2 = MRES (number of microsteps per fullstep)

Figure 15.4 DCO Interface to motion controller – step generator stops when DCO is asserted

16 Sine-Wave Look-up Table

The TMC2160 driver provides a programmable look-up table for storing the microstep current wave. As a default, the table is pre-programmed with a sine wave, which is a good starting point for most stepper motors. Reprogramming the table to a motor specific wave allows drastically improved microstepping especially with low-cost motors.

16.1 User Benefits

Microstepping – extremely improved with low-cost motors
Motor – runs smooth and quiet
Torque – reduced mechanical resonances yields improved torque

16.2 Microstep Table

In order to minimize required memory and the amount of data to be programmed, only a quarter of the wave becomes stored. The internal microstep table maps the microstep wave from 0° to 90°. It becomes symmetrically extended to 360°. When reading out the table the 10-bit microstep counter *MSCNT* addresses the fully extended wave table. The table is stored in an incremental fashion, using each one bit per entry. Therefore only 256 bits (*ofs00* to *ofs255*) are required to store the quarter wave. These bits are mapped to eight 32-bit registers. Each *ofs* bit controls the addition of an inclination *Wx* or *Wx*+1 when advancing one step in the table. When *Wx* is 0, a 1 bit in the table at the actual microstep position means "add one" when advancing to the next microstep. As the wave can have a higher inclination than 1, the base inclinations *Wx* can be programmed to -1, 0, 1, or 2 using up to four flexible programmable segments within the quarter wave. This way even negative inclination can be realized. The four inclination segments are controlled by the position registers *X1* to *X3*. Inclination segment 0 goes from microstep position 0 to *X1*-1 and its base inclination is controlled by *W0*, segment 1 goes from *X1* to *X2*-1 with its base inclination controlled by *W1*, etc.

When modifying the wave, care must be taken to ensure a smooth and symmetrical zero transition when the quarter wave becomes expanded to a full wave. The maximum resulting swing of the wave should be adjusted to a range of -248 to 248, to give the best possible resolution while leaving headroom for the hysteresis-based chopper to add an offset.

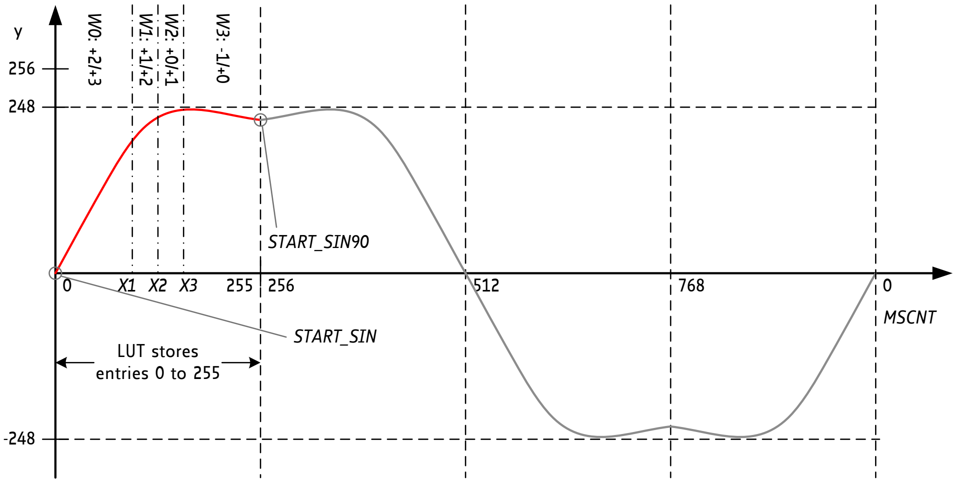


Figure 16.1 LUT programming example

When the microstep sequencer advances within the table, it calculates the actual current values for the motor coils with each microstep and stores them to the registers *CUR_A* and *CUR_B*. However, the incremental coding requires an absolute initialization, especially when the microstep table becomes modified. Therefore *CUR_A* and *CUR_B* become initialized whenever *MSCNT* passes zero.

Two registers control the starting values of the tables:

- As the starting value at zero is not necessarily 0 (it might be 1 or 2), it can be programmed into the starting point register *START_SIN*.
- In the same way, the start of the second wave for the second motor coil needs to be stored in *START_SIN90*. This register stores the resulting table entry for a phase shift of 90° for a 2-phase motor.

> *Hint*
> Refer chapter 5.3 for the register set and for the default table function stored in the drivers. The default table is a good base for realizing an own table.
> The TMC2160-EVAL comes with a calculation tool for own waves.

Initialization example for the default microstep table:

MSLUT[0]= %10101010101010101011010101010100 = 0xAAAAB554
MSLUT[1]= %01001010100101010101010010101010 = 0x4A9554AA
MSLUT[2]= %00100100010010010010100100101001 = 0x24492929
MSLUT[3]= %00010000000100000100001000100010 = 0x10104222
MSLUT[4]= %11111011111111111111111111111111 = 0xFBFFFFFF
MSLUT[5]= %10110101101110110111011101111101 = 0xB5BB777D
MSLUT[6]= %01001001001010010101010101010110 = 0x49295556
MSLUT[7]= %00000000010000000100001000100010 = 0x00404222

MSLUTSEL= 0xFFFF8056:
X1=128, *X2*=255, *X3*=255
W3=%01, *W2*=%01, *W1*=%01, *W0*=%10

MSLUTSTART= 0x00F70000:
START_SIN_0= 0, *START_SIN90*= 247

17 Emergency Stop

The driver provides a negative active enable pin DRV_ENN to safely switch off all power MOSFETs. This allows putting the motor into freewheeling. Further, it is a safe hardware function whenever an emergency-stop not coupled to software is required. Some applications may require the driver to be put into a state with active holding current or with a passive braking mode. This is possible by programming the pin DCIN to act as a step disable function. Set GCONF flag *stop_enable* to activate this option. Whenever DCIN becomes pulled up, the motor will stop abruptly and go to the power down state, as configured via *IHOLD*, *IHOLDDELAY* and StealthChop standstill options. Disabling the driver via DRV_ENN will require three clock cycles to safely switch off the driver.

18 Quick Configuration Guide

This guide is meant as a practical tool to come to a first configuration and do a minimum set of measurements and decisions for tuning the driver. It does not cover all advanced functionalities but concentrates on the basic function set to make a motor run smoothly. Once the motor runs, you may decide to explore additional features, e.g., freewheeling, and further functionality in more detail. A current probe on one motor coil is a good aid to find the best settings, but it is not a must.

CURRENT SETTING AND FIRST STEPS WITH STEALTHCHOP

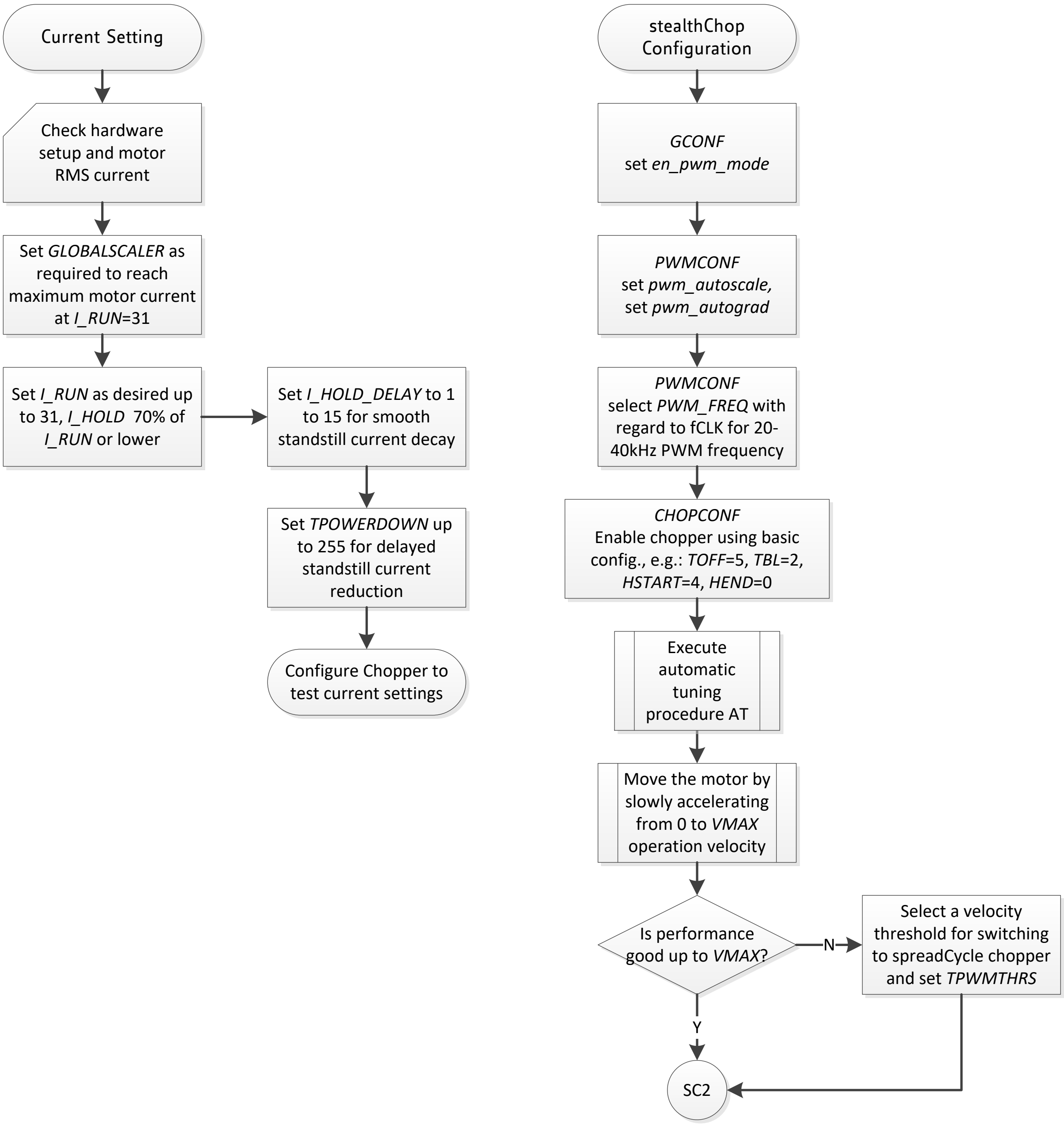


Figure 18.1 Current setting and first steps with StealthChop

TUNING STEALTHCHOP AND SPREADCYCLE

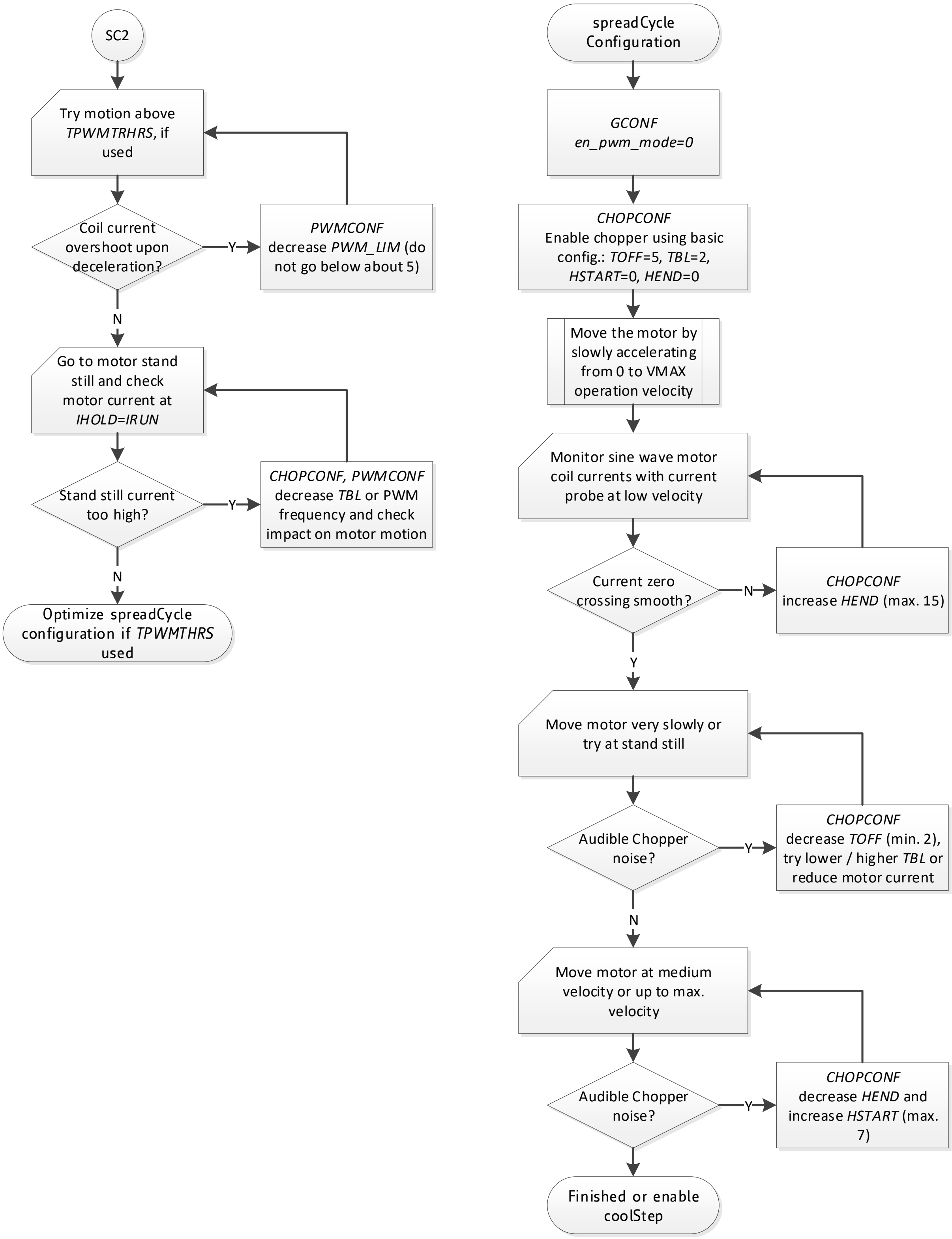


Figure 18.2 Tuning StealthChop and SpreadCycle

ENABLING COOLSTEP (ONLY IN COMBINATION WITH SPREADCYCLE)

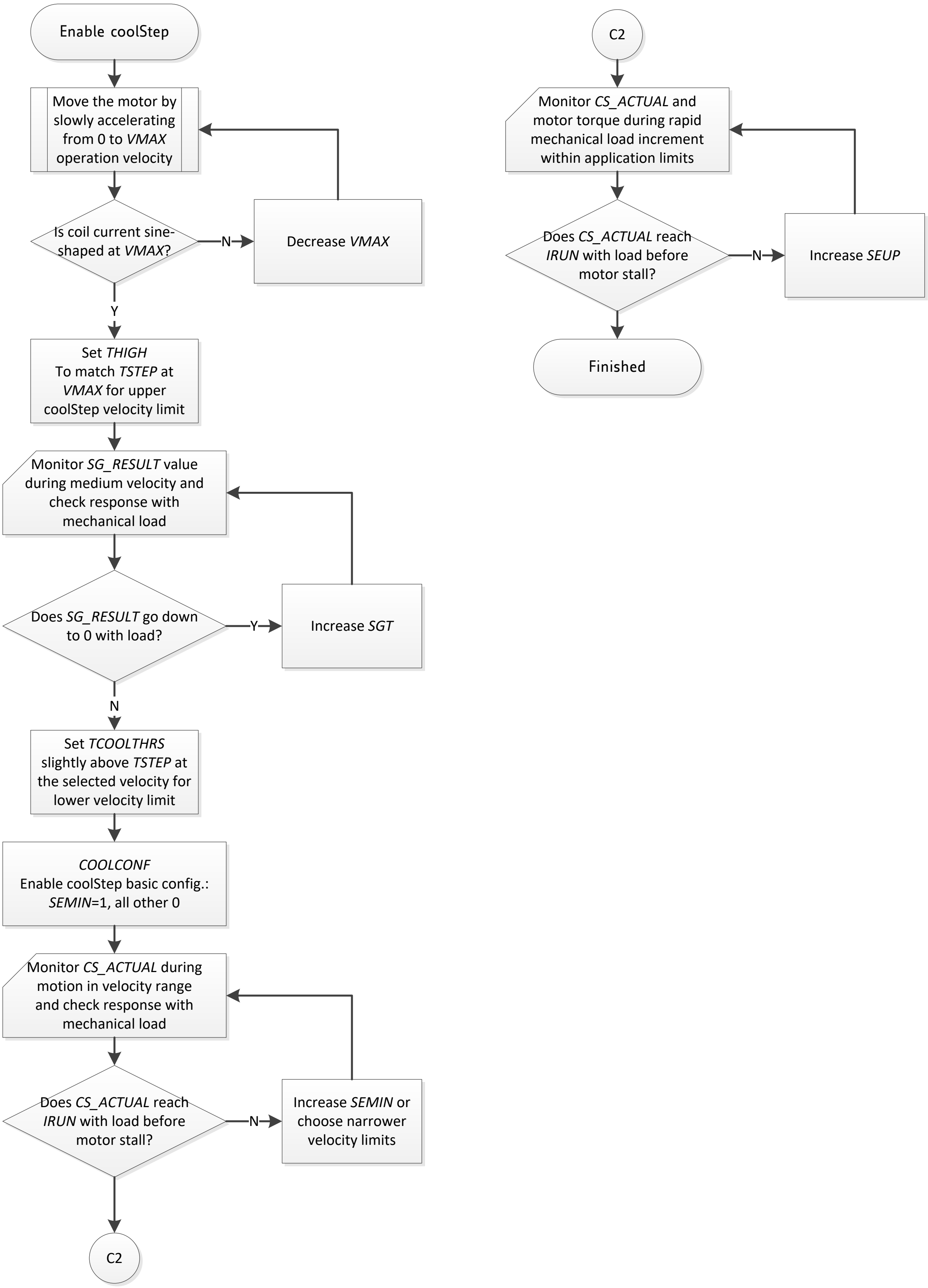


Figure 18.3 Enabling CoolStep (only in combination with SpreadCycle)

SETTING UP DCSTEP

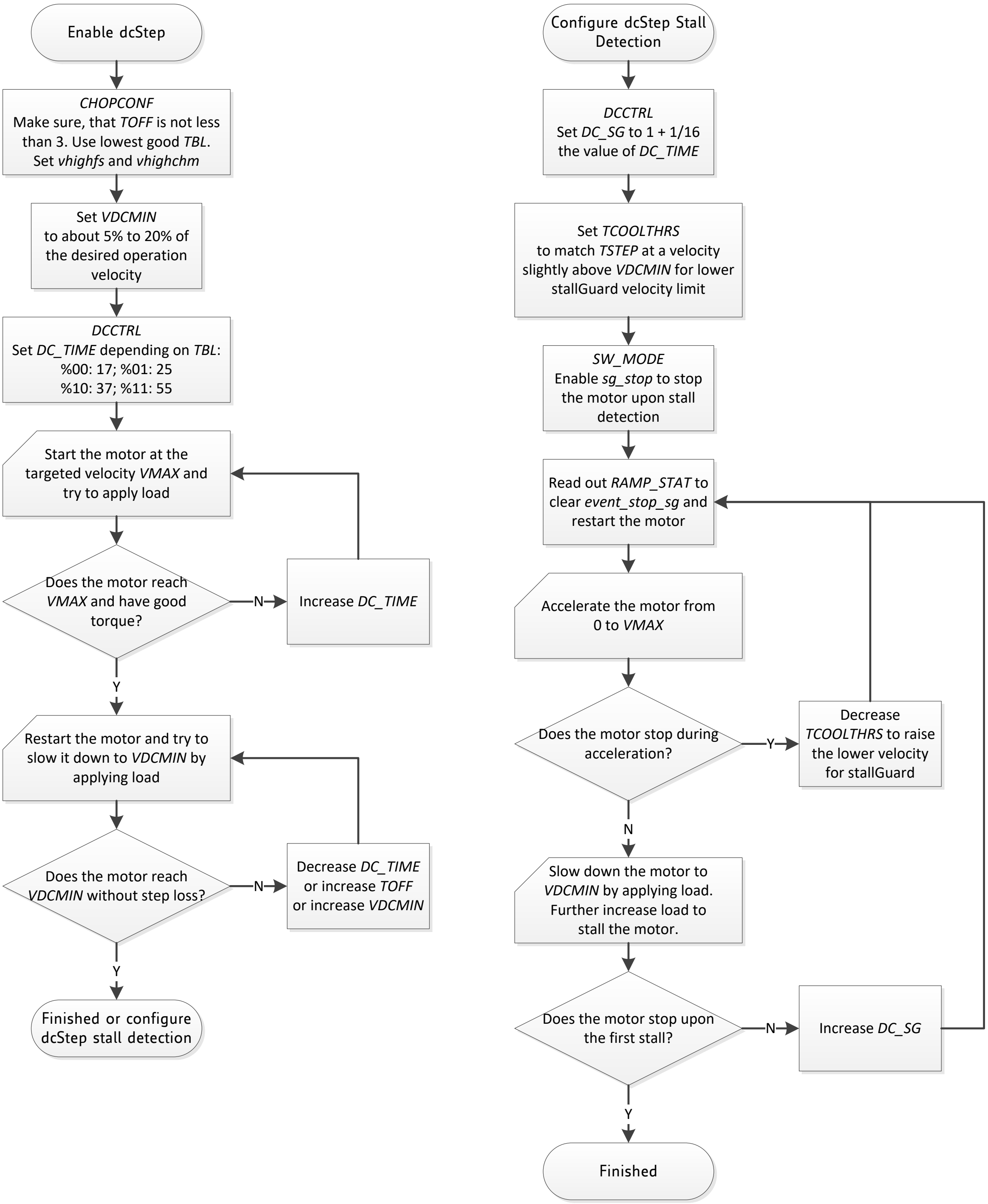


Figure 18.4 Setting up DcStep (using TMC4361 as motion controller)

19 Getting Started

Please refer to the TMC2160 evaluation board to allow a quick start with the device, and to allow interactive tuning of the device setup in your application. Chapter 18 will guide you through the process of correctly setting up all registers.

19.1 Initialization Examples

SPI datagram example sequence to enable the driver for step and direction operation and initialize the chopper for SpreadCycle operation and for StealthChop at <30 RPM @ 12MHz clock:

```
SPI send: 0xEC000100C3;  // CHOPCONF: TOFF=3, HSTRT=4, HEND=1, TBL=2, CHM=0 (SpreadCycle)
SPI send: 0x9000061F0A;  // IHOLD_IRUN: IHOLD=10, IRUN=31 (max. current), IHOLDDELAY=6
SPI send: 0x910000000A;  // TPOWERDOWN=10: Delay before power down in stand still
SPI send: 0x8000000004;  // EN_PWM_MODE=1 enables StealthChop (with default PWMCONF)
SPI send: 0x93000001F4;  // TPWM_THRS=500 yields a switching velocity about 35000 = ca. 30RPM
```

Hint
Tune the configuration parameters for your motor and application for optimum performance.

20 Standalone Operation

For standalone operation, no SPI interface is required to configure the TMC2160. All pins with suffix CFG0 to CFG6 have a special meaning in this mode and can be tied either to VCC_IO or to GND.

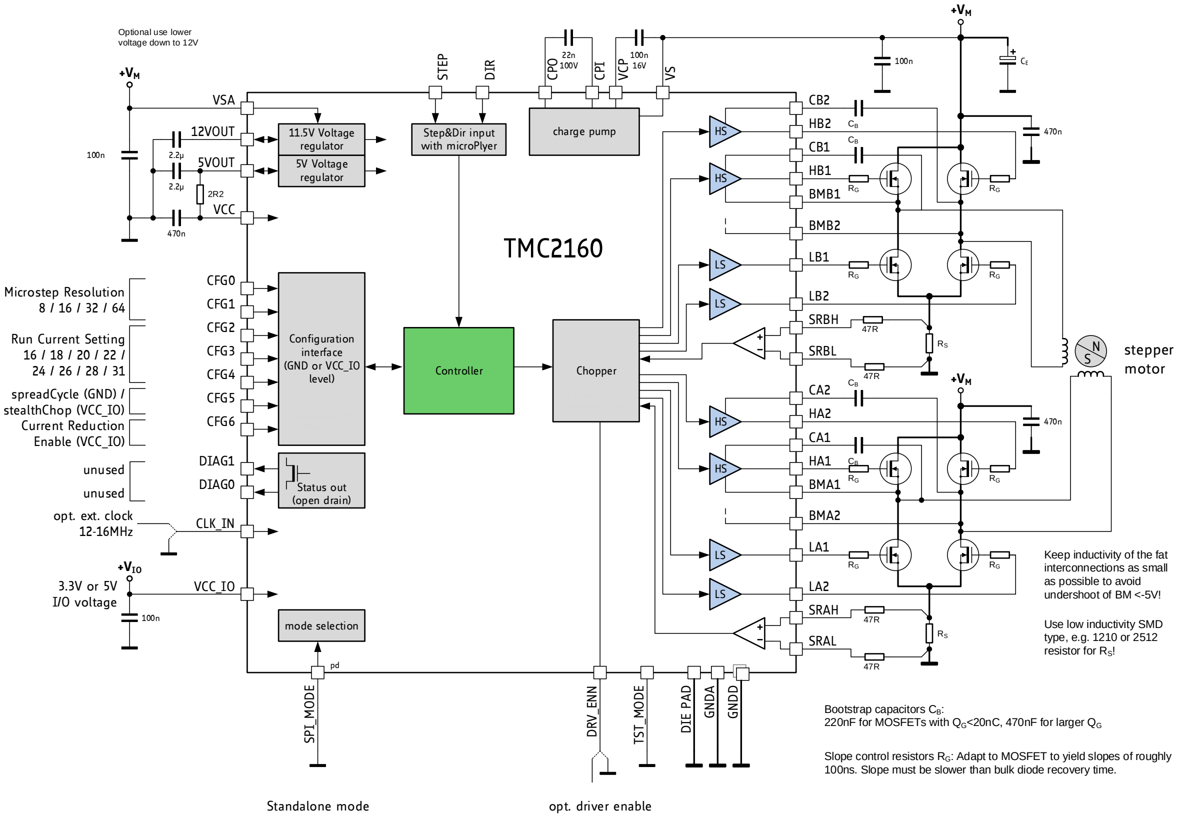


Figure 20.1 Standalone operation with TMC2160 (pins shown with their standalone mode names)

To activate standalone mode, tie pin SPI_MODE to GND. In this mode, the driver acts as a pure STEP and DIR driver. SPI and single wire are off. The driver works in SpreadCycle mode or StealthChop mode. With regard to the register set, the following settings are activated:

The following settings are affected by the CFG pins to ensure correct configuration:

CFG0/CFG1: CONFIGURATION OF MICROSTEP RESOLUTION FOR STEP INPUT		
CFG1	**CFG0**	**Microstep Setting**
GND	GND	8 microsteps, *MRES*=5
GND	VCC_IO	16 microsteps, *MRES*=4
VCC_IO	GND	32 microsteps, *MRES*=3
VCC_IO	VCC_IO	64 microsteps, *MRES*=2

CFG4/CFG3/CFG2: CONFIGURATION OF RUN CURRENT

CFG4	CFG3	CFG2	*IRUN* Setting
GND	GND	GND	*IRUN*=16
GND	GND	VCC_IO	*IRUN*=18
GND	VCC_IO	GND	*IRUN*=20
GND	VCC_IO	VCC_IO	*IRUN*=22
VCC_IO	GND	GND	*IRUN*=24
VCC_IO	GND	VCC_IO	*IRUN*=26
VCC_IO	VCC_IO	GND	*IRUN*=28
VCC_IO	VCC_IO	VCC_IO	*IRUN*=31

CFG5: SELECTION OF CHOPPER MODE

CFG5	Chopper Setting
GND	SpreadCycle operation. (*TOFF*=3)
VCC_IO	StealthChop operation. (*GCONF*.en_PWM_mode=1)

CFG6: CONFIGURATION OF HOLD CURRENT REDUCTION

CFG6*)	Chopper Setting
GND	No hold current reduction. IHOLD=IRUN
VCC_IO	Reduction to 50%. IHOLD=1/2 IRUN

Hint
Be sure to allow the motor to rest for at least 100ms (assuming a minimum of 10MHz f_{CLK}) before starting a motion using StealthChop. This will allow the current regulation to set the initial motor current.

**) CFG6: Attention*
CFG6 pin draws significant current (20mA) when driven to a different level than CFG5, because the output driver tries to make CFG6 level equal to CFG5. Therefore, a 0 Ohm resistor is required to pull up/down CFG6. Due to this, setting CFG6 different from CFG5 is only recommended with external VCC_IO supply at 3.3V level.

Attention:
DIAG outputs are not configured per default. They can be activated using the interfaces before switching to standalone mode.

21 Power-Up Reset

The chip is loaded with default values during power-up via its internal power-on reset. It will also reset to power-up defaults in case any of the supply voltages monitored by internal reset circuitry (VSA, +5VOUT or VCC_IO) falls below the undervoltage threshold. VCC is not monitored. Therefore, VCC must not be lost during operation of the chip. In case of a microcontroller software re-boot, disable the driver (*TOFF*=0), re-initialize all registers used by the software and stop any motion in progress by slowing down the ramp generator. A hardware reset requires cycling VCC_IO while keeping all digital inputs at a low level at the same time. Actively drive VCC_IO to a low level to ensure that it falls below the lower reset threshold. Current consumed from VCC_IO is low and therefore it has simple driving requirements. Due to the input protection diodes not allowing the digital inputs to rise above VCC_IO level, any active high input would hinder VCC_IO from going down.

22 Clock Oscillator and Input

The clock is the timing reference for all functions: the chopper, DcStep, blank time, etc. Many parameters are scaled with the clock frequency; thus, a precise reference allows a more deterministic result. The factory-trimmed on-chip clock oscillator provides a good and stable timing for most use cases.

22.1 Using the Internal Clock

Directly tie the CLK input to GND near to the IC if the internal clock oscillator is to be used. It will be sufficient for applications, where a velocity precision of roughly +-4% is tolerable.

22.2 Using an External Clock

When an external clock is available, a frequency of 10 MHz to 16 MHz is recommended for optimum performance. The duty cycle of the clock signal is uncritical, as long as minimum high or low input time for the pin is satisfied (refer to electrical characteristics). Up to 18 MHz can be used, when the clock duty cycle is 50%. Make sure, that the clock source supplies clean CMOS output logic levels and steep slopes when using a high clock frequency. The external clock input is enabled with the second positive polarity seen on the CLK input.

Hint

Switching off the external clock frequency prevents the driver from operating normally. Therefore, an internal watchdog switches back to internal clock in case the external signal is missing for more than roughly 32 internal clock cycles.

22.2.1 Considerations on the Frequency

A higher frequency allows faster SPI operation and higher chopper frequencies. On the other hand, it causes more power dissipation in the TMC2160 digital core and 5V voltage regulator. Generally, a frequency of 10 MHz to 12 MHz should be sufficient for most applications. At higher clock frequency, the VSA supply voltage should be connected to a lower voltage for applications working at more than 24V nominal supply voltage. For reduced requirements concerning the motor dynamics, a clock frequency of down to 8 MHz (or even lower) can be considered.

23 Absolute Maximum Ratings

The maximum ratings may not be exceeded under any circumstances. Operating the circuit at or near more than one maximum rating at a time for extended periods shall be avoided by application design.

Parameter	Symbol	Min	Max	Unit
Supply voltage operating with inductive load	V_{VS}, V_{VSA}	-0.5	60	V
Supply and bridge voltage short time peak (limited by peak voltage on charge pump output and Cxx pins*)	V_{VSMAX}		64	V
VSA when different from VS	V_{VSAMAX}	-0.5	60	V
Peak voltages on Cxx bootstrap pins and VCP	V_{CxCP}		76	V
Supply voltage V12	V_{12VOUT}	-0.5	14	V
Peak voltages on BM pins (due to stray inductivity)	V_{BMx}	-6	V_{VS}+6	V
Peak voltages on Cxx bootstrap pins relative to BM	V_{CxBMx}	-0.5	16	V
I/O supply voltage on VCC_IO	V_{VIO}	-0.5	5.5	V
digital VCC supply voltage (normally supplied by 5VOUT)	V_{VCC}	-0.5	5.5	V
Logic input voltage	V_I	-0.5	V_{VIO}+0.5	V
Maximum current to / from digital pins and analog low voltage I/Os (short time peak current)	I_{IO}		+/-500	mA
5V regulator output current (internal plus external load)	I_{5VOUT}		30	mA
5V regulator continuous power dissipation (V_{VSA}-5V) * I_{5VOUT}	P_{5VOUT}		1	W
12V regulator output current (internal plus external load)	I_{12VOUT}		20	mA
12V regulator continuous power dissipation (V_{VSA}-12V) * I_{12VOUT}	P_{12VOUT}		0.5	W
Junction temperature	T_J	-50	150	°C
Storage temperature	T_{STG}	-55	150	°C
ESD-Protection for interface pins (Human body model, HBM)	V_{ESDAP}		4	kV
ESD-Protection for handling (Human body model, HBM)	V_{ESD}		1	kV

*) Stray inductivity of power routing will lead to ringing of the supply voltage when driving an inductive load. This ringing results from the fast switching slopes of the driver outputs in combination with reverse recovery of the body diodes of the output driver MOSFETs. Even small trace inductivities as well as stray inductivity of sense resistors can easily generate a few volts of ringing leading to temporary voltage overshoot. This should be considered when working near the maximum voltage.

24 Electrical Characteristics

24.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	T_J	-40	125	°C
Supply voltage for motor and bridge	V_{VS}	10	55	V
Supply voltage VSA	V_{VSA}	10	50	V
Supply voltage for VSA and 12OUT (internal gate voltage regulator bridged)	V_{12VOUT}, V_{VSA}	10	13	V
Lower Supply voltage (reduced spec, short to GND protection not functional), lower limit depending on MOSFETs gate threshold voltage and load current	V_{VS}	8		V
I/O supply voltage on VCC_IO	V_{VIO}	3.00	5.25	V

24.2 DC and Timing Characteristics

DC characteristics contain the spread of values guaranteed within the specified supply voltage range unless otherwise specified. Typical values represent the average value of all parts measured at +25°C. Temperature variation also causes stray to some values. A device with typical values will not leave Min/Max range within the full temperature range.

Power supply current	**DC-Characteristics** $V_{VS} = V_{VSA} = 24.0V$					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Total supply current, driver disabled $I_{VS} + I_{VSA}$	I_S	f_{CLK}=12MHz / internal clock		18	24	mA
VSA supply current (VS and VSA separated)	I_{VSA}	f_{CLK}=12MHz / internal clock, driver disabled		15		mA
Total supply current, operating, MOSFETs AOD4126, $I_{VS} + I_{VSA}$	I_S	f_{CLK}=12MHz, 23.4kHz chopper, no load		25		mA
Internal current consumption from 5V supply on VCC pin	I_{VCC}	f_{CLK}=12MHz		10		mA
Internal current consumption from 5V supply on VCC pin	I_{VCC}	f_{CLK}=16MHz		12.5		mA
IO supply current on VCC_IO (typ. at 5V)	I_{VIO}	no load on outputs, inputs at V_{IO} or GND Excludes pullup / pull-down resistors		15	30	µA

Motor driver section	**DC- and Timing-Characteristics** V_{VS} = 24.0V; Tj=50°C					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
RDS_{ON} lowside off driver	R_{ONL}	Gate off		1.8	3	Ω
RDS_{ON} highside off driver	R_{ONH}	Gate off		2.2	3.5	Ω
Gate drive current low side MOSFET turning on at 2V V_{GS}	I_{SLPON0}	*DRVSTRENGTH*=0		200		mA
	I_{SLPON2}	*DRVSTRENGTH*=2		400		mA
	I_{SLPON3}	*DRVSTRENGTH*=3		600		mA
Gate drive current high side MOSFET turning on at 2V V_{GS}	I_{SLPON0}	*DRVSTRENGTH*=0		150		mA
	I_{SLPON2}	*DRVSTRENGTH*=2		300		mA
	I_{SLPON3}	*DRVSTRENGTH*=3		450		mA
BBM time via internal delay (start of gate switching off to start of gate switching on)	t_{BBM0}	*BBMCLKS*=0 *BBMTIME*=0	75	100		ns
	t_{BBM16}	*BBMTIME*=16		200		ns
	t_{BBM16}	*BBMTIME*=24		375	500	ns

Charge pump	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Charge pump output voltage	V_{VCP}-V_{VS}	operating	V_{12VOUT} - 2	V_{12VOUT} - 1		V
Charge pump voltage threshold for undervoltage detection	V_{VCP}-V_{VS}	rising, using internal 5V regulator voltage	4.5	5	6.5	V
Charge pump frequency	f_{CP}			1/16 f_{CLKOSC}		

Linear regulator	**DC-Characteristics** V_{VS} = V_{VSA} = 24.0V					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Output voltage	V_{5VOUT}	T_J = 25°C	4.80	5.0	5.20	V
Deviation of output voltage over the full temperature range	$V_{5VOUT(DEV)}$	drivers disabled T_J = full range		+/-30	+/-100	mV
Deviation of output voltage over the full supply voltage range	$V_{5VOUT(DEV)}$	drivers disabled, internal clock T_A = 25°C V_{VSA} = 10V to 30V			+/-50	mV / 10V
Output voltage	V_{12VOUT}	operating, internal clock T_J = 25°C	10.8	11.5	12.2	V

Clock oscillator and input	**Timing-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Clock oscillator frequency (factory calibrated)	f_{CLKOSC}	t_J=-50°C		11.7		MHz
	f_{CLKOSC}	t_J=50°C	11.5	12.0	12.5	MHz
	f_{CLKOSC}	t_J=150°C		12.1		MHz
External clock frequency (operating)	f_{CLK}		4	10-16	18	MHz
External clock high / low level time	t_{CLKH} / t_{CLKL}	CLK driven to 0.1 V_{VIO} / 0.9 V_{VIO}	16			ns
External clock first pulse to trigger switching to external CLK	t_{CLKH} / t_{CLKL}	CLK driven high *A-version*	16			ns
External clock first pulse to trigger switching to external CLK	t_{CLKH} / t_{CLKL}	CLK driven high *non-A-version* only	30	25		ns
External clock timeout detection in cycles of internal f_{CLKOSC}	t_{CLKH1}	CLK driven high	32		48	cycles f_{CLKOSC}

Short detection	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Short to GND / Short to VS detector delay (Start of gate switch on to short detected) Including 100ns filtering time	t_{SD0}	*FILT_ISENSE*=0 *S2xx_LEVEL*=6 *shortdelay*=0	0.5	0.85	1.1	µs
	t_{SD1}	*shortdelay*=1	1.1	1.6	2.2	µs
Short detector level S2VS (measurement includes drop in sense resistor)	V_{BM}	*S2VS_LEVEL*=15	1.4	1.56	1.72	V
		S2VS_LEVEL=6	0.55	0.625	0.70	V
Short detector level S2G	V_S - V_{BM}	*S2G_LEVEL*=15; VS<52V	1.2	1.56	1.9	V
		S2G_LEVEL=15; VS<55V	0.85			V
		S2G_LEVEL=6; VS<52V	0.46	0.625	0.80	V

Detector levels	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
V_{VSA} undervoltage threshold for RESET	V_{UV_VSA}	V_{VSA} rising	3.6	4	4.6	V
V_{5VOUT} undervoltage threshold for RESET	V_{UV_5VOUT}	V_{5VOUT} rising		3.5		V
V_{VCC_IO} undervoltage threshold for RESET	V_{UV_VIO}	V_{VCC_IO} rising (delay typ. 10µs)	2.0	2.5	3.0	V
V_{VCC_IO} undervoltage detector hysteresis	$V_{UV\_VIOHYST}$			0.3		V
Overtemperature prewarning 120°C	T_{OTPW}	Temperature rising	100	120	140	°C
Overtemperature shutdown 136 °C	T_{OT136}	Temperature rising		136		°C
Overtemperature shutdown 143 °C	T_{OT143}	Temperature rising		143		°C
Overtemperature shutdown 150 °C	T_{OT150}	Temperature rising	135	150	170	°C

Sense resistor voltage levels	**DC-Characteristics** **f_{CLK}=16MHz**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Sense input peak threshold voltage (V_{SRxH}-V_{SRxL})	V_{SRT}	*GLOBALSCALER*=0 *csactual*=31 *sin_x*=248 Hyst.=0; I_{BRxy}=0		325		mV
Sense input tolerance / motor current full-scale tolerance -using internal reference	I_{COIL}	*GLOBALSCALER*=0	-5		+5	%

Digital pins	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Input voltage low level	V_{INLO}		-0.3		0.3 V_{VIO}	V
Input voltage high level	V_{INHI}		0.7 V_{VIO}		V_{VIO}+0.3	V
Input Schmitt trigger hysteresis	V_{INHYST}			0.12 V_{VIO}		V
Output voltage low level	V_{OUTLO}	I_{OUTLO} = 2mA			0.2	V
Output voltage high level	V_{OUTHI}	I_{OUTHI} = -2mA	V_{VIO}-0.2			V
Input leakage current	I_{ILEAK}		-10		10	µA
Pullup / pull-down resistors	R_{PU}/R_{PD}		132	166	200	kΩ
Digital pin capacitance	C			3.5		pF

24.3 Thermal Characteristics

The following table shall give an idea on the thermal resistance of the package. The thermal resistance for a four-layer board will provide a good idea on a typical application. Actual thermal characteristics will depend on the PCB layout, PCB type and PCB size. The thermal resistance will benefit from thicker CU (inner) layers for spreading heat horizontally within the PCB. Also, air flow will reduce thermal resistance.

Parameter	Symbol	Conditions	Typ	Unit
Typical power dissipation	P_D	StealthChop or SpreadCycle, 40 or 20kHz chopper, 24V, internal supply regulators	0.6	W
Thermal resistance junction to ambient on a multilayer board	R_{TMJA}	Dual signal and two internal power plane board (2s2p) as defined in JEDEC EIA JESD51-5 and JESD51-7 (FR4, 35µm CU, 70mm x 133mm, d=1.5mm)	21	K/W
Thermal resistance junction to board	R_{TJB}	PCB temperature measured within 1mm distance to the package leads	8	K/W
Thermal resistance junction to case	R_{TJC}	Junction temperature to heat slug of package	3	K/W

Table 24.1 Thermal characteristics TQFP48-EP

The thermal resistance in an actual layout can be tested by checking for the heat up caused by the standby power consumption of the chip. When no motor is attached, all power seen on the power supply is dissipated within the chip.

25 Layout Considerations

25.1 Exposed Die Pad

The TMC2160 uses its die attach pad to dissipate heat from the gate drivers and the linear regulator to the board. For best electrical and thermal performance, use a reasonable amount of solid, thermally conducting vias between the die attach pad and the ground plane. The printed circuit board should have a solid ground plane spreading heat into the board and providing for a stable GND reference.

25.2 Wiring GND

All signals of the TMC2160 are referenced to their respective GND. Directly connect all GND pins under the device to a common ground area (GND, GNDP, GNDA and die attach pad). The GND plane right below the die attach pad should be treated as a virtual star point. For thermal reasons, the PCB top layer shall be connected to a large PCB GND plane spreading heat within the PCB.

Attention
Place the TMC2160 near to the MOSFET bridge and sense resistor GND to avoid ringing leading to GND differences and to dangerous inductive peak voltages.

25.3 Wiring Bridge Supply

The power bridge will draw the full coil current in pulses with extremely high dI/dt. Thus, any inductivity between VS supply filtering and the MOSFETs can lead to severe voltage spikes. This must be avoided. Avoid any bend in the supply traces between filtering capacitors and MOSFET switches and keep distance as small as possible. Especially for high current, use a separate plane for the supply voltage, and a sufficient number and capacity for supply filtering. Use an additional capacitor for the IC VS pin, as additional ripple voltage would cause severe current spikes on the charge pump capacitor. A tiny series resistor can be added to avoid this.

Attention
Keep supply voltage ripple low, by using sufficient filtering capacity close to the MOSFET bridge.

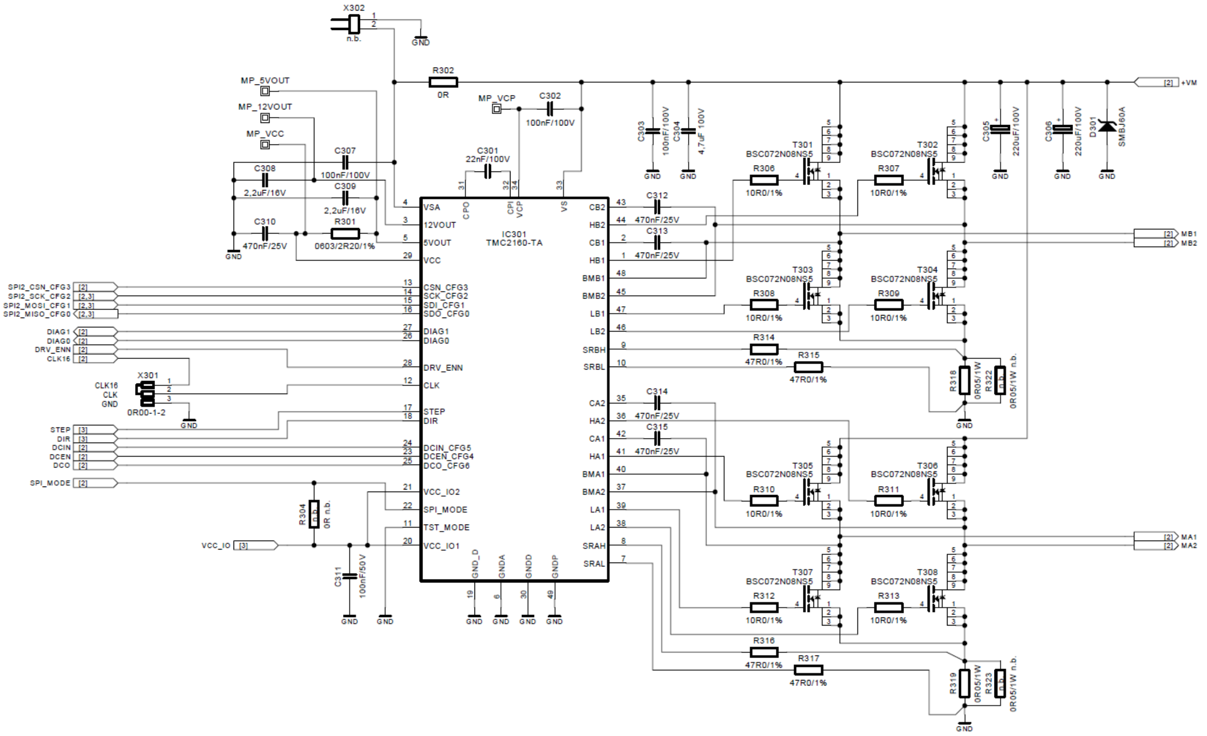
25.4 Supply Filtering

The 5VOUT output voltage ceramic filtering capacitor (2.2 to 4.7 µF recommended) should be placed as close as possible to the 5VOUT pin, with its GND return going directly to the GNDA pin. This ground connection shall not be shared with other loads or additional vias to the GND plane. Use as short and as thick connections as possible. For best microstepping performance and lowest chopper noise an additional filtering capacitor should be used for the VCC pin to GND, to avoid digital part ripple influencing motor current regulation. Therefore, place a ceramic filtering capacitor (470nF recommended) as close as possible (1-2mm distance) to the VCC pin with GND return going to the ground plane. VCC can be coupled to 5VOUT using a 2.2 Ω or 3.3 Ω resistor to supply the digital logic from 5VOUT while keeping ripple away from this pin. A 100 nF filtering capacitor should be placed as close as possible to the VSA pin to ground plane. Make sure, that VS does not see excessive voltage spikes caused by bridge operation and place a 100 nF or larger filter capacitor to GND close to the VS pin.

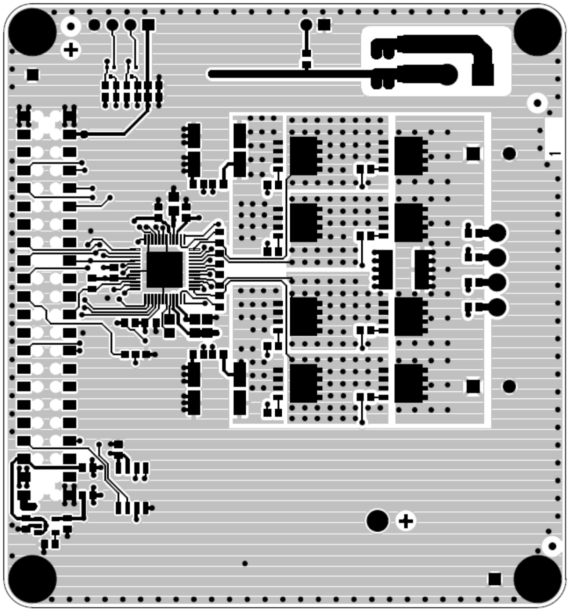
Please carefully read chapters 3.3 and 3.4 to understand the special considerations regarding layout and component selection for the external MOSFET power bridges.

25.5 Layout Example

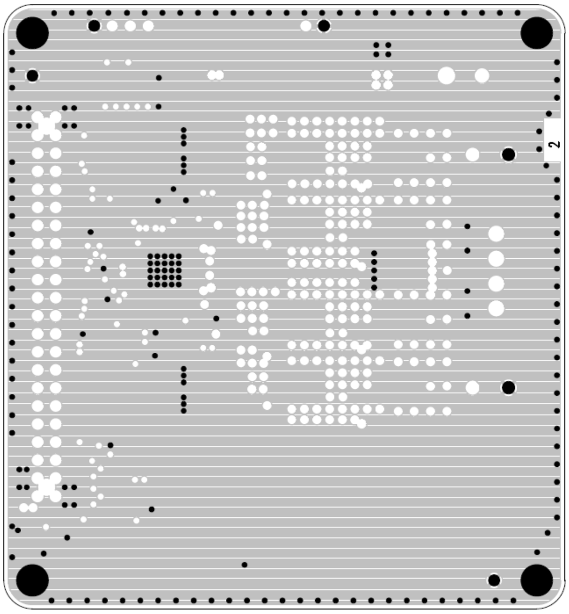
Schematic (TMC2160+MOSFETs shown)



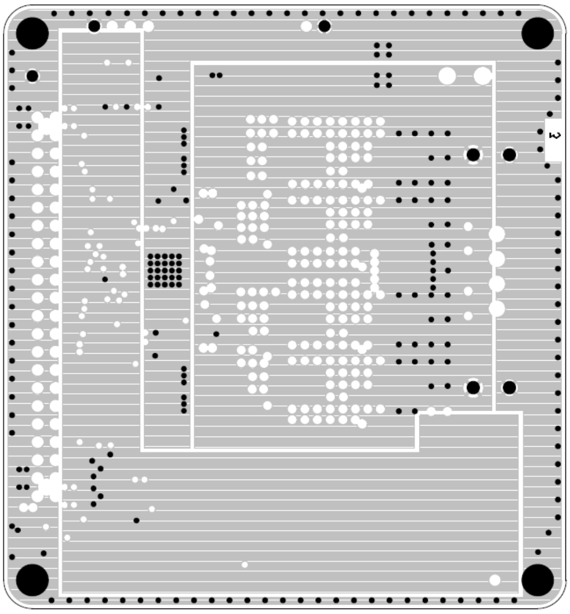
1- Top Layer (assembly side)



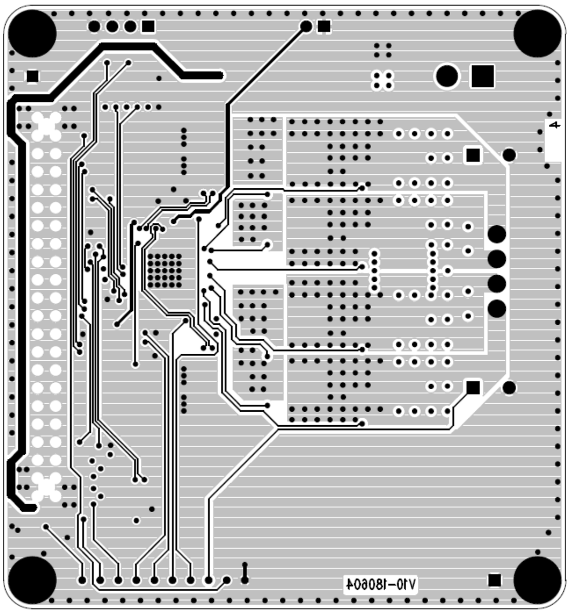
2- Inner Layer (GND)



3- Inner Layer (supply VS)



4- Bottom Layer



Components

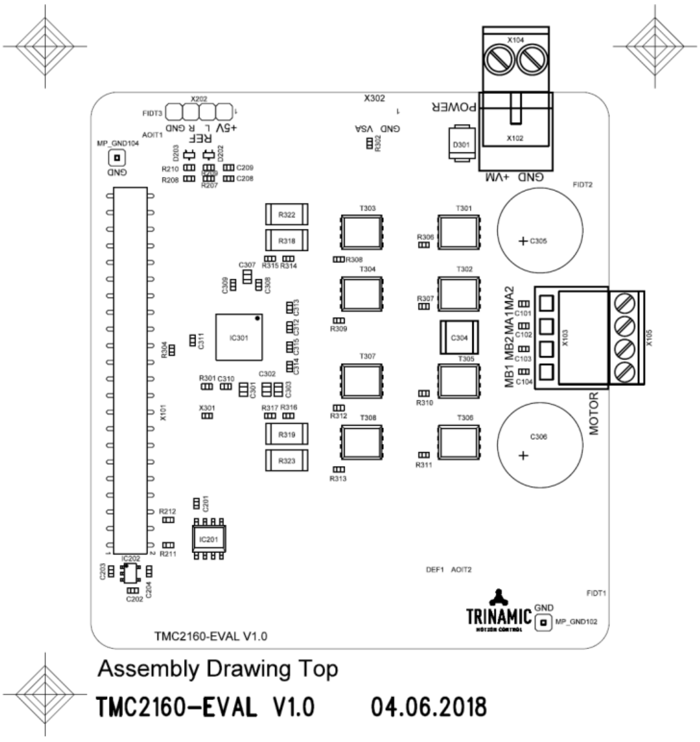


Figure 25.1 Layout example

Hint

When using the TQFP package in designs for more than 30V consider PCB coating to satisfy sufficient creeping distances.

26 Package Mechanical Data

26.1 Dimensional Drawings TQFP48-EP

Drawings not to scale.

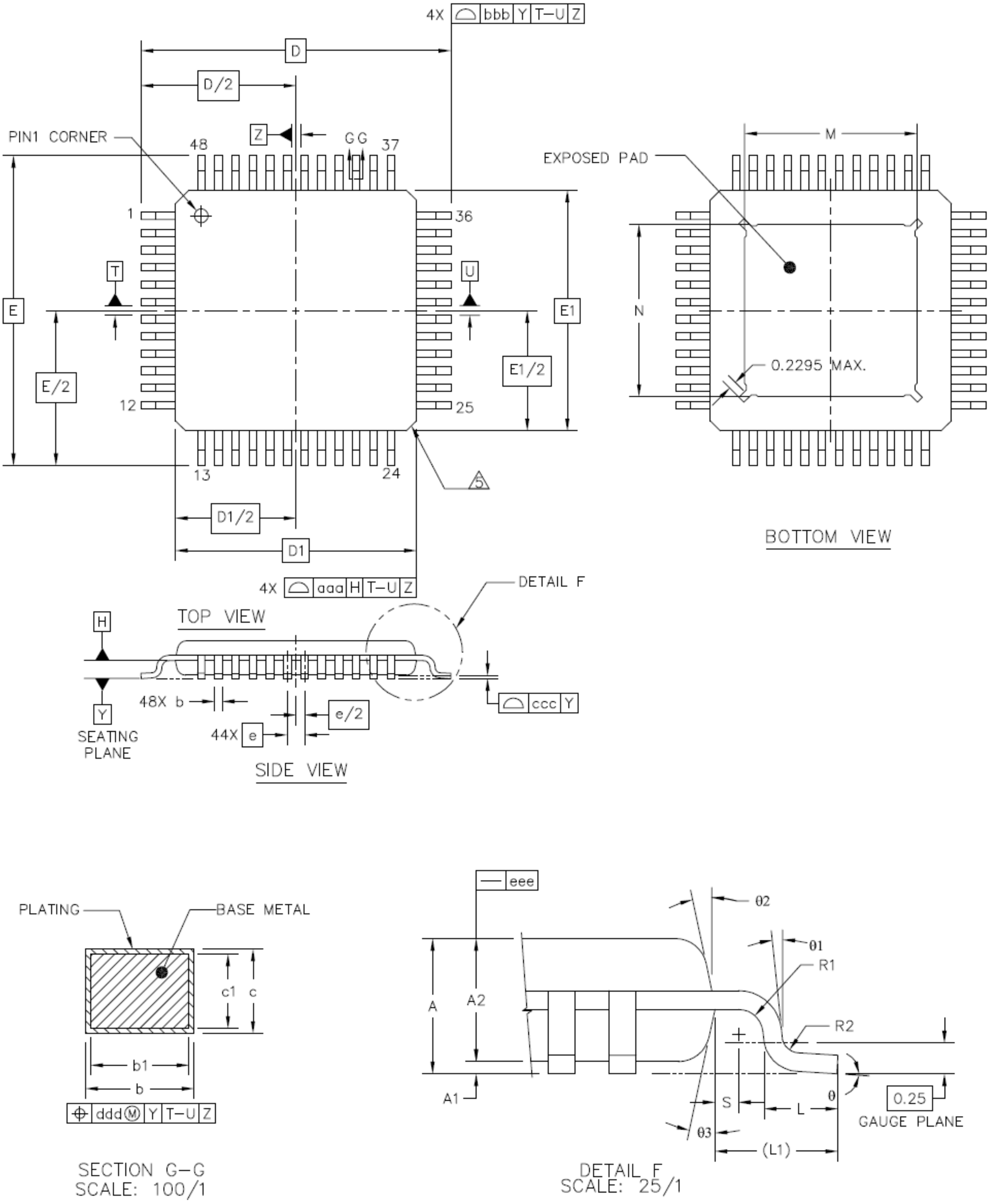


Figure 26.1 Dimensional drawings TQFP48-EP

Parameter	Ref	Min	Nom	Max
total thickness	A	-	-	1.2
stand off	A1	0.05	-	0.15
mold thickness	A2	0.95	1	1.05
lead width (plating)	b	0.17	0.22	0.27
lead width	b1	0.17	0.2	0.23
lead frame thickness (plating)	c	0.09	-	0.2
lead frame thickness	c1	0.09	-	0.16
body size X (over pins)	D		9.0	
body size Y (over pins)	E		9.0	
body size X	D1		7.0	
body size Y	E1		7.0	
lead pitch	e		0.5	
lead	L	0.45	0.6	0.75
footprint	L1		1 REF	
	Θ	0°	3.5°	7°
	Θ1	0°	-	-
	Θ2	11°	12°	13°
	Θ3	11°	12°	13°
	R1	0.08	-	-
	R2	0.08	-	0.2
	S	0.2	-	-
exposed die pad size X	M	4.9	5	5.1
exposed die pad size Y	N	4.9	5	5.1
package edge tolerance	aaa			0.2
lead edge tolerance	bbb			0.2
coplanarity	ccc			0.08
lead offset	ddd			0.08
mold flatness	eee			0.05

26.2 Package Codes

Type	Package	Temperature range	Code & marking
TMC2160A-TA	TQFP-EP48 (RoHS)	-40°C … +125°C	TMC2160A-TA
TMC2160A-TA	TQFP-EP48 (RoHS)	-40°C … +125°C	TMC2160A-*L*A (by mistake printed for devices produced with Date Codes: 2002, 2007, 2016, 2019)

27 Disclaimer

TRINAMIC Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Motion Control GmbH & Co. KG. Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

Information given in this data sheet is believed to be accurate and reliable. However, no responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties which may result from its use.

Specifications are subject to change without notice.

All trademarks used are property of their respective owners.

28 ESD Sensitive Device

The TMC2160 is an ESD sensitive CMOS device sensitive to electrostatic discharge. Take special care to use adequate grounding of personnel and machines in manual handling. After soldering the devices to the board, ESD requirements are more relaxed. Failure to do so can result in defect or decreased reliability.

29 Designed for Sustainability

Sustainable growth is one of the most important and urgent challenges today. We at Trinamic try to contribute by designing highly efficient IC products, to minimize energy consumption, ensure best customer experience and long-term satisfaction by smooth and silent run, while minimizing the demand for external resources, e.g., for power supply, cooling infrastructure, reduced motor size and magnet material by intelligent control interfaces and advanced algorithms.

Please help and design efficient and durable products made for a sustainable world.

30 Table of Figures

31 Revision History

Version	Date	Author BD= Bernhard Dwersteg	Description
V0.91	2018-MAY-25	BD	First version of datasheet based on datasheet TMC5160 V1.04
V1.00	2018-JUN-06	BD	Added errata / limitations for initial tuning of AT#1 / AT#2 phase Minor wording, added evaluation board drawing
V1.01	2018-OKT-29	BD	Minor changes, added -T suffix option, S2G >52V hints/limits updated
V1.02	2018-NOV-19	BD	Added hints for tuning MOSFET bridge, added wiring bridge supply
V1.03	2018-FEB-05	BD	Corrected timing requirements for CLK input (30ns for first pulse) / some minor fixes
V1.04	2019-AUG-05	BD	Added changes for TMC2160A, Errata for DIAG output in standalone mode
V1.05	2019-NOV-18	BD	Minor changes
V1.06	2020-MAY-19	BD	Updated logo, added marking error
V1.07	2021-JUN-01	BD	Minor changes, Corrected CUR_A / CUR_B position, Corrected DRVSTRENGTH reset default is 0 (instead of 2)
V1.08	2022-FEB-01	BD	Updated logo & order codes; minor re-wording; Corrected condition for autotuning to include current scale CS; Corrected pre-conditions for open-load detection; added Attention texts
V1.09	2022-MAY-25	BD	P52: Added attention box for open load condition; UV_CP not visible on DIAG0_ERR; Minor fixes

Table 31.1 Document Revisions

32 References

[TMC2160-EVAL] TMC2160-EVAL Evaluation board
[AN001] Trinamic Application Note 001 - Parameterization of SpreadCycle™, www.trinamic.com
[AN002] Trinamic Application Note 002 - Parameterization of StallGuard2™ & CoolStep™, www.trinamic.com
[AN003] Trinamic Application Note 003 - DcStep™, www.trinamic.com
Calculation sheet TMC2160_Calculations.xlsx