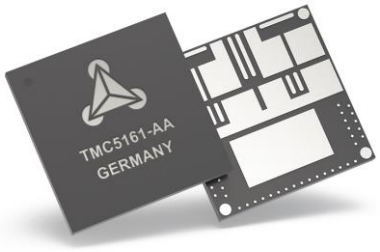
TMC5161 DATASHEET

Compact, low power-dissipation Driver & Controller for two-phase stepper motors. stealthChop™ for quiet movement. Up to 5.5A peak coil current. With Step/Dir Interface and SPI.



dcStep™

coolStep™

stallGuard2™

Applications

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

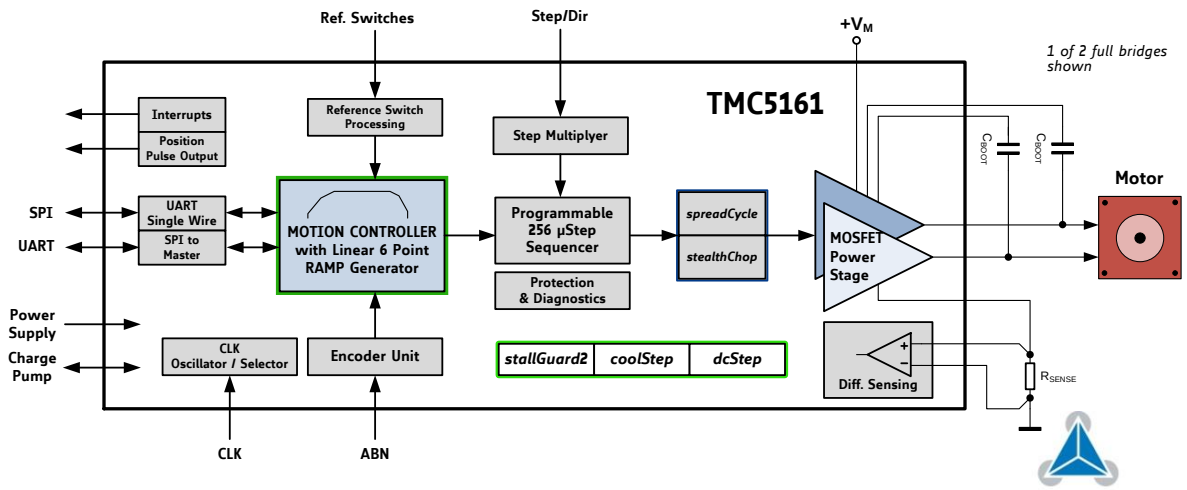
Features and Benefits

2-phase stepper motors up to 3.5A RMS coil current
Step/Dir Interface with 3D optimized interpolation **microPlyer™**
Motion Controller with **sixPoint™** ramp as intelligent peripheral
Voltage Range 8 … 40V DC (55V peak)
Low RDSon integrated 45mΩ MOSFETs
SPI & Single Wire UART
Encoder Interface and **2x Ref.-Switch Input**
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 10x10mm² aQFN

Description

The TMC5161 is a highly compact stepper motor controller and driver IC. Its power stage is optimized for lowest power dissipation and highest dynamics with Nema 17 and Nema 23 motors. It combines a flexible ramp generator for automatic target positioning with industries' most advanced stepper motor driver. Based on TRINAMICs sophisticated spreadCycle and stealthChop choppers, it 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 complete solution reduces learning curve to a minimum while giving best performance in class. Interface-compatible to TMC5160.

Block Diagram



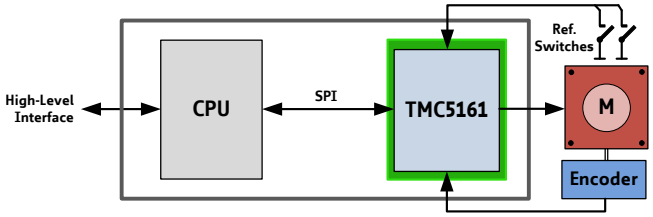
TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

TRINAMIC
MOTION CONTROL

APPLICATION EXAMPLES: COMPACT DRIVES – MULTIPURPOSE USE

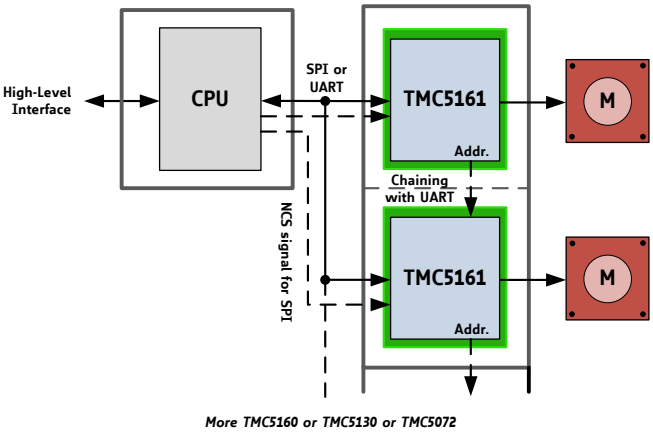
The TMC5161 scores with complete motion controlling features, a powerful integrated MOSFET driver stage, and high-quality current regulation. It offers a versatility that covers a wide spectrum of applications from battery powered, high efficiency systems up to embedded applications with 4A motor current per coil. The TMC5161 contains the complete intelligence which is required to drive a motor. Receiving target positions the TMC5161 manages motor movement. Based on TRINAMICs unique features stallGuard2, coolStep, dcStep, spreadCycle, and stealthChop, the TMC5161 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 TMC5161 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 compatible, integrated TMC5130 driver provides 1.4A of motor current.

MINIATURIZED DESIGN FOR ONE STEPPER MOTOR

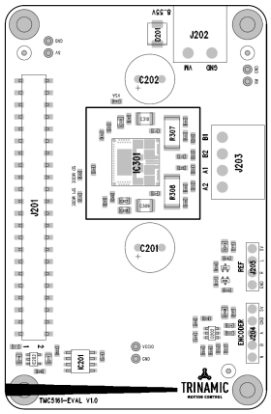


An ABN encoder interface with scaler unit and two reference switch inputs are used to ensure correct motor movement. Automatic interrupt upon deviation is available.

COMPACT DESIGN FOR MULTIPLE STEPPER MOTORS



An application with 2 stepper motors is shown. Additionally, the ABN Encoder interface and two reference switches can be used for each motor. A single CPU controls the whole system, as there are no real time tasks required to move a motor. The CPU-board and the controller / driver boards are highly economical and space saving.



The TMC5161-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 TMC5161 evaluation board system consists of three parts: LANDUNGSBRÜCKE (base board), ESELSBRÜCKE (connector board including several test points), and TMC5161-EVAL.

ORDER CODES

Order code	Description	Size [mm²]
TMC5161-AA	stepper controller/driver with internal MOSFETs; QFN10x10	10 x 10
TMC5161-EVAL	Evaluation board for TMC5161 two phase stepper motor controller/driver	85 x 55
LANDUNGSBRÜCKE	Baseboard for TMC5161-EVAL and further evaluation boards.	85 x 55
ESELSBRÜCKE	Connector board for plug-in evaluation board system.	61 x 38

Table of Contents

1 Principles of Operation

The TMC5161 motion controller and driver chip is an intelligent power component interfacing between CPU and a high power stepper motor. All stepper motor logic is completely within the TMC5161. No software is required to control the motor – just provide target positions. The TMC5161 offers a number of unique enhancements which are enabled by the system-on-chip integration of driver and controller. The sixPoint ramp generator of the TMC5161 uses stealthChop, dcStep, coolStep, and stallGuard2 automatically to optimize every motor movement. The TMC5161 ideally extends the TMC220x, TMC222x, TMC2100, TMC2130 and TMC5130 family to higher motor currents.

THE TMC5161 OFFERS THREE BASIC MODES OF OPERATION:

MODE 1: Full Featured Motion Controller & Driver
All stepper motor logic is completely within the TMC5161. No software is required to control the motor – just provide target positions. Enable this mode by tying low pin SD_MODE.

MODE 2: 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 TMC5161 takes care of intelligent current and mode control and delivers feedback on the state of the motor. The microPlyer automatically smoothens motion. Tie SD_MODE high.

MODE 3: Simple Step & Direction Driver
The TMC5161 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 TMC5161. Optional feedback signals allow error detection and synchronization. Enable this mode by tying pin SPI_MODE low and SD_MODE high.

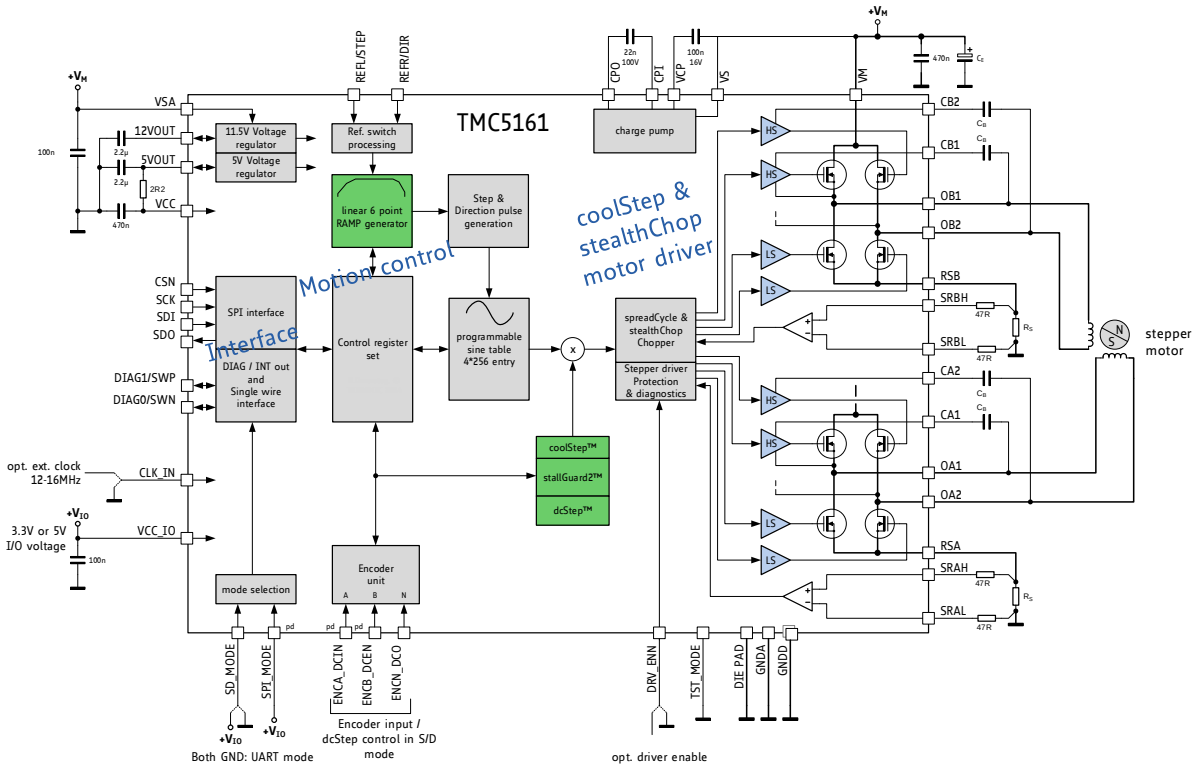


Figure 1.1 TMC5161 basic application block diagram (motion controller)

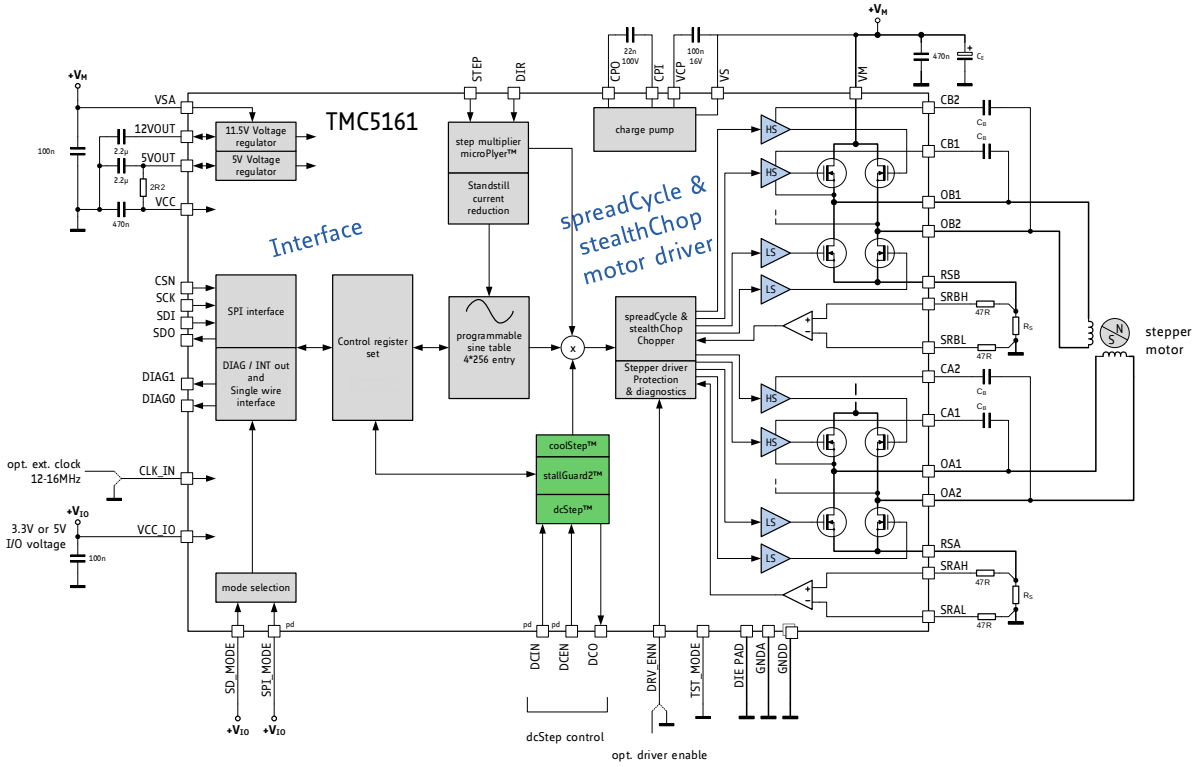


Figure 1.2 TMC5161 STEP/DIR application diagram

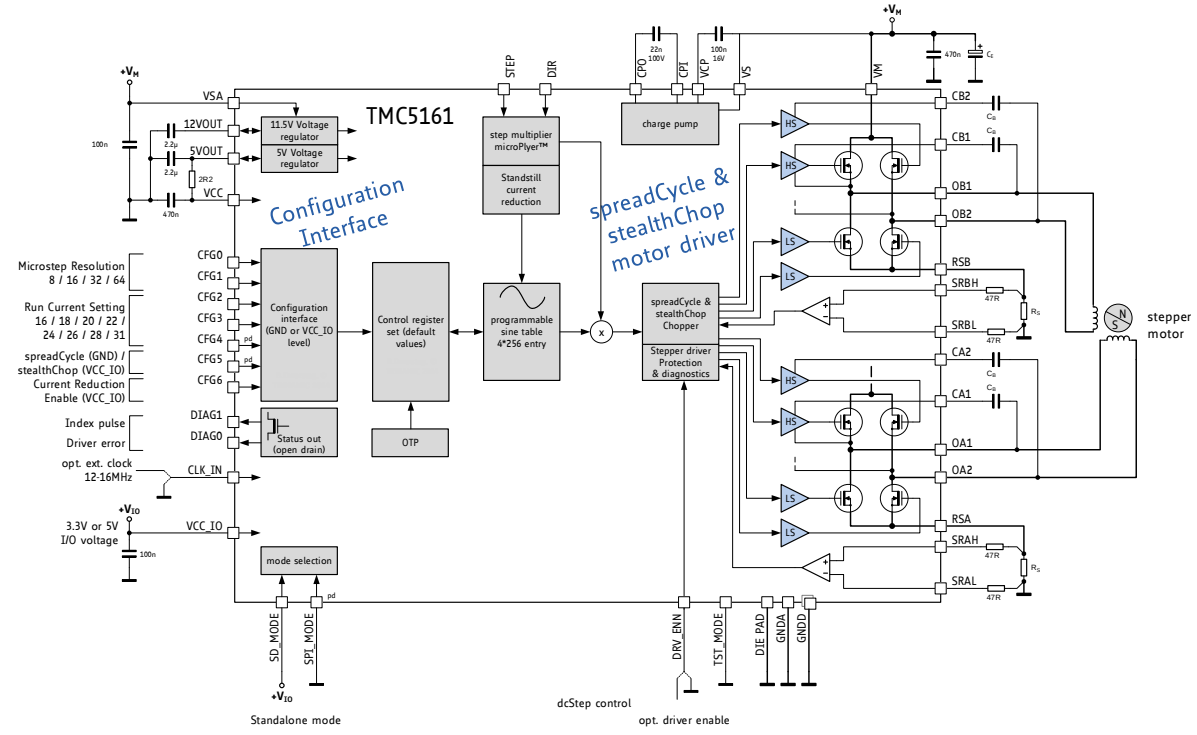


Figure 1.3 TMC5161 standalone driver application diagram

1.1 Key Concepts

The TMC5161 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 TMC5161 supports both, an SPI interface and a UART based single wire interface with CRC checking. Selection of the actual interface is done via the configuration pin SW_SEL, 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 TMC5161 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.

1.2.2 UART Interface

The single wire interface allows differential operation similar to RS485 (using SWP and SWN) or single wire interfacing (leaving open SWN). It can be driven by any standard UART. No baud rate configuration is required.

1.3 Software

From a software point of view the TMC5161 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 and Controlling the Motor

1.4.1 Integrated Motion Controller

The integrated 32 bit motion controller automatically drives the motor to target positions, or accelerates to target velocities. All motion parameters can be changed on the fly. The motion controller recalculates immediately. A minimum set of configuration data consists of acceleration and deceleration values and the maximum motion velocity. A start and stop velocity is supported as well as a second acceleration and deceleration setting. The integrated motion controller supports immediate reaction to mechanical reference switches and to the sensorless stall detection stallGuard2.

Benefits are:

- Flexible ramp programming
- Efficient use of motor torque for acceleration and deceleration allows higher machine throughput
- Immediate reaction to stop and stall conditions

1.4.2 STEP/DIR Interface

The motor can optionally be controlled by a step and direction input. In this case, the motion controller remains unused. 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 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.

1.5 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 both, ramp generator and STEP/DIR operation.

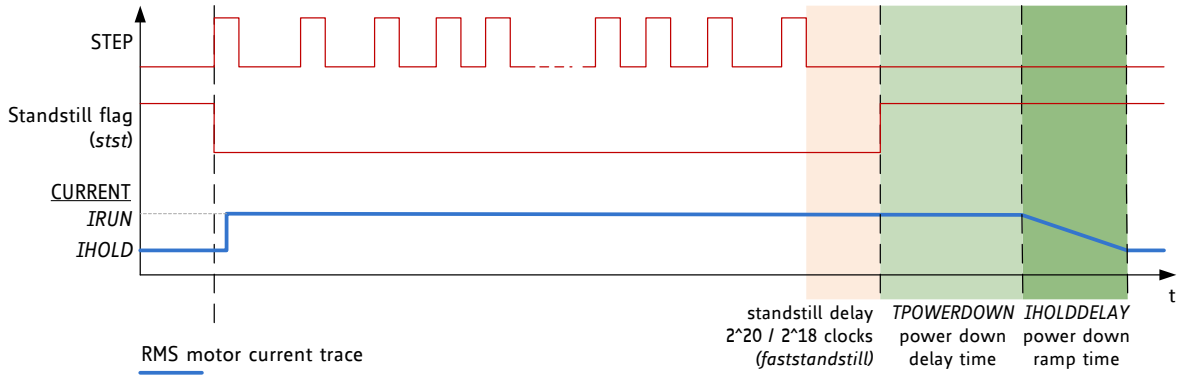


Figure 1.4 Automatic Motor Current Power Down

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 or UART 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 Control

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 1.5 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.

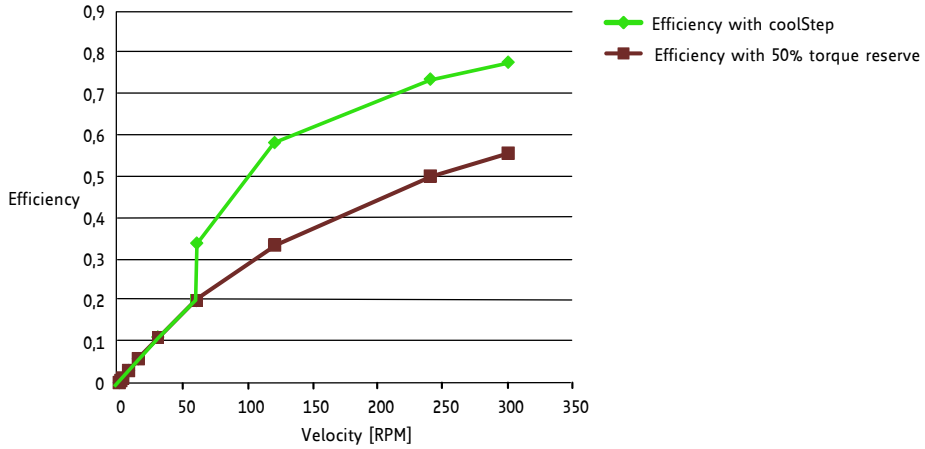


Figure 1.5 Energy efficiency with coolStep (example)

1.9 dcStep – Load Dependent Speed Control

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

1.10 Encoder Interface

The TMC5161 provides an encoder interface for external incremental encoders. The encoder provides automatic checking for step loss and can be used for homing of the motion controller (alternatively to reference switches). A programmable prescaler allows the adaptation of the encoder resolution to the motor resolution. A 32 bit encoder counter is provided.

2 Pin Assignments

2.1 Package Outline

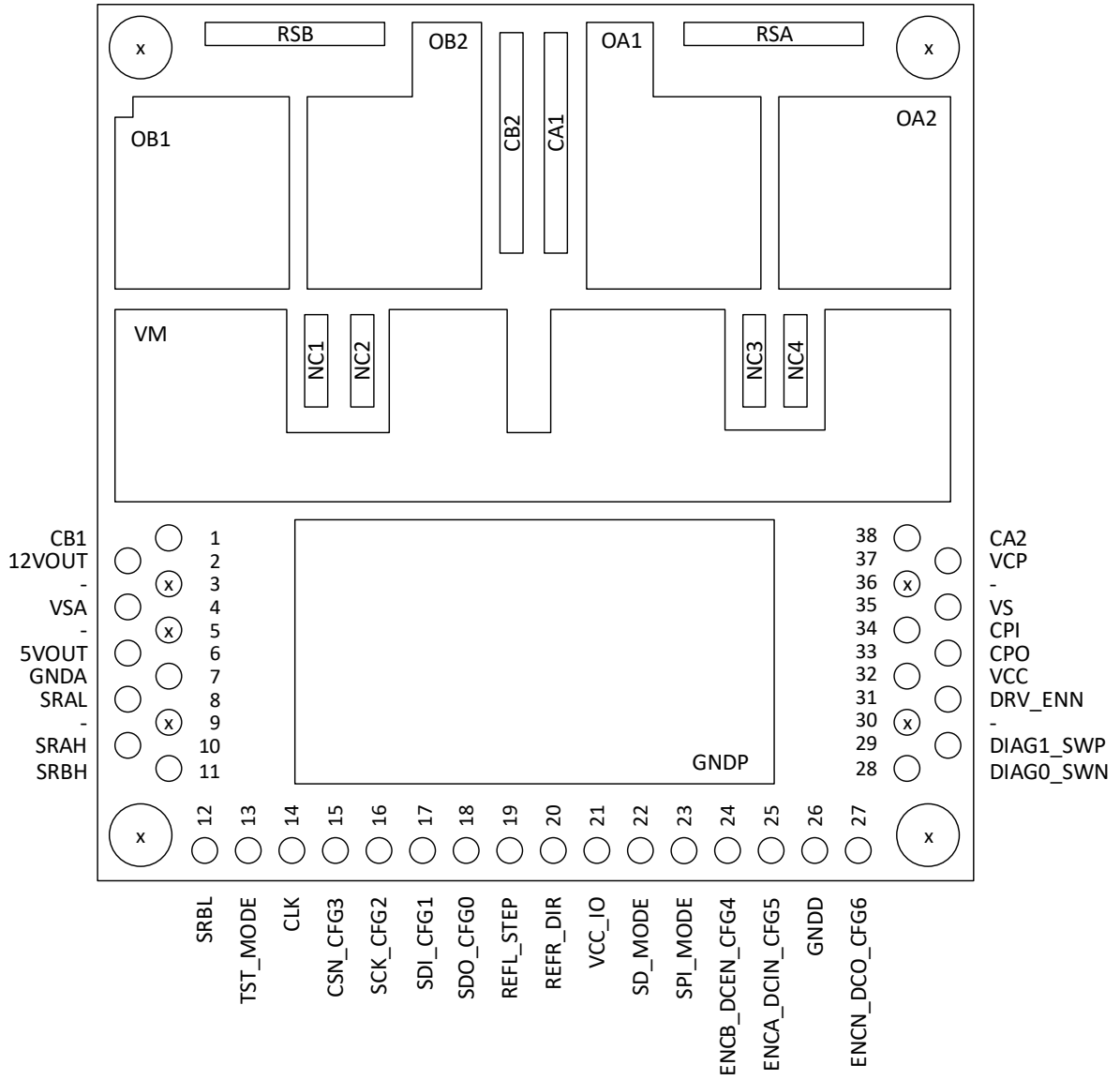


Figure 2.1 TMC5161-LA package and pinning QFN (10x10mm²)

2.2 Signal Descriptions

Pin	Pin	Type	Function
CB1	1		Bootstrap capacitor positive connection.
12VOUT	2		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. In case an external gate voltage supply is available, tie VSA and 12VOUT to the external supply.
unused / x	3, 5, 9, 30, 36, x		Unused pins. May be left open or connected to any potential. The corner pins should be soldered to improve centering.

Pin	Pin	Type	Function
VSA	4		Analog supply voltage for 11.5V and 5V regulator. Normally tied to VS. Provide a 100nF filtering capacitor.
5VOUT	6		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	7		Analog GND. Connect to GND plane near pin.
SRAL	8	AI	Sense resistor GND connection for phase A. Connect to the GND side of the sense resistor in order to compensate for voltage drop on the GND interconnection.
SRAH	10	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	11	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	12	AI	Sense resistor GND connection for phase B. Connect to the GND side of the sense resistor in order to compensate for voltage drop on the GND interconnection.
TST_MODE	13	DI	Test mode input. Tie to GND using short wire.
CLK	14	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	15	DI	SPI chip select input (negative active) (SPI_MODE=1) or Configuration input (SPI_MODE=0)
SCK_CFG2	16	DI	SPI serial clock input (SPI_MODE=1) or Configuration input (SPI_MODE=0)
SDI_CFG1	17	DI	SPI data input (SPI_MODE=1) or Configuration input (SPI_MODE=0) or Next address input (NAI) for single wire interface.
SDO_CFG0	18	DIO	SPI data output (tristate) (SPI_MODE=1) or Configuration input (SPI_MODE=0) or Next address output (NAO) for single wire interface.
REFL_STEP	19	DI	Left reference input (for internal ramp generator) or STEP input when (SD_MODE=1).
REFR_DIR	20	DI	Right reference input (for internal ramp generator) or DIR input (SD_MODE=1).
VCC_IO	21		3.3V to 5V IO supply voltage for all digital pins.
SD_MODE	22	DI	Mode selection input. When tied low, the internal ramp generator generates step pulses. When tied high, the STEP/DIR inputs control the driver. SD_MODE=0 and SPI_MODE=0 enable UART operation.
SPI_MODE	23	DI (pd)	Mode selection input. When tied low with SD_MODE=1, the chip is in standalone mode and pins have their CFG functions. When tied high, the SPI interface is enabled. Integrated pull down resistor.
ENCB_DCEN_CFG4	24	DI (pd)	Encoder B-channel input (when using internal ramp generator) or dcStep enable input (SD_MODE=1, SPI_MODE=1) – leave open or tie to GND for normal operation in this mode (no dcStep). Configuration input (SPI_MODE=0)

Pin	Pin	Type	Function
ENCA_DCIN_CFG5	25	DI (pd)	Encoder A-channel input (when using internal ramp generator) or dcStep gating input for axis synchronization (SD_MODE=1, SPI_MODE=1) or Configuration input (SPI_MODE=0)
GNDD	26		Digital GND. Connect to GND plane near pin.
ENCN_DCO_CFG6	27	DIO	Encoder N-channel input (SD_MODE=0) or dcStep ready output (SD_MODE=1). With SD_MODE=0, pull to GND or VCC_IO, if the pin is not used.
DIAG0_SWN	28	DIO (pu+ pd)	Diagnostics output DIAG0. Interrupt or STEP output for motion controller (SD_MODE=0, SPI_MODE=1). Use external pullup resistor with 47k or less in open drain mode. Single wire I/O (negative) (only with SD_MODE=0 and SPI_MODE=0)
DIAG1_SWP	29	DIO (pd)	Diagnostics output DIAG1. Position compare or DIR output for motion controller (SD_MODE=0, SPI_MODE=1). Use external pullup resistor with 47k or less in open drain mode. Single wire I/O (positive) (only with SD_MODE=0 and SPI_MODE=0)
DRV_ENN	31	DI	Enable input. The power stage becomes switched off (all motor outputs floating) when this pin is driven high.
VCC	32		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. When using an external supply, make sure, that VCC comes up before or in parallel to 5VOUT or VCC_IO, whichever comes up later!
CPO	33		Charge pump capacitor output.
CPI	34		Charge pump capacitor input. Tie to CPO using 22nF 100V capacitor.
VS	35		Motor supply voltage. Provide filtering capacity near pin with short loop to GND plane. Must be tied to the positive bridge supply voltage.
VCP	37		Charge pump voltage. Tie to VS using 100nF capacitor.
VM	VM		Motor supply voltage and common cooling terminal for all HS MOSFETs. Connect to identical potential as VS.
OA2	OA2		Motor driver output and cooling terminal for LS MOSFET.
RSA	RSA		Sense resistor connection.
OA1	OA1		Motor driver output and cooling terminal for LS MOSFET.
CA1	CA1		Bootstrap capacitor positive connection.
CB2	CB2		Bootstrap capacitor positive connection.
OB2	OB2		Motor driver output and cooling terminal for LS MOSFET.
RSB	RSB		Sense resistor connection.
OB1	OB1		Motor driver output and cooling terminal for LS MOSFET.
NC1-NC4	NC1-NC4		Do not connect. Internal low side gate. May be left out in PCB footprint.

Pin	Pin	Type	Function
GNDPAD	GNDPAD		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

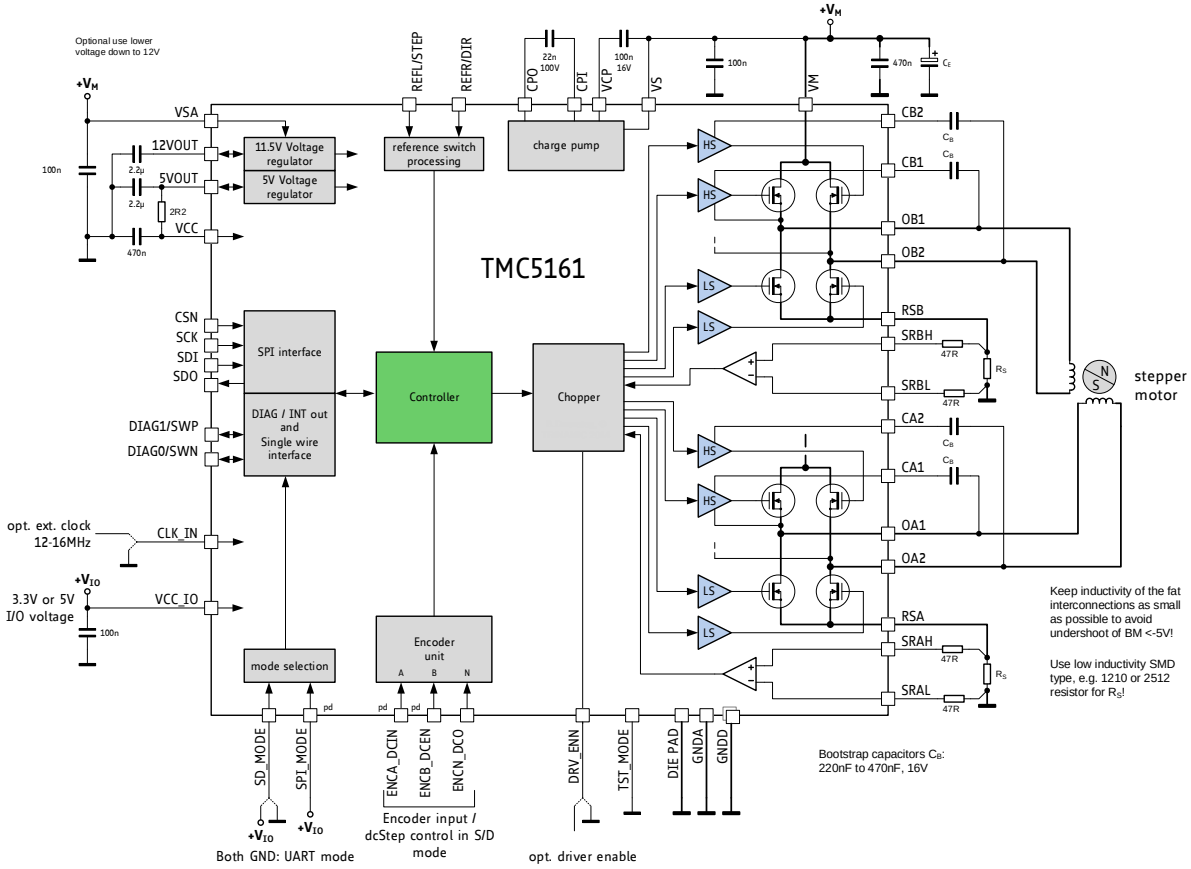


Figure 3.1 Standard application circuit

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 9 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. In order 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 3.2).

Basic layout hints
Place sense resistors and all filter capacitors as close as possible to the power MOSFETs. Place the TMC5161 near to the MOSFETs and use short interconnection lines in order 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.

Attention
In case VSA is supplied by a different voltage source, make sure that VSA does not drop out during motor operation.

3.2 External Gate Voltage Regulator

At high supply voltages like 36V, the internal gate voltage regulator and the internal 5V regulator have considerable power dissipation, especially with 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 TMC5161. Figure 3.2 shows the required connection. The internal gate voltage regulator becomes disabled in this constellation. 12V +/-1V are recommended for best results.

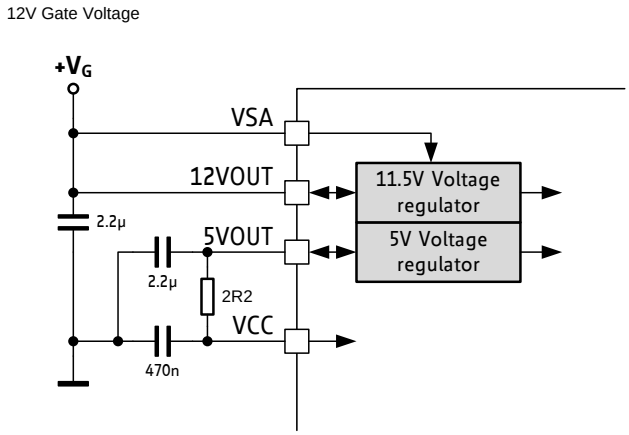


Figure 3.2 External gate voltage supply

3.3 MOSFETs and Slope Control

The TMC5161 integrates a discrete MOSFET power stage in order to yield a complete driver. The MOSFET driver stage allows adaptation of parameters like gate driver current and blank time, in order to optimally fit the driver with the MOSFETs for the given application. The tiny internal driver stage operates at 10ns slope with minimum gate driver current setting, which is absolutely sufficient for minimum power dissipation. Due to the fast slopes, minimum BBM time setting is sufficient.

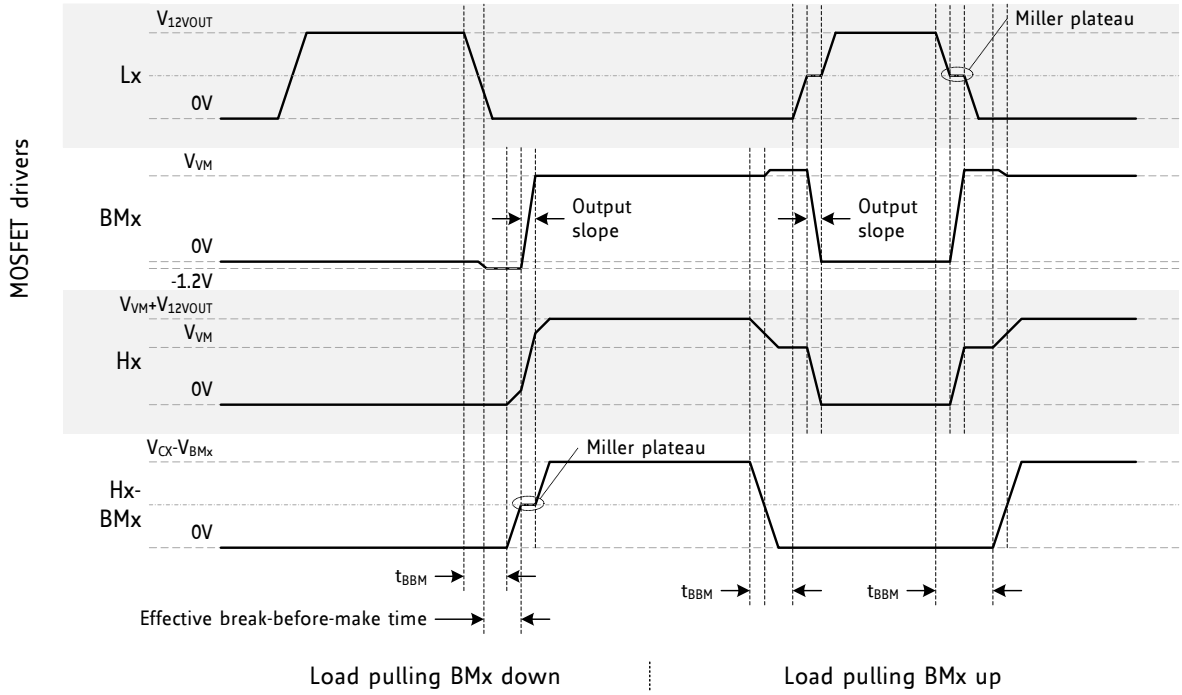


Figure 3.3 Slopes, Miller plateau and blank time

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. As the TMC5161 switches very fast, a setting of 0 is sufficient.	0…8	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 to 2 recommended. As the TMC5161 switches very fast, a setting of 0 is sufficient. However, there is only negligible difference with settings 2 or 4.	0…15	0: off, use *BBMTIME* *Reset Default: OTP 4 or 2*
DRV_ STRENGTH	Selection of gate driver current. Higher settings give faster slopes. 0 recommended.	0…2	*Reset Default = 2* 0 recommended.
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

3.4 Driver Protection and EME Circuitry

Electromagnetic emission is an important optimization area, to keep cost for shielding low. Further, some applications have to cope with ESD events caused by motor operation or external influence. Despite ESD circuitry within the driver chips, ESD events occurring during operation can cause a reset or even a destruction of the motor driver, depending on their energy. Especially plastic housings and belt drive systems tend to cause ESD events of several kV. It is best practice to avoid ESD events by attaching all conductive parts, especially the motors themselves to PCB ground, or to apply electrically conductive plastic parts. In addition, the driver can be protected up to a certain degree against ESD events or live plugging / pulling the motor, which also causes high voltages and high currents into the motor connector terminals. A simple scheme uses capacitors at the driver outputs to reduce the dV/dt caused by MOSFET diode recovery, and additionally caused by external ESD events. Larger capacitors will bring more benefit concerning ESD suppression, but cause additional current flow in each chopper cycle, and thus increase driver power dissipation, especially at high supply voltages. The values shown are example values – they might be varied between 100pF and 1nF. The capacitors dampen high frequency resulting from MOSFET switching and noise injected from digital parts of the application PCB circuitry and thus reduce electromagnetic emission. A more elaborate scheme uses LC filters to de-couple the driver outputs from the motor connector. Varistors in between of the coil terminals eliminate coil overvoltage caused by live plugging. Optionally protect all outputs by a varistor against ESD voltage.

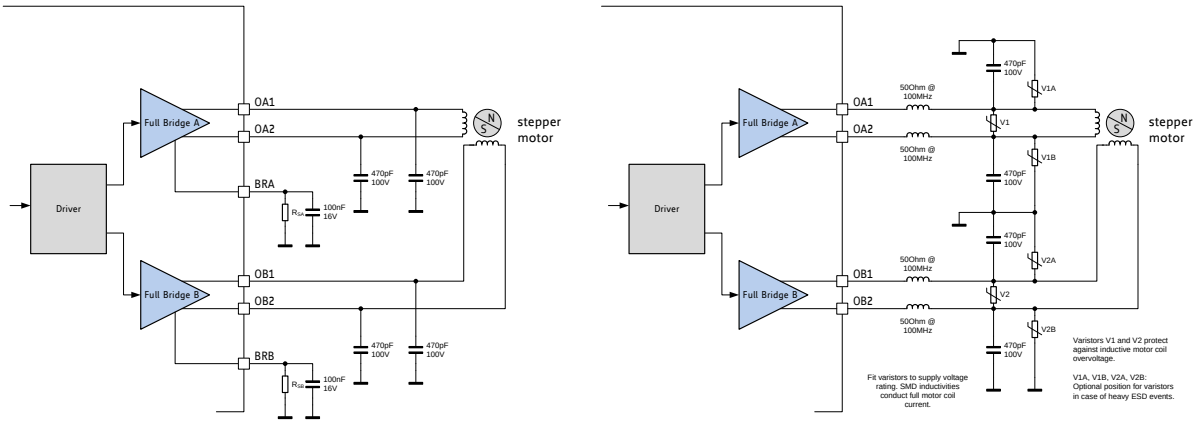


Figure 3.4 Simple ESD enhancement and more elaborate motor output protection

4 SPI Interface

4.1 SPI Datagram Structure

The TMC5161 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.

SPI DATAGRAM STRUCTURE

MSB (transmitted first)	40 bit	LSB (transmitted last)
39 …		… 0
→ 8 bit address **← 8 bit SPI status**	**← → 32 bit data**	
39 … 32	31 … 0	
→ to TMC5161 RW + 7 bit address ← from TMC5161 8 bit SPI status	8 bit data	8 bit data	8 bit data	8 bit data
39 / 38 … 32	31 … 24	23 … 16	15 … 8	7 … 0

w	38…32							31…28				27…24				23…20				19…16				15…12				11…8				7…4				3…0			
39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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 TMC5161, 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 (*XACTUAL*) with the address 0x21, the address byte has to be set to 0x21 in the access preceding the read access. For a write access to the register (*VACTUAL*), the address byte has to be set to 0x80 + 0x22 = 0xA2. For read access, the data bit might have any value (-). So, one can set them to 0.

action	data sent to TMC5161	data received from TMC5161
read *XACTUAL*	→ 0x2100000000	← 0xSS & unused data
read *XACTUAL*	→ 0x2100000000	← 0xSS & XACTUAL
write *VMAX*:= 0x00ABCDEF	→ 0xA700ABCDEF	← 0xSS & XACTUAL
write *VMAX*:= 0x00123456	→ 0xA700123456	← 0xSS00ABCDEF

*)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	*status_stop_r*	*RAMP_STAT*[1] – 1: Signals stop right switch status (motion controller only)
6	*status_stop_l*	*RAMP_STAT*[0] – 1: Signals stop left switch status (motion controller only)
5	*position_reached*	*RAMP_STAT*[9] – 1: Signals target position reached (motion controller only)
4	*velocity_reached*	*RAMP_STAT*[8] – 1: Signals target velocity reached (motion controller only)
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 by reading *GSTAT*)
0	*reset_flag*	*GSTAT*[0] – 1: Signals, that a reset has occurred (clear by reading *GSTAT*)

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 TMC5161 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 TMC5161.

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 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.

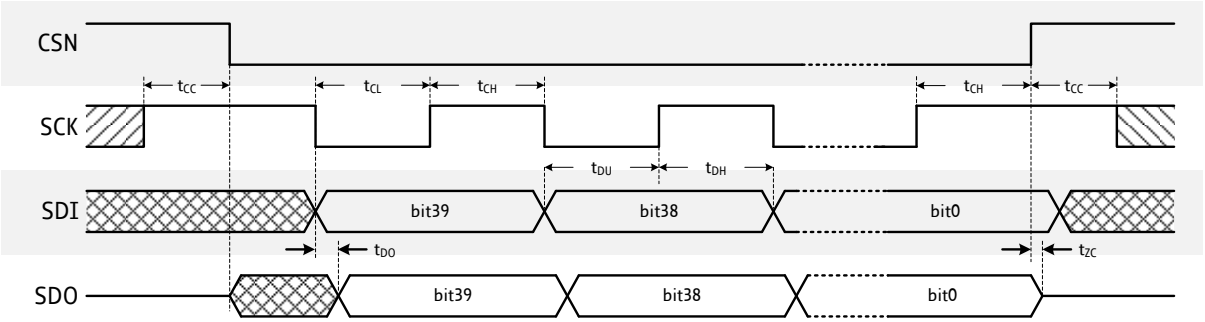


Figure 4.1 SPI timing

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 UART Single Wire Interface

The UART single wire interface allows the control of the TMC5161 with any microcontroller UART. It shares transmit and receive line like an RS485 based interface. Data transmission is secured using a cyclic redundancy check, so that increased interface distances (e.g. over cables between two PCBs) can be bridged without the danger of wrong or missed commands even in the event of electro-magnetic disturbance. The automatic baud rate detection and an advanced addressing scheme make this interface easy and flexible to use.

5.1 Datagram Structure

5.1.1 Write Access

UART WRITE ACCESS DATAGRAM STRUCTURE																			
each byte is LSB…MSB, highest byte transmitted first																			
0 … 63																			
sync + reserved								**8 bit slave address**			**RW + 7 bit register addr.**			**32 bit data**			**CRC**		
0…7								8…15			16…23			24…55			56…63		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*			register address		1	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	55	56	…	63

A sync nibble precedes each transmission to and from the TMC5161 and is embedded into the first transmitted byte, followed by an addressing byte. Each transmission allows a synchronization of the internal baud rate divider to the master clock. The actual baud rate is adapted and variations of the internal clock frequency are compensated. Thus, the baud rate can be freely chosen within the valid range. Each transmitted byte starts with a start bit (logic 0, low level on SWP) and ends with a stop bit (logic 1, high level on SWP). The bit time is calculated by measuring the time from the beginning of start bit (1 to 0 transition) to the end of the sync frame (1 to 0 transition from bit 2 to bit 3). All data is transmitted byte wise. The 32 bit data words are transmitted with the highest byte first.

A minimum baud rate of 9000 baud is permissible, assuming 20 MHz clock (worst case for low baud rate). Maximum baud rate is f_{CLK}/16 due to the required stability of the baud clock.

The slave address is determined by the register *SLAVEADDR*. If the external address pin NEXTADDR is set, the slave address becomes incremented by one.

The communication becomes reset if a pause time of longer than 63 bit times between the start bits of two successive bytes occurs. This timing is based on the last correctly received datagram. In this case, the transmission needs to be restarted after a failure recovery time of minimum 12 bit times of bus idle time. This scheme allows the master to reset communication in case of transmission errors. Any pulse on an idle data line below 16 clock cycles will be treated as a glitch and leads to a timeout of 12 bit times, for which the data line must be idle. Other errors like wrong CRC are also treated the same way. This allows a safe re-synchronization of the transmission after any error conditions. Remark, that due to this mechanism an abrupt reduction of the baud rate to less than 15 percent of the previous value is not possible.

Each accepted write datagram becomes acknowledged by the receiver by incrementing an internal cyclic datagram counter (8 bit). Reading out the datagram counter allows the master to check the success of an initialization sequence or single write accesses. Read accesses do not modify the counter.

5.1.2 Read Access

UART READ ACCESS REQUEST DATAGRAM STRUCTURE																					
each byte is LSB...MSB, highest byte transmitted first																					
sync + reserved								**8 bit slave address**				**RW + 7 bit register address**				**CRC**					
0...7								8...15				16...23				24...31					
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*				register address			0	CRC					
0	1	2	3	4	5	6	7	8	...		15	16	...		23	24	...		31		

The read access request datagram structure is identical to the write access datagram structure, but uses a lower number of user bits. Its function is the addressing of the slave and the transmission of the desired register address for the read access. The TMC5161 responds with the same baud rate as the master uses for the read request.

In order to ensure a clean bus transition from the master to the slave, the TMC5161 does not immediately send the reply to a read access, but it uses a programmable delay time after which the first reply byte becomes sent following a read request. This delay time can be set in multiples of eight bit times using *SENDDELAY* time setting (default=8 bit times) according to the needs of the master. In a multi-slave system, set *SENDDELAY* to min. 2 for all slaves. Otherwise a non-addressed slaves might detect a transmission error upon read access to a different slave.

UART READ ACCESS REPLY DATAGRAM STRUCTURE																					
each byte is LSB...MSB, highest byte transmitted first																					
0 63																					
sync + reserved								**8 bit slave address**				**RW + 7 bit register addr.**				**32 bit data**				**CRC**	
0...7								8...15				16...23				24...55				56...63	
1	0	1	0	reserved (0)				0xFF				register address			0	data bytes 3, 2, 1, 0 (high to low byte)				CRC	
0	1	2	3	4	5	6	7	8	...		15	16	...		23	24	...		55	56	...	63

The read response is sent to the master using address code %1111. The transmitter becomes switched inactive four bit times after the last bit is sent.

Address %11111111 is reserved for read accesses going to the master. A slave cannot use this address.

5.2 CRC Calculation

An 8 bit CRC polynomial is used for checking both read and write access. It allows detection of up to eight single bit errors. The CRC8-ATM polynomial with an initial value of zero is applied LSB to MSB, including the sync- and addressing byte. The sync nibble is assumed to always be correct. The TMC5161 responds only to correctly transmitted datagrams containing its own slave address. It increases its datagram counter for each correctly received write access datagram.

$$CRC = x^8 + x^2 + x^1 + x^0$$

SERIAL CALCULATION EXAMPLE

```
CRC = (CRC << 1) OR (CRC.7 XOR CRC.1 XOR CRC.0 XOR [new incoming bit])
```

C-CODE EXAMPLE FOR CRC CALCULATION

```
void swuart_calcCRC(UCHAR* datagram, UCHAR datagramLength)
{
  int i,j;
  UCHAR* crc = datagram + (datagramLength-1); // CRC located in last byte of message
  UCHAR currentByte;

  *crc = 0;
  for (i=0; i<(datagramLength-1); i++) {      // Execute for all bytes of a message
    currentByte = datagram[i];                // Retrieve a byte to be sent from Array
    for (j=0; j<8; j++) {
      if ((*crc >> 7) ^ (currentByte&0x01))   // update CRC based result of XOR operation
      {
        *crc = (*crc << 1) ^ 0x07;
      }
      else
      {
        *crc = (*crc << 1);
      }
      currentByte = currentByte >> 1;
    } // for CRC bit
  } // for message byte
}
```

5.3 UART Signals

The UART interface on the TMC5161 comprises four signals:

TMC5161 UART INTERFACE SIGNALS	
SWP	Non-inverted data input and output
SWN	Inverted data input and output for use in differential transmission. Can be left open in a 5V IO voltage system. Tie to the half IO level voltage for best performance in a 3.3V single wire non-differential application.
SDI_CFG1 (NAI)	Address increment pin for chained sequential addressing scheme
SDO_CFG0 (NAO)	Next address output pin for chained sequential addressing scheme (reset default= high)

In UART mode (SPI_MODE low and SD_MODE low) the slave checks the single wire SWP and SWN for correctly received datagrams with its own address continuously. Both signals are switched as input during this time. It adapts to the baud rate based on the sync nibble, as described before. In case of a read access, it switches on its output drivers on SWP and SWN and sends its response using the same baud rate.

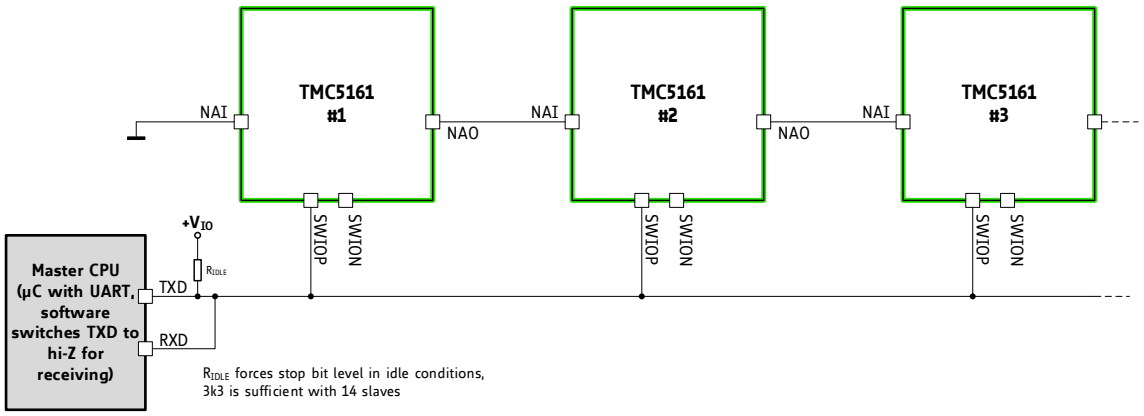
5.4 Addressing Multiple Slaves

Addressing one or two Slaves

If only one or two TMC5161 are addressed by a master using a single UART interface, a hardware address selection can be done by *setting the NAI pins of both devices to different levels*.

Addressing up to 255 Slaves

A different approach can address any number of devices by *using the input NAI as a selection pin*. Addressing up to 255 units is possible.



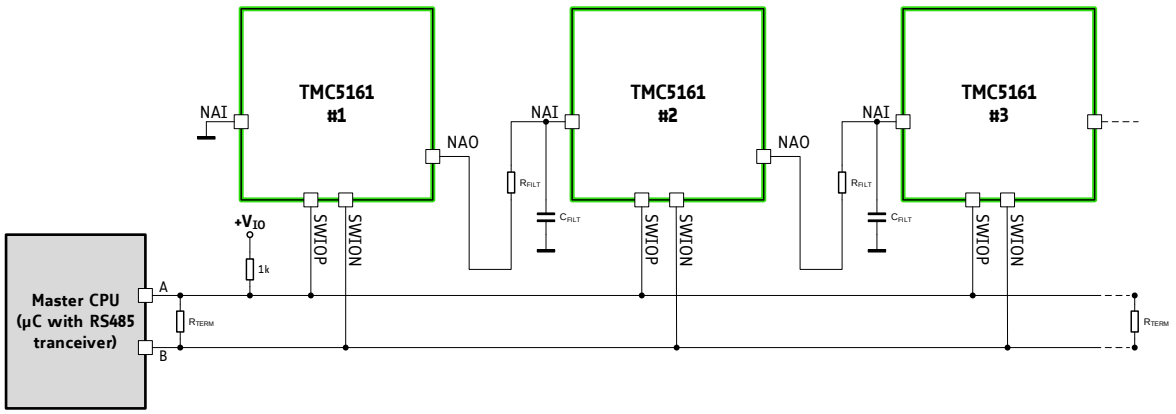
Example for addressing up to 255 TMC5161

Addressing phase 1:	address 0, NAO is high	address 1	address 1
Addressing phase 2:	program to address 254 & set NAO low	address 0, NAO is high	address 1
Addressing phase 3:	address 254	program to address 253 & set NAO low	address 0
Addressing phase 4:	address 254	address 253	program to address 252 & set NAO low
Addressing phase *X*:	*continue procedure*		

Figure 5.1 Addressing multiple TMC5160 / TMC5161 via single wire interface using chaining

Proceed as follows:

- Tie the NAI pin of your first TMC5161 to GND.
- Interconnect NAO output of the first TMC5161 to the next drivers NAI pin. Connect further drivers in the same fashion.
- Now, the first driver responds to address 0. Following drivers are set to address 1.
- Program the first driver to its dedicated slave address. Note: once a driver is initialized with its slave address, its NAO output, which is tied to the next drivers NAI has to be programmed to logic 0 in order to differentiate the next driver from all following devices.
- Now, the second driver is accessible and can get its slave address. Further units can be programmed to their slave addresses sequentially.



EXAMPLE FOR ADDRESSING UP TO 255 TMC5161			
Addressing phase 1:	address 0, NAO high	*address 1*	*address 1*
Addressing phase 2:	program to address 254 & set NAO low	address 0, NAO high	*address 1*
Addressing phase 3:	address 254	program to address 253 & set NAO low	address 0, NAO high
Addressing phase 4:	address 254	address 253	program to address 252 & set NAO low
Addressing phase *X*:	*continue procedure*		

Figure 5.2 Addressing TMC5160 / TMC5161 via differential interface, additional filtering for NAI

A different scheme (not shown) uses bus switches (like 74HC4066) to connect the bus to the next unit in the chain without using the NAI input. The bus switch can be controlled in the same fashion, using the NAO output to enable it (low level shall enable the bus switch). Once the bus switch is enabled it allows addressing the next bus segment. As bus switches add a certain resistance, the maximum number of nodes will be reduced.

It is possible to mix different styles of addressing in a system. For example, a system using two boards with each two TMC5161 can have both devices on a board with a different level on NEXTADDR, while the next board is chained using analog switches separating the bus until the drivers on the first board have been programmed.

6 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+C	Clear upon read

OVERVIEW REGISTER MAPPING

REGISTER	**DESCRIPTION**
General Configuration Registers	These registers contain - global configuration - global status flags - interface configuration - and I/O signal configuration
Ramp Generator Motion Control Register Set	This register set offers registers for - choosing a ramp mode - choosing velocities - homing - acceleration and deceleration - target positioning - reference switch and stallGuard2 event configuration - ramp and reference switch status
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
Encoder Register Set	The encoder register set offers all registers needed for proper ABN encoder 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

6.1 General Configuration Registers

GENERAL CONFIGURATION REGISTERS (0X00...0X0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
RW	0x00	17	*GCONF*	**Bit**	***GCONF* – Global configuration flags**
				0	*recalibrate* 1: Zero crossing recalibration during driver disable (via 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* (only with SD_MODE=1) 1: Enable DIAG0 active on driver errors: Over temperature (*ot*), short to GND (*s2g*), undervoltage chargepump (*uv_cp*) DIAG0 always shows the reset-status, i.e. is active low during reset condition.
				6	*diag0_otpw* (only with SD_MODE=1) 1: Enable DIAG0 active on driver over temperature prewarning (*otpw*)
				7	*diag0_stall* (with SD_MODE=1) 1: Enable DIAG0 active on motor stall (set *TCOOLTHRS* before using this feature) *diag0_step* (with SD_MODE=0) 0: DIAG0 outputs interrupt signal 1: Enable DIAG0 as STEP output (dual edge triggered steps) for external STEP/DIR driver
				8	*diag1_stall* (with SD_MODE=1) 1: Enable DIAG1 active on motor stall (set *TCOOLTHRS* before using this feature) *diag1_dir* (with SD_MODE=0) 0: DIAG1 outputs position compare signal 1: Enable DIAG1 as DIR output for external STEP/DIR driver
				9	*diag1_index* (only with SD_MODE=1) 1: Enable DIAG1 active on index position (microstep look up table position 0)
				10	*diag1_onstate* (only with SD_MODE=1) 1: Enable DIAG1 active when chopper is on (for the coil which is in the second half of the fullstep)
				11	*diag1_steps_skipped* (only with SD_MODE=1) 1: Enable output toggle when steps are skipped in dcStep mode (increment of *LOST_STEPS*). Do not enable in conjunction with other DIAG1 options.

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
				12	*diag0_int_pushpull* 0: SWN_DIAG0 is open collector output (active low) 1: Enable SWN_DIAG0 push pull output (active high)
				13	*diag1_poscomp_pushpull* 0: SWP_DIAG1 is open collector output (active low) 1: Enable SWP_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
				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 *XTARGET* (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 ENCN_DCO. *IHOLD*[1..0] selects the function of ENCN_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 since the last read access to *GSTAT*. 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 since the last read access. 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	0x02	8	*IFCNT*	Interface transmission counter. This register becomes incremented with each successful UART interface write access. It can be read out to check the serial transmission for lost data. Read accesses do not change the content. Disabled in SPI operation. The counter wraps around from 255 to 0.	

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
W	0x03	8 + 4	*SLAVECONF*	**Bit**	***SLAVECONF***
				7..0	*SLAVEADDR*: These eight bits set the address of unit for the UART interface. The address becomes incremented by one when the external address pin NEXTADDR is active. Range: 0-253 (254 cannot be incremented), *default=0*
				11..8	*SENDDELAY*: 0, 1: 8 bit times (not allowed with multiple slaves) 2, 3: 3*8 bit times 4, 5: 5*8 bit times 6, 7: 7*8 bit times 8, 9: 9*8 bit times 10, 11: 11*8 bit times 12, 13: 13*8 bit times 14, 15: 15*8 bit times
R	0x04	8 + 8	*IOIN*	**Bit**	***INPUT***
					Reads the state of all input pins available
				0	REFL_STEP
				1	REFR_DIR
				2	ENCB_DCEN_CFG4
				3	ENCA_DCIN_CFG5
				4	DRV_ENN
				5	ENC_N_DCO_CFG6
				6	SD_MODE (1=External step and dir source)
				7	SWCOMP_IN (Shows voltage difference of SWN and SWP. Bring DIAG outputs to high level with pushpull disabled to test the comparator.)
				31..24	*VERSION*: 0x30=first version of the IC Identical numbers mean full digital compatibility.
W	0x04	1	*OUTPUT*	**Bit**	***OUTPUT***
					Sets the IO output pin polarity in UART mode
				0	In UART mode, SDO_CFG0 is an output. This bit programs the output polarity of this pin. Its main purpose it to use SDO_CFG0 as NAO next address output signal for chain addressing of multiple ICs. *Hint:* Reset Value is 1 for use as NAO to next IC in single wire chain
W	0x05	32	*X_COMPARE*	Position comparison register for motion controller position strobe. The Position pulse is available on output SWP_DIAG1. *XACTUAL = X_COMPARE*: - Output signal PP (position pulse) becomes high. It returns to a low state, if the positions mismatch.	
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

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
				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) **10 recommended for normal operation** *Hint:* Settings from 1 to 3 will trigger during normal operation due to voltage drop on sense resistor. *(Reset Default: 12 via OTP)*
				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) **6 to 10 recommended** *(Reset Default: 12 via OTP)*
				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)*
W	0x0A	22	*DRV_CONF*	**Bit**	**DRV_CONF**
				4..0	*BBMTIME:* Break-Before make delay 0=shortest (100ns) … 16 (200ns) … 24=longest (375ns) >24 not recommended, use *BBMCLKS* instead *Hint:* 0 recommended due to fast switching MOSFETs *(Reset Default = 0)*

GENERAL CONFIGURATION REGISTERS (0x00…0x0F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
				11..8	*BBMCLKS*: 0..15: Digital BBM time in clock cycles (typ. 83ns). The longer setting rules (*BBMTIME* vs. *BBMCLKS*). *(Reset Default: 2 via OTP)* *Hint:* 2, or down to 0 recommended due to fast switching MOSFETs
				17..16	*OTSELECT*: Selection of over temperature level for bridge disable, switch on after cool down to 120°C / OTPW level. 00: 150°C (not recommended – MOSFET might overheat) 01: 143°C 10: 136°C ← R*ecommended* 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: Normal slope ← *Recommended* 01: Normal+TC (medium above OTPW level) 10: Fast slope *(Reset Default = %10)*
				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)*
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)

6.1.1 *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 ← *Default – cannot be changed!*
5	*otp0.5*	*otp_S2_LEVEL*	Reset default for *Short detection Levels*: 0: *S2G_LEVEL = S2VS_LEVEL = 6* 1: *S2G_LEVEL = S2VS_LEVEL = 12* ← *Default - cannot be changed!*
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*		

6.2 Velocity Dependent Driver Feature Control Register Set

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0X10...0X1F)					
R/W	**Addr**	**n**	***Register***	**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_OFFS_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. When working with the motion controller, the measured TSTEP for a given velocity V is in the range $(2^{24} / V) \leq TSTEP \leq 2^{24} / V - 1$. 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.	

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0X10...0X1F)				
R/W	**Addr**	**n**	***Register***	**Description */ bit names***
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
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 stop on stall function (enable with *sg_stop* when using internal motion controller) and the stall output signal become 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* - Stop on stall is enabled, if configured - 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). - chopSync2 is switched off (*SYNC*=0) - If *vhighfs* is set, the motor operates in fullstep mode and the stall detection becomes switched over to dcStep stall detection.

Microstep velocity time reference t for velocities: TSTEP = f_{CLK} / f_{STEP}

6.3 Ramp Generator Registers

6.3.1 Ramp Generator Motion Control Register Set

RAMP GENERATOR MOTION CONTROL REGISTER SET (0X20...0X2D)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	**Range [Unit]**
RW	0x20	2	*RAMPMODE*	*RAMPMODE:* 0: Positioning mode (using all A, D and V parameters) 1: Velocity mode to positive *VMAX* (using *AMAX* acceleration) 2: Velocity mode to negative *VMAX* (using *AMAX* acceleration) 3: Hold mode (velocity remains unchanged, unless stop event occurs)	0...3
RW	0x21	32	*XACTUAL*	Actual motor position (signed) *Hint:* This value normally should only be modified, when homing the drive. In positioning mode, modifying the register content will start a motion.	-2^31... +(2^31)-1
R	0x22	24	*VACTUAL*	Actual motor velocity from ramp generator (signed) The sign matches the motion direction. A negative sign means motion to lower *XACTUAL*.	+-(2^23)-1 [µsteps / t]
W	0x23	18	*VSTART*	Motor start velocity (unsigned) Set *VSTOP ≥ VSTART!*	0...(2^18)-1 [µsteps / t]
W	0x24	16	*A1*	First acceleration between *VSTART* and *V1* (unsigned)	0...(2^16)-1 [µsteps / ta²]
W	0x25	20	*V1*	First acceleration / deceleration phase threshold velocity (unsigned) 0: Disables *A1* and *D1* phase, use *AMAX*, *DMAX* only	0...(2^20)-1 [µsteps / t]
W	0x26	16	*AMAX*	Second acceleration between *V1* and *VMAX* (unsigned) This is the acceleration and deceleration value for velocity mode.	0...(2^16)-1 [µsteps / ta²]
W	0x27	23	*VMAX*	Motion ramp target velocity (for positioning ensure *VMAX ≥ VSTART*) (unsigned) This is the target velocity in velocity mode. It can be changed any time during a motion.	0...(2^23)-512 [µsteps / t]
W	0x28	16	*DMAX*	Deceleration between *VMAX* and *V1* (unsigned)	0...(2^16)-1 [µsteps / ta²]
W	0x2A	16	*D1*	Deceleration between *V1* and *VSTOP* (unsigned) *Attention: Do not set 0 in positioning mode, even if V1=0!*	1...(2^16)-1 [µsteps / ta²]

RAMP GENERATOR MOTION CONTROL REGISTER SET (0X20...0X2D)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	**Range [Unit]**
W	0x2B	18	*VSTOP*	Motor stop velocity (unsigned) *Hint:* Set VSTOP ≥ VSTART to allow positioning for short distances *Attention: Do not set 0 in positioning mode, minimum 10 recommend!*	1...(2^18)-1 [µsteps / t] Reset Default=1
W	0x2C	16	*TZEROWAIT*	Defines the waiting time after ramping down to zero velocity before next movement or direction inversion can start. Time range is about 0 to 2 seconds. This setting avoids excess acceleration e.g. from *VSTOP* to -*VSTART*.	0...(2^16)-1 * 512 t_{CLK}
RW	0x2D	32	*XTARGET*	Target position for ramp mode (signed). Write a new target position to this register in order to activate the ramp generator positioning in *RAMPMODE*=0. Initialize all velocity, acceleration and deceleration parameters before. *Hint:* The position is allowed to wrap around, thus, *XTARGET* value optionally can be treated as an unsigned number. *Hint:* The maximum possible displacement is +/-((2^31)-1). *Hint:* When increasing *V1*, *D1 or DMAX during* a motion, rewrite *XTARGET* afterwards in order to trigger a second acceleration phase, if desired.	-2^31... +(2^31)-1

6.3.2 Ramp Generator Driver Feature Control Register Set

RAMP GENERATOR DRIVER FEATURE CONTROL REGISTER SET (0X30...0X36)				
R/W	**Addr**	**n**	***Register***	**Description */ bit names***
W	0x33	23	*VDCMIN*	Automatic commutation dcStep becomes enabled above velocity *VDCMIN* (unsigned) (only when using internal ramp generator, not for STEP/DIR interface – in STEP/DIR mode, dcStep becomes enabled by the external signal DCEN) In this mode, the actual position is determined by the sensorless motor commutation and becomes fed back to *XACTUAL*. In case the motor becomes heavily loaded, *VDCMIN* also is used as the minimum step velocity. Activate stop on stall (*sg_stop*) to detect step loss. 0: Disable, dcStep off *|VACT|* ≥ *VDCMIN* ≥ 256: - Triggers the same actions as exceeding *THIGH* setting. - Switches on automatic commutation dcStep *Hint:* Also set *DCCTRL* parameters in order to operate dcStep. (Only bits 22... 8 are used for value and for comparison)
RW	0x34	12	***SW_MODE***	Switch mode configuration *See separate table!*
R+ WC	0x35	14	***RAMP_STAT***	Ramp status and switch event status *See separate table!*
R	0x36	32	*XLATCH*	Ramp generator latch position, latches *XACTUAL* upon a programmable switch event (see *SW_MODE*). *Hint:* The encoder position can be latched to *ENC_LATCH* together with *XLATCH* to allow consistency checks.

Time reference t for velocities: t = 2^24 / f_{CLK}
Time reference ta^2 for accelerations: ta^2 = 2^41 / $(f_{CLK})^2$

6.3.2.1 SW_MODE – Reference Switch & stallGuard2 Event Configuration Register

0x34: *SW_MODE* – REFERENCE SWITCH AND STALLGUARD2 EVENT CONFIGURATION REGISTER		
Bit	**Name**	**Comment**
11	*en_softstop*	0: Hard stop 1: Soft stop The soft stop mode always uses the deceleration ramp settings *DMAX*, *V1*, *D1*, *VSTOP* and *TZEROWAIT* for stopping the motor. A stop occurs when the velocity sign matches the reference switch position (REFL for negative velocities, REFR for positive velocities) and the respective switch stop function is enabled. A hard stop also uses *TZEROWAIT* before the motor becomes released. *Attention: Do not use soft stop in combination with stallGuard2. Use soft stop for stealthChop operation at high velocity. In this case, hard stop must be avoided, as it could result in severe overcurrent.*
10	*sg_stop*	1: Enable stop by stallGuard2 (also available in dcStep mode). Disable to release motor after stop event. Program *TCOOLTHRS* for velocity threshold. *Hint:* Do not enable during motor spin-up, wait until the motor velocity exceeds a certain value, where stallGuard2 delivers a stable result. This velocity threshold should be programmed using TCOOLTHRS.
9	*en_latch_encoder*	1: Latch encoder position to *ENC_LATCH* upon reference switch event.
8	*latch_r_inactive*	1: Activates latching of the position to *XLATCH* upon an inactive going edge on the right reference switch input REFR. The active level is defined by *pol_stop_r*.
7	*latch_r_active*	1: Activates latching of the position to *XLATCH* upon an active going edge on the right reference switch input REFR. *Hint:* Activate *latch_r_active* to detect any spurious stop event by reading *status_latch_r*.
6	*latch_l_inactive*	1: Activates latching of the position to *XLATCH* upon an inactive going edge on the left reference switch input REFL. The active level is defined by *pol_stop_l*.
5	*latch_l_active*	1: Activates latching of the position to *XLATCH* upon an active going edge on the left reference switch input REFL. *Hint:* Activate *latch_l_active* to detect any spurious stop event by reading *status_latch_l*.
4	*swap_lr*	1: Swap the left and the right reference switch input REFL and REFR
3	*pol_stop_r*	Sets the active polarity of the right reference switch input 0=non-inverted, high active: a high level on REFR stops the motor 1=inverted, low active: a low level on REFR stops the motor
2	*pol_stop_l*	Sets the active polarity of the left reference switch input 0=non-inverted, high active: a high level on REFL stops the motor 1=inverted, low active: a low level on REFL stops the motor
1	*stop_r_enable*	1: Enables automatic motor stop during active right reference switch input *Hint:* The motor restarts in case the stop switch becomes released.
0	*stop_l_enable*	1: Enables automatic motor stop during active left reference switch input *Hint:* The motor restarts in case the stop switch becomes released.

6.3.2.2 RAMP_STAT – Ramp & Reference Switch Status Register

0x35: *RAMP_STAT* – RAMP AND REFERENCE SWITCH STATUS REGISTER			
R/W	**Bit**	**Name**	**Comment**
R	13	*status_sg*	1: Signals an active stallGuard2 input from the coolStep driver or from the dcStep unit, if enabled. *Hint:* When polling this flag, stall events may be missed – activate *sg_stop* to be sure not to miss the stall event.
R+ WC	12	*second_move*	1: Signals that the automatic ramp required moving back in the opposite direction, e.g. due to on-the-fly parameter change (Write '1' to clear)
R	11	*t_zerowait_ active*	1: Signals, that *TZEROWAIT* is active after a motor stop. During this time, the motor is in standstill.
R	10	*vzero*	1: Signals, that the actual velocity is 0.
R	9	*position_ reached*	1: Signals, that the target position is reached. This flag becomes set while *XACTUAL* and *XTARGET* match.
R	8	*velocity_ reached*	1: Signals, that the target velocity is reached. This flag becomes set while *VACTUAL* and *VMAX* match.
R+ WC	7	*event_pos_ reached*	1: Signals, that the target position has been reached (*position_reached* becoming active). (Write '1' to clear flag and interrupt condition) This bit is ORed to the *interrupt output* signal.
R+ WC	6	*event_stop_ sg*	1: Signals an active StallGuard2 stop event. Reading the register will clear the stall condition and the motor may re-start motion, unless the motion controller has been stopped. (Write '1' to clear flag and interrupt condition) This bit is ORed to the *interrupt output* signal.
R	5	*event_stop_r*	1: Signals an active stop right condition due to stop switch. The stop condition and the interrupt condition can be removed by setting *RAMP_MODE* to hold mode or by commanding a move to the opposite direction. In *soft_stop* mode, the condition will remain active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor will continue motion. This bit is ORed to the *interrupt output* signal.
	4	*event_stop_l*	1: Signals an active stop left condition due to stop switch. The stop condition and the interrupt condition can be removed by setting *RAMP_MODE* to hold mode or by commanding a move to the opposite direction. In *soft_stop* mode, the condition will remain active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor will continue motion. This bit is ORed to the *interrupt output* signal.
R+ WC	3	*status_latch_r*	1: Latch right ready (enable position latching using *SW_MODE* settings *latch_r_active* or *latch_r_inactive*) (Write '1' to clear)
	2	*status_latch_l*	1: Latch left ready (enable position latching using *SW_MODE* settings *latch_l_active* or *latch_l_inactive*) (Write '1' to clear)
R	1	*status_stop_r*	Reference switch right status (1=active)
	0	*status_stop_l*	Reference switch left status (1=active)

6.4 Encoder Registers

ENCODER REGISTER SET (0X38...0X3C)					
R/W	**Addr**	**n**	***Register***	**Description *I bit names***	**Range [Unit]**
RW	0x38	11	***ENCMODE***	Encoder configuration and use of N channel *See separate table!*	
RW	0x39	32	*X_ENC*	Actual encoder position (signed)	-2^31... +(2^31)-1
W	0x3A	32	*ENC_CONST*	Accumulation constant (signed) 16 bit integer part, 16 bit fractional part *X_ENC* accumulates *+/- ENC_CONST /* (2^16**X_ENC*) (binary) or *+/-ENC_CONST /* (10^4**X_ENC*) (decimal) *ENCMODE* bit *enc_sel_decimal* switches between decimal and binary setting. Use the sign, to match rotation direction!	binary: ± [µsteps/2^16] ±(0 ... 32767.999847) decimal: ±(0.0 ... 32767.9999) *reset default = 1.0 (=65536)*
R+ WC	0x3B	2	*ENC_STATUS*	Encoder status information bit 0: *n_event* bit 1*: deviation_warn* 1: Event detected. To clear the status bit, write with a 1 bit at the corresponding position. Deviation_warn cannot be cleared while a warning still persists. Set *ENC_DEVIATION* zero to disable. Both bits are ORed to the *interrupt output* signal.	
R	0x3C	32	*ENC_LATCH*	Encoder position *X_ENC* latched on N event	
W	0x3D	20	*ENC_ DEVIATION*	Maximum number of steps deviation between encoder counter and XACTUAL for deviation warning Result in flag ENC_STATUS.*deviation_warn* 0=Function is off.	

6.4.1 *ENCMODE* – Encoder Register

0x38: *ENCMODE* – ENCODER REGISTER			
Bit	**Name**	**Comment**	
10	*enc_sel_decimal*	0	Encoder prescaler divisor binary mode: Counts *ENC_CONST(fractional part)* /65536
		1	Encoder prescaler divisor decimal mode: Counts in *ENC_CONST(fractional part)* /10000
9	*latch_x_act*	1: Also latch *XACTUAL* position together with *X_ENC*. Allows latching the ramp generator position upon an N channel event as selected by *pos_edge* and *neg_edge*.	
8	*clr_enc_x*	0	Upon N event, *X_ENC* becomes latched to *ENC_LATCH* only
		1	Latch and additionally clear encoder counter *X_ENC* at N-event
7	*neg_edge*	***n p***	**N channel event sensitivity**
6	*pos_edge*	0 0	N channel event is active during an active N event level
		0 1	N channel is valid upon active going N event
		1 0	N channel is valid upon inactive going N event
		1 1	N channel is valid upon active going and inactive going N event
5	*clr_once*	1: Latch or latch and clear *X_ENC* on the next N event following the write access	
4	*clr_cont*	1: Always latch or latch and clear *X_ENC* upon an N event (once per revolution, it is recommended to combine this setting with edge sensitive N event)	
3	*ignore_AB*	0	An N event occurs only when polarities given by *pol_N*, *pol_A* and *pol_B* match.
		1	Ignore A and B polarity for N channel event
2	*pol_N*	Defines active polarity of N (0=low active, 1=high active)	
1	*pol_B*	Required B polarity for an N channel event (0=neg., 1=pos.)	
0	*pol_A*	Required A polarity for an N channel event (0=neg., 1=pos.)	

6.5 Motor Driver Registers

MICROSTEPPING CONTROL REGISTER SET (0X60…0X6B)					
R/W	**Addr**	**n**	***Register***	**Description *I 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	32x 0 or 1 *reset default= sine wave table*
W	0x61 … 0x67	7 x 32	*MSLUT[1…7]* microstep table entries 32…255	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*	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_A*. *CUR_B* 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_A* (signed): Actual microstep current for motor phase A as read from MSLUT (not scaled by current) bit 24… 16: *CUR_B* (signed): Actual microstep current for motor phase B 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	22	***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. Only with SDMODE=1.		

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

6.5.1 *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			

6.5.2 *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
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*

0x6C: *CHOPCONF* – CHOPPER CONFIGURATION				
Bit	**Name**	**Function**	**Comment**	
10 9 8 7	*hend3* *hend2* *hend1* *hend0*	*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.
			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 5 4	*hstrt2* *hstrt1* *hstrt0*	*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
		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 2 1 0	*toff3* *toff2* *toff1* *toff0*	*TOFF* off time and driver enable	Off time setting controls duration of slow decay phase N_{CLK}= 12 + 32**TOFF* %0000: Driver disable, all bridges off %0001: 1 – use only with *TBL* ≥ 2 %0010 … %1111: 2 … 15	

6.5.3 *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*		

6.5.4 *PWMCONF* – Voltage PWM Mode stealthChop

0x70: *PWMCONF* – VOLTAGE MODE PWM STEALTHCHOP				
Bit	**Name**	**Function**		**Comment**
31 30 29 28	*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)
27 26 25 24	*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)
23	-	reserved		set to 0
22	-	reserved		set to 0
21 20	*freewheel1* *freewheel0*	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
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 in order 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 1.5 * *PWM_OFS_AUTO* < *PWM_SCALE_SUM* < 4* *PWM_OFS_AUTO* and *PWM_SCALE_SUM* < 255. 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*.
18	*pwm_ autoscale*	PWM automatic amplitude scaling	0	User defined feed forward PWM amplitude. The current settings *IRUN* and *IHOLD* have no influence! 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)*

0x70: *PWMCONF* – VOLTAGE MODE PWM STEALTHCHOP			
Bit	**Name**	**Function**	**Comment**
17 16	*pwm_freq1* *pwm_freq0*	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}
15 14 13 12 11 10 9 8	*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*.
7 6 5 4 3 2 1 0	*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*).

6.5.5 *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 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 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 stand still 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 in order 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			

7 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 absolutely 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.

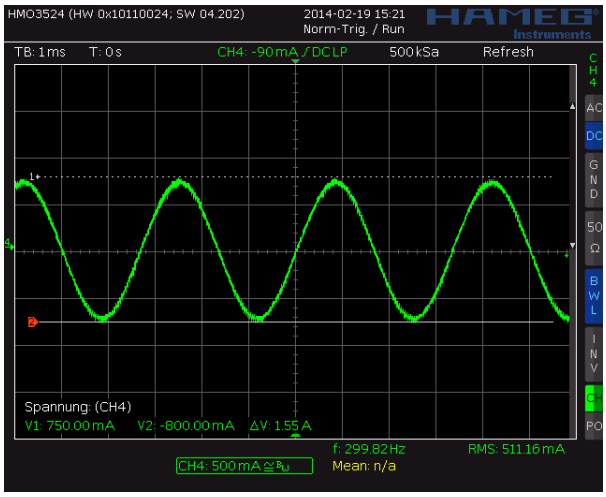


Figure 7.1 Motor coil sine wave current with stealthChop (measured with current probe)

7.1 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 7.2 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* < *PWM_SCALE_SUM* < 4 * *PWM_OFS_AUTO* - *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 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:
Successful completion of AT#1 tuning phase is not safely detected by the TMC5161. 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.

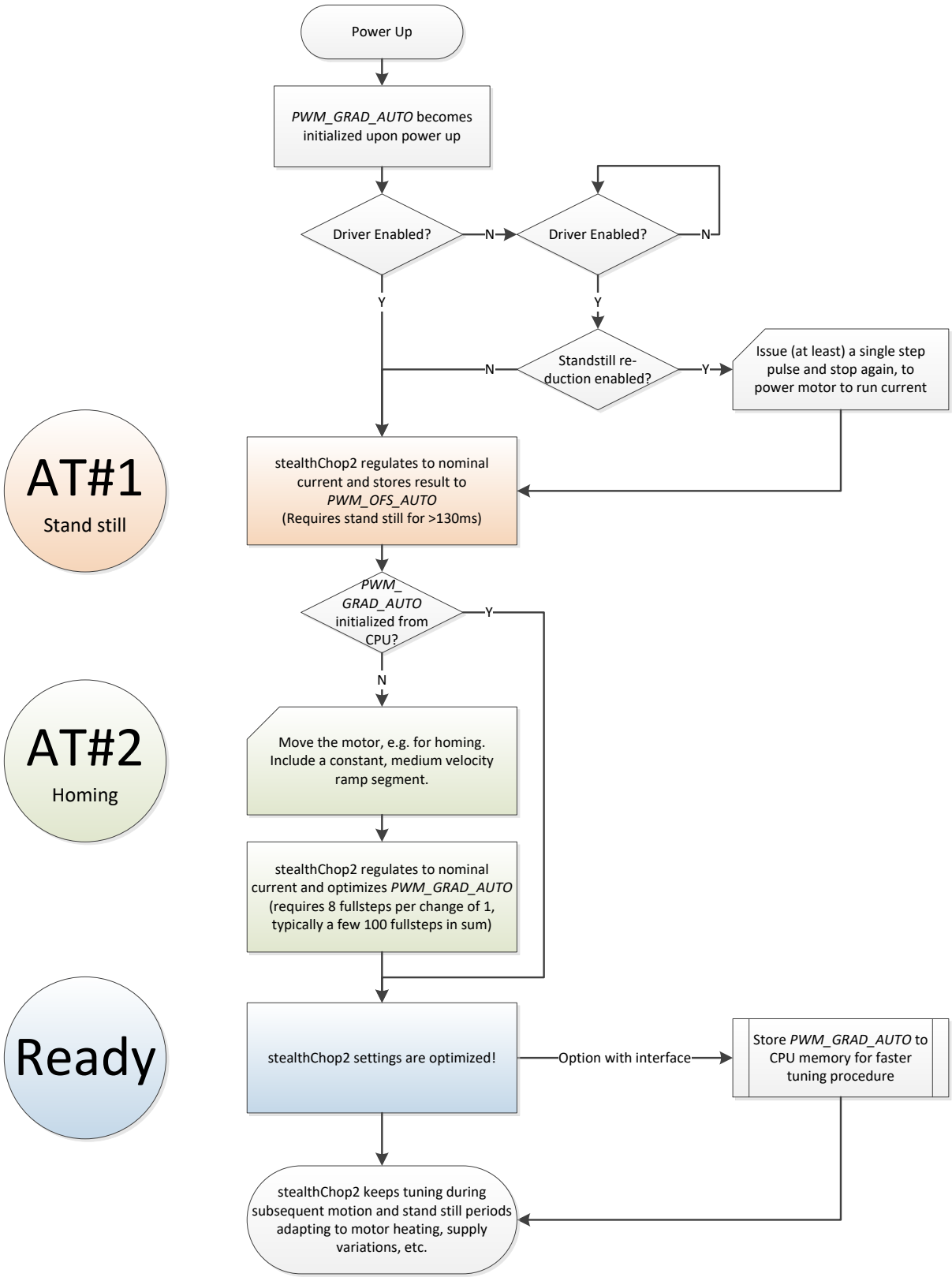


Figure 7.2 stealthChop2 automatic tuning procedure

Attention
Modifying *GLOBALSCALER* or VS voltage invalidates the result of the automatic tuning process. Motor current regulation cannot compensate significant changes until next AT#1 phase. Automatic tuning adapts to changed conditions whenever AT#1 and AT#2 conditions are fulfilled in the later operation.

7.2 stealthChop Options

In order 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 operate in automatic tuning mode, but use pre-tuned starting values. The parameters will be further adapted during operation.

Non-automatic mode (*pwm_autoscale*=0) should be taken into account 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 in order 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 7.1 Choice of PWM frequency – green / light green: recommended

7.3 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 in order 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 the motor target current (e.g. change of VREF). 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 7.3 and Figure 7.4). 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 7.5.

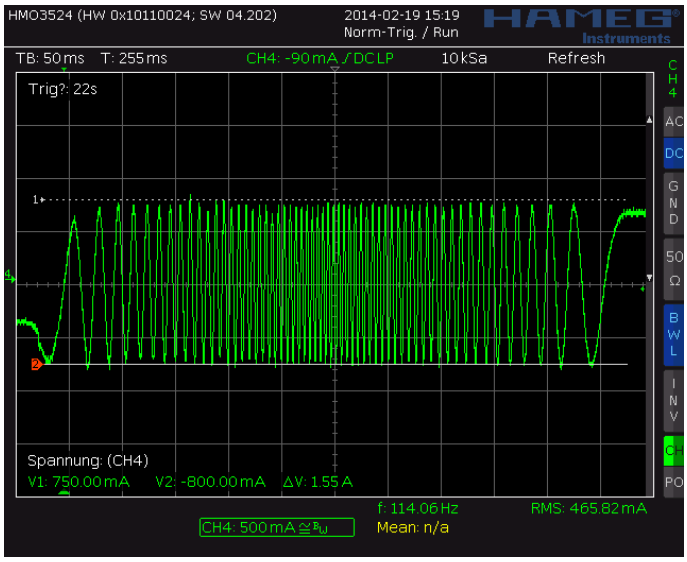


Figure 7.3 Scope shot: good setting for PWM_REG

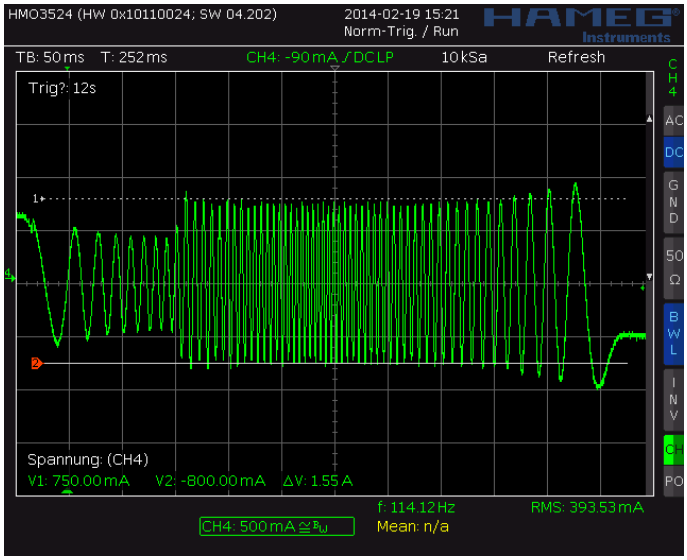


Figure 7.4 Scope shot: too small setting for PWM_REG during AT#2

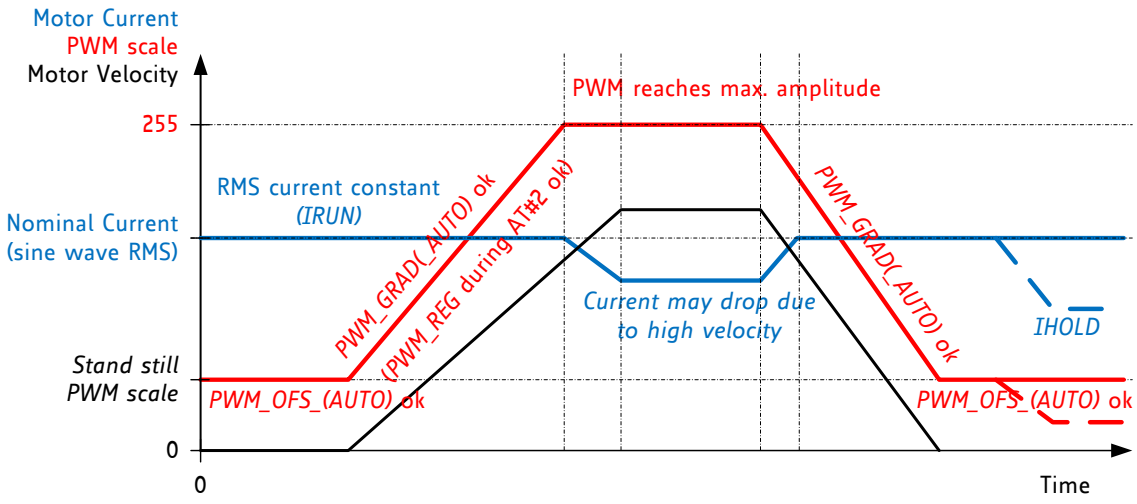


Figure 7.5 Successfully determined PWM_GRAD(_AUTO) and PWM_OFS(_AUTO)

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 22.

7.3.1 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. The motor current in automatic tuning phase AT#1 must exceed this lower limit. $I_{LOWER\ LIMIT}$ can be calculated or measured 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*.

7.4 Velocity Based Scaling

Velocity based scaling scales the stealthChop amplitude based on the time between each two steps, i.e. based on *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 TMC5161 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, e.g. 51200 = 256µsteps multiplied by 200 fullsteps for a 1.8° motor.

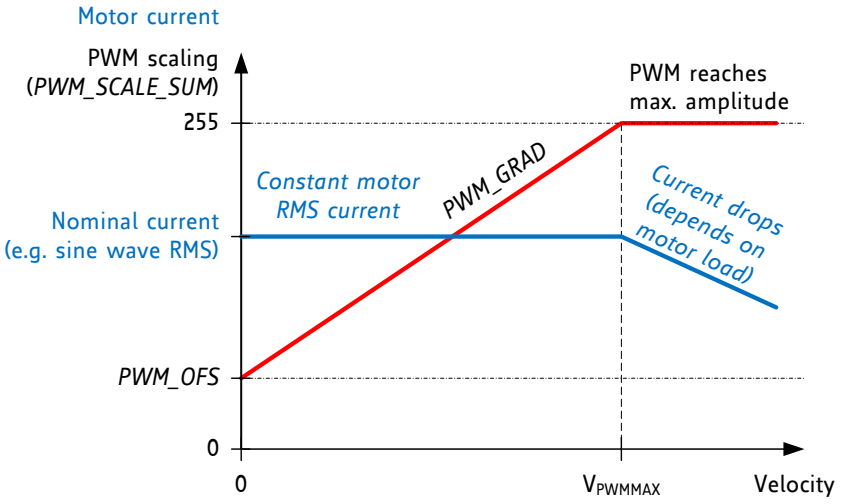


Figure 7.6 Velocity based PWM scaling (pwm_autoscale=0)

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.

7.5 Combining 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 TMC5161 allows combining stealthChop and spreadCycle based on a velocity threshold (Figure 7.7) (*TPWMTHRS*). With this, stealthChop is only active at low velocities.

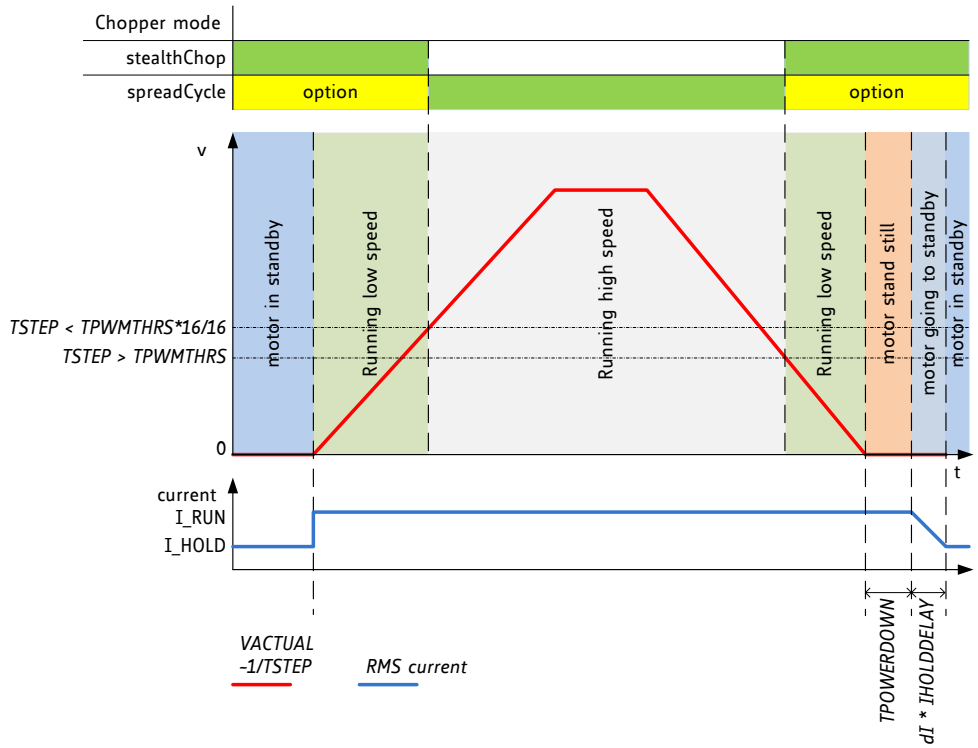


Figure 7.7 TPWMTHRS for optional switching to spreadCycle

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 in order 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.

7.6 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.

7.6.1 Open Load Flags

In stealthChop mode, 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 can be reached on both coils.

- A flickering OLA or OLB can result from asymmetries in the sense resistors or in the motor coils.
- An interrupted motor coil leads to a continuously active open load flag for the coil.
- One or both flags are active, if the current regulation did not succeed in scaling up to the full target current within the last few fullsteps (because no motor is attached or a high velocity exceeds the PWM limit).

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.

7.6.2 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.

7.7 Freewheeling and Passive Braking

stealthChop provides different options for motor standstill. These options can be enabled by setting the standstill current *IHOLD* to zero and choosing the desired option using the *FREEWHEEL* setting. The desired option becomes enabled after a time period specified by *TPOWERDOWN* and *IHOLD_DELAY*. Current regulation becomes frozen once the motor target current is at zero current in order to ensure a quick startup. With the freewheeling options, both freewheeling and passive braking can be realized. Passive braking is an effective eddy current motor braking, which consumes a minimum of energy, because no active current is driven into the coils. However, passive braking will allow slow turning of the motor when a continuous torque is applied.

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.

PARAMETERS RELATED TO STEALTHCHOP			
Parameter	**Description**	**Setting**	**Comment**
en_pwm_mode	General enable for use of stealthChop (register *GCONF*). Actual use depends on velocity thresholds	1	stealthChop enabled
		0	stealthChop off
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}

8 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 8.1 the different chopper phases are shown.

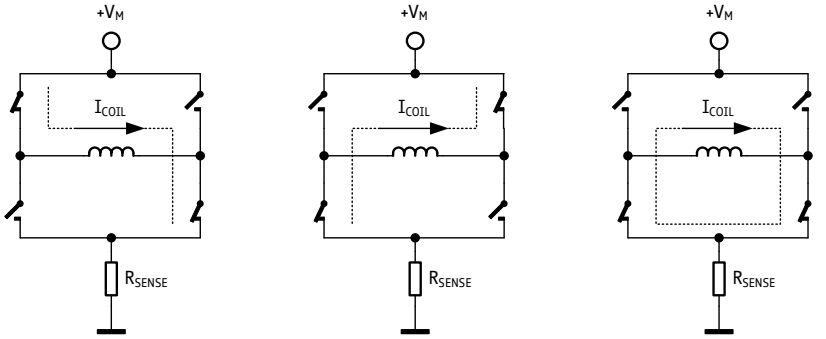


Figure 8.1 Chopper phases

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}= 12 + 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)

8.1 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 8.3). 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:

$$TOFF = (t_{OFF} * f_{CLK} - 12)/32$$

With 12 MHz clock this gives a setting of TOFF=3.4, i.e. 3 or 4.
With 16 MHz clock this gives a setting of TOFF=4.6, 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 8.2). 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.

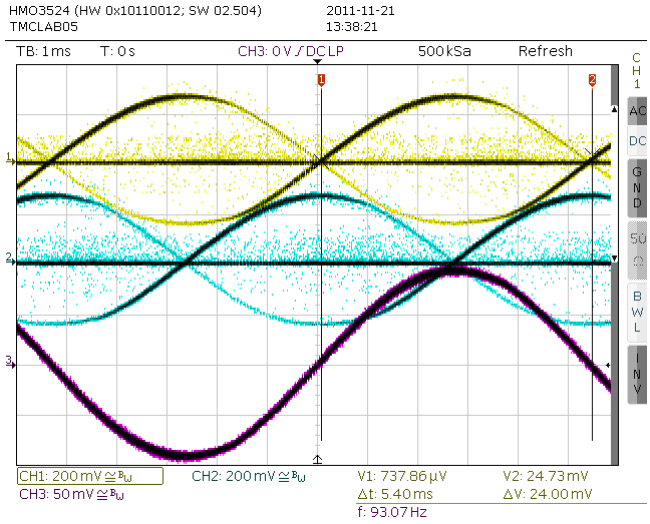


Figure 8.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)

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 22.
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.

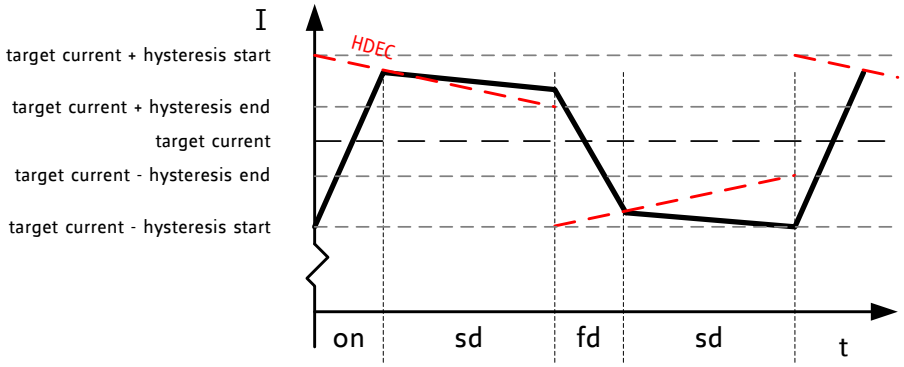


Figure 8.3 spreadCycle chopper scheme showing coil current during a chopper cycle

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.

8.2 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.

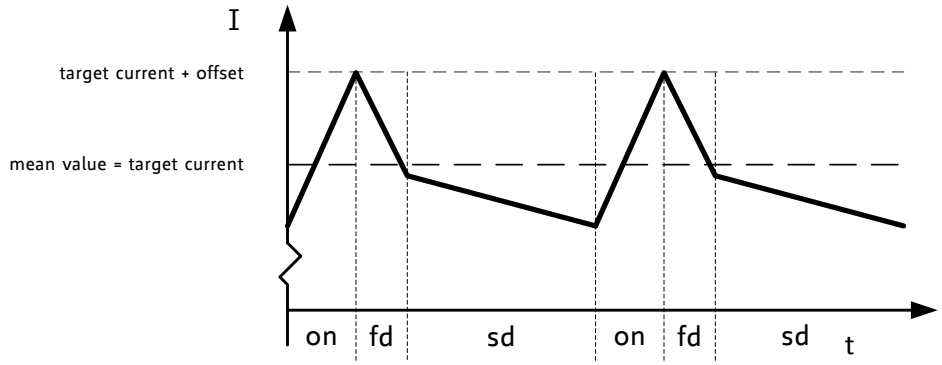


Figure 8.4 Classic const. off time chopper with offset showing coil current

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 8.5). 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.

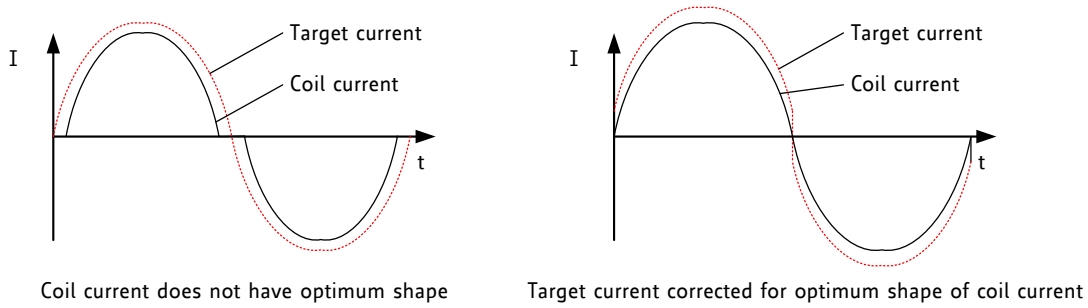


Figure 8.5 Zero crossing with classic chopper and correction using sine wave offset

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

9 Selecting Sense Resistors

The TMC5161 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*=255		
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
0.060	3.8	5.4

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 29.

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* = 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. Use for initial tuning, only.	0 … 255	scales in 1/256 steps 0=full scale

10 Velocity Based Mode Control

The TMC5161 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.

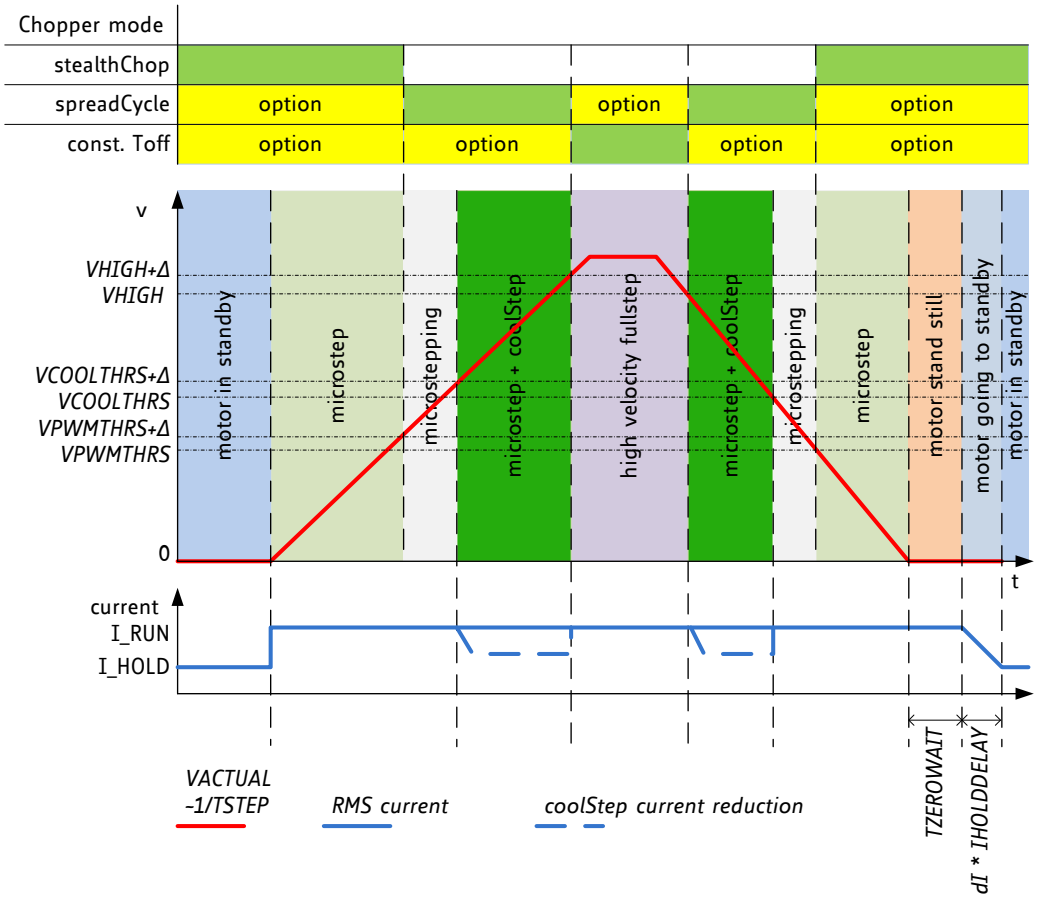


Figure 10.1 Choice of velocity dependent modes

Figure 10.1 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 stealthChop threshold *TPWMTHRS* is not shown. It can be included with VPWMTHRS < VCOOLTHRS. 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.	*0…255*	Time in multiples of 2^18 t_{CLK}
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* - Stop on stall and 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). - chopSync2 is switched off (*SYNC*=0) - 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 1	Hysteresis is 1/16 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 1	No switch to fullstep 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 in order to avoid doubling of the chopper frequency.	0 1	No change of chopper mode 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 1	No stealthChop StealthChop below VPWMTHRS

11 Diagnostics and Protection

The TMC5161 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.

11.1 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.

11.2 Short Protection

The TMC5161 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 11.1). 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, only.	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 (higher value at higher voltage)	*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).

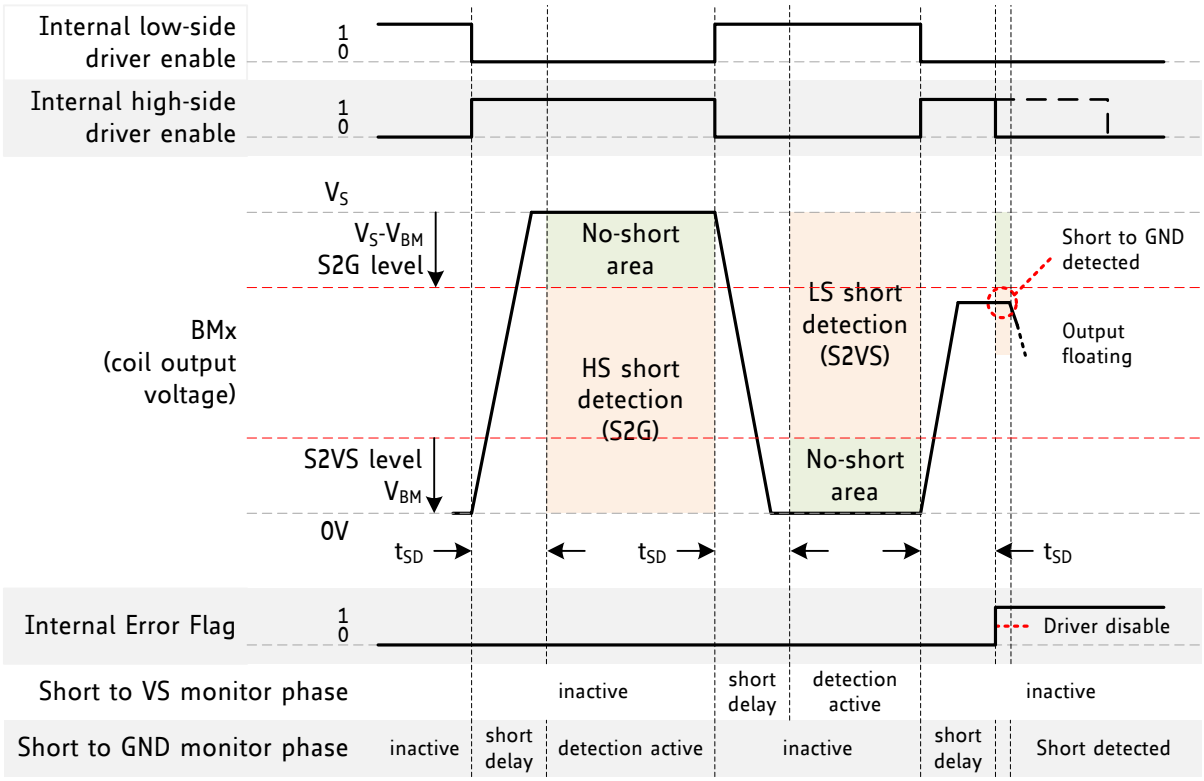


Figure 11.1 Short detection

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 (level +3 counts) is required if motor supply voltage can overshoot up to 55V
High-side short detection may falsely trigger if motor supply voltage overshoots 55V.

11.3 Open Load Diagnostics

Interrupted cables are a common cause for systems failing, e.g. when connectors are not firmly plugged. The TMC5161 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.
In order to safely detect an interrupted coil connection, read out the open load flags at low or nominal motor velocity operation, only. If possible, use spreadCycle for testing, as it provides the most accurate test. However, the *ola* and *olb* flags have just informative character and do not cause any action of the driver.

12 Ramp Generator

The ramp generator allows motion based on target position or target velocity. It automatically calculates the optimum motion profile taking into account acceleration and velocity settings. The TMC5161 integrates a new type of ramp generator, which offers faster machine operation compared to the classical linear acceleration ramps. The sixPoint ramp generator allows adapting the acceleration ramps to the torque curves of a stepper motor and uses two different acceleration settings each for the acceleration phase and for the deceleration phase. See Figure 12.2.

12.1 Real World Unit Conversion

The TMC5161 uses its internal or external clock signal as a time reference for all internal operations. Thus, all time, velocity and acceleration settings are referenced to f_{CLK}. For best stability and reproducibility, it is recommended to use an external quartz oscillator as a time base, or to provide a clock signal from a microcontroller.

The units of a TMC5161 register content are written as register[5161].

PARAMETER VS. UNITS		
Parameter / Symbol	**Unit**	**calculation / description / comment**
f_{CLK}[Hz]	[Hz]	clock frequency of the TMC5161 in [Hz]
s	[s]	second
US	µstep	
FS	fullstep	
µstep velocity v[Hz]	µsteps / s	v[Hz] = v[5161] * (f_{CLK}[Hz]/2 / 2^23)
µstep acceleration a[Hz/s]	µsteps / s^2	a[Hz/s] = a[5161] * f_{CLK}[Hz]^2 / (512*256) / 2^24
USC microstep count	counts	microstep resolution in number of microsteps (i.e. the number of microsteps between two fullsteps – normally 256)
rotations per second v[rps]	rotations / s	v[rps] = v[µsteps/s] / USC / FSC FSC: motor fullsteps per rotation, e.g. 200
rps acceleration a[rps/s^2]	rotations / s^2	a[rps/s^2] = a[µsteps/s^2] / USC / FSC
ramp steps[µsteps] = rs	µsteps	rs = (v[5161])^2 / a[5161] / 2^8 microsteps during linear acceleration ramp (assuming acceleration from 0 to v)
TSTEP, T…THRS	-	TSTEP = f_{CLK} / f_{STEP} The time reference for velocity thresholds is referred to the actual microstep frequency of the clock input respectively velocity v[Hz].

In rare cases, the upper acceleration limit might impose a limitation to the application, e.g. when working with a reduced clock frequency or high gearing and low load on the motor. In order to increase the effective acceleration possible, the microstep resolution of the sequencer input may be decreased. Setting the *CHOPCONF* options *intpol*=1 and *MRES*=%0001 will double the motor velocity for the same speed setting and thus also double effective acceleration and deceleration. The motor will have the same smoothness, but half position resolution with this setting.

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 22.

12.2 Motion Profiles

For the ramp generator register set, please refer to the chapter 6.3.

12.2.1 Ramp Mode

The ramp generator delivers two phase acceleration and two phase deceleration ramps with additional programmable start and stop velocities (see Figure 12.1).

Note
The start velocity can be set to zero, if not used.
The stop velocity can be set to ten (or down to one), if not used.
Take care to always set *VSTOP* identical to or above *VSTART*. This ensures that even a short motion can be terminated successfully at the target position.

The two different sets of acceleration and deceleration can be combined freely. *A common transition speed V1 allows for velocity dependent switching between both acceleration and deceleration settings*. A typical use case will use lower acceleration and deceleration values at higher velocities, as the motors torque declines at higher velocity. When considering friction in the system, it becomes clear, that typically deceleration of the system is quicker than acceleration. Thus, deceleration values can be higher in many applications. This way, operation speed of the motor in time critical applications can be maximized.
As target positions and ramp parameters may be changed any time during the motion, the motion controller will always use the optimum (fastest) way to reach the target, while sticking to the constraints set by the user. This way it might happen, that the motion becomes automatically stopped, crosses zero and drives back again. This case is flagged by the special flag *second_move*.

12.2.2 Start and Stop Velocity

When using increased levels of start- and stop velocity, it becomes clear, that a subsequent move into the opposite direction would provide a jerk identical to *VSTART+VSTOP*, rather than only *VSTART*. As the motor probably is not able to follow this, you can set a time delay for a subsequent move by setting *TZEROWAIT*. An active delay time is flagged by the flag *t_zerowait_active*. Once the target position is reached, the flag *position_reached* becomes active.

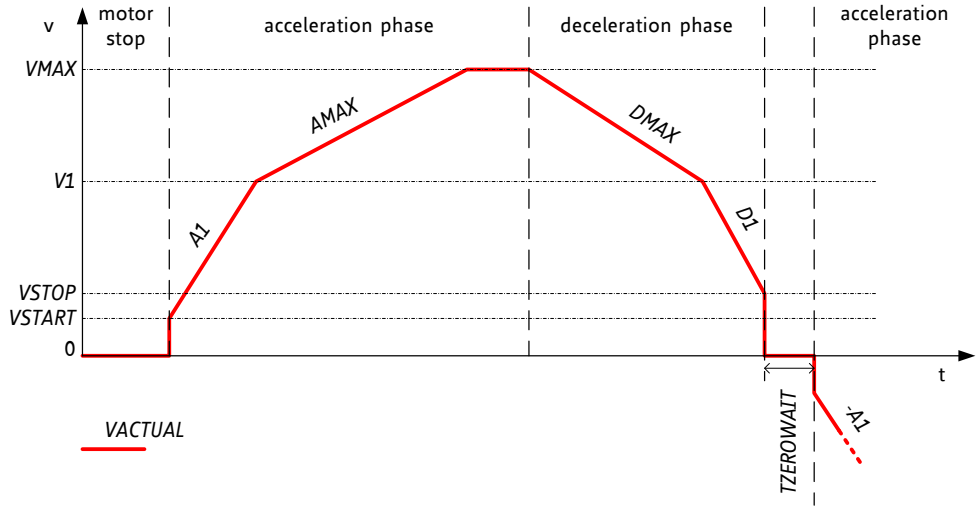


Figure 12.1 Ramp generator velocity trace showing consequent move in negative direction

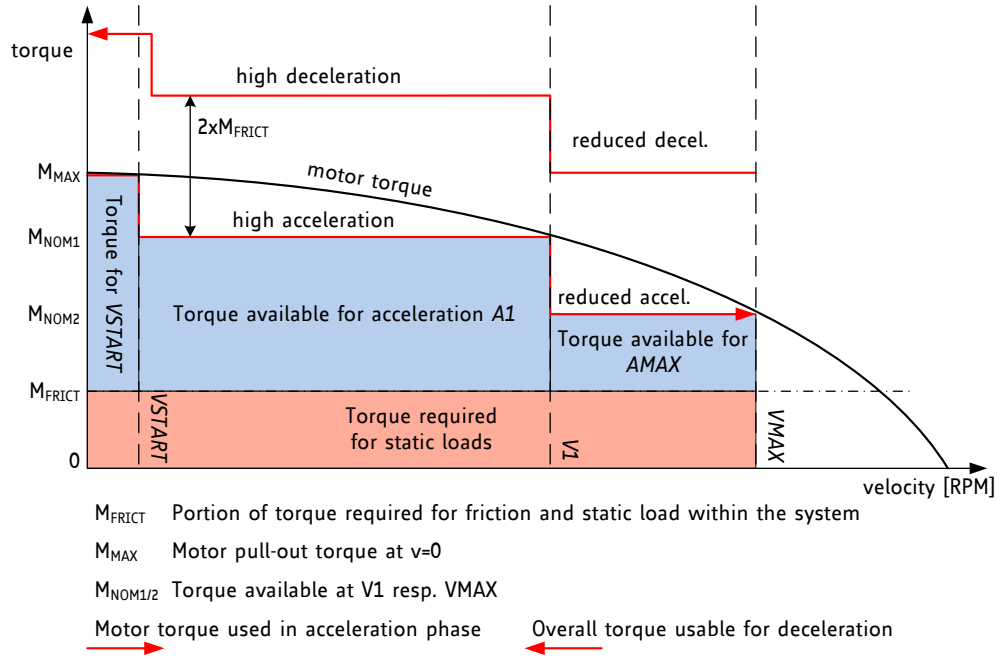


Figure 12.2 Illustration of optimized motor torque usage with TMC5161 ramp generator

12.2.3 Velocity Mode

For the ease of use, velocity mode movements do not use the different acceleration and deceleration settings. You need to set *VMAX* and *AMAX* only for velocity mode. The ramp generator always uses *AMAX* to accelerate or decelerate to *VMAX* in this mode.

In order to decelerate the motor to stand still, it is sufficient to set *VMAX* to zero. The flag *vzero* signals standstill of the motor. The flag *velocity_reached* always signals, that the target velocity has been reached.

12.2.4 Early Ramp Termination

In cases where users can interact with a system, some applications require terminating a motion by ramping down to zero velocity before the target position has been reached.

OPTIONS TO TERMINATE MOTION USING ACCELERATION SETTINGS:

a) Switch to velocity mode, set *VMAX*=0 and *AMAX* to the desired deceleration value. This will stop the motor using a linear ramp.

b) For a stop in positioning mode, set *VSTART*=0 and *VMAX*=0. *VSTOP* is not used in this case. The driver will use *AMAX* and *A1* (as determined by *V1*) for going to zero velocity.

c) For a stop using *D1*, *DMAX* and *VSTOP*, trigger the deceleration phase by copying *XACTUAL* to *XTARGET*. Set *TZEROWAIT* sufficiently to allow the CPU to interact during this time. The driver will decelerate and eventually come to a stop. Poll the actual velocity to terminate motion during *TZEROWAIT* time using option a) or b).

d) Activate a stop switch. This can be done by means of the hardware input, e.g. using a wired 'OR' to the stop switch input. If you do not use the hardware input and have tied the REFL and REFR to a fixed level, enable the stop function (*stop_l_enable*, *stop_r_enable*) and use the inverting function (*pol_stop_l*, *pol_stop_r*) to simulate the switch activation.

12.2.5 Application Example: Joystick Control

Applications like surveillance cameras can be optimally enhanced using the motion controller: while joystick commands operate the motor at a user defined velocity, the target ramp generator ensures that the valid motion range never is left.

REALIZE JOYSTICK CONTROL

1. Use positioning mode in order to control the motion direction and to set the motion limit(s).
2. Modify *VMAX* at any time in the range *VSTART* to your maximum value. With *VSTART*=0, you can also stop motion by setting *VMAX*=0. The motion controller will use *A1* and *AMAX* as determined by *V1* to adapt velocity for ramping up and ramping down.
3. In case you do not modify the acceleration settings, you do not need to rewrite *XTARGET*, just modify *VMAX*.
4. *DMAX*, *D1* and *VSTOP* only become used when the ramp controller slows down due to reaching the target position, or when the target position has been modified to point to the other direction.

12.3 Velocity Thresholds

The ramp generator provides a number of velocity thresholds coupled with the actual velocity *VACTUAL*. The different ranges allow programming the motor to the optimum step mode, coil current and acceleration settings. Most applications will not require all of the thresholds, but in principle all modes can be combined as shown in Figure 12.1. VHIGH and VCOOLTHRS are determined by the settings *THIGH* and *TCOOLTHRS* in order to allow determination of the velocity when an external step source is used. *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 stealthChop threshold *TPWMTHRS* is not shown. It can be included with VPWMTHRS < VCOOLTHRS.

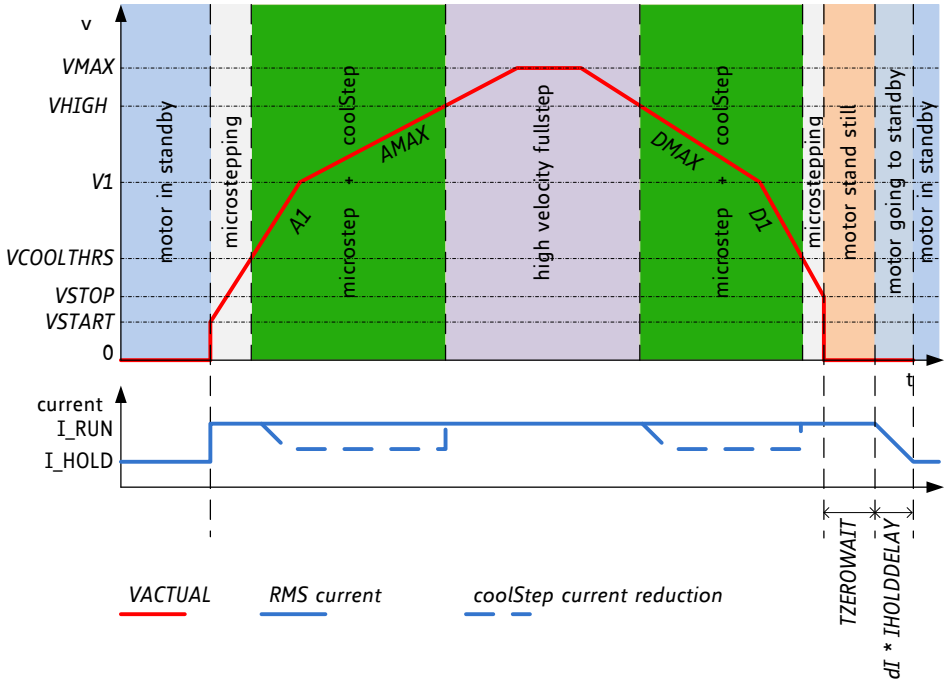


Figure 12.3 Ramp generator velocity dependent motor control

The velocity thresholds for the different chopper modes and sensorless operation features are coupled to the time between each two microsteps *TSTEP*.

12.4 Reference Switches

Prior to normal operation of the drive an absolute reference position must be set. The reference position can be found using a mechanical stop which can be detected by stall detection, or by a reference switch.

In case of a linear drive, the mechanical motion range must not be left. This can be ensured also for abnormal situations by enabling the stop switch functions for the left and the right reference switch. Therefore, the ramp generator responds to a number of stop events as configured in the *SW_MODE* register. There are two ways to stop the motor:

- It can be stopped abruptly, when a switch is hit. This is useful in an emergency case and for stallGuard based homing.
- Or the motor can be softly decelerated to zero using deceleration settings (DMAX, V1, D1).

Hint
Latching of the ramp position *XACTUAL* to the holding register *XLATCH* upon a switch event gives a precise snapshot of the position of the reference switch.

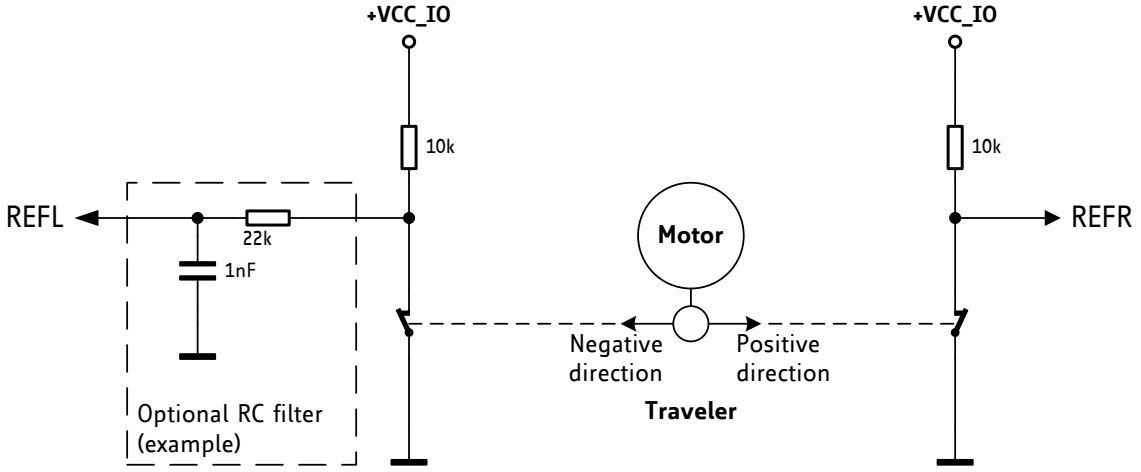


Figure 12.4 Using reference switches (example)

Normally open or normally closed switches can be used by programming the switch polarity or selecting the pullup or pull-down resistor configuration. A normally closed switch is failsafe with respect to an interrupt of the switch connection. Switches which can be used are:

- mechanical switches,
- photo interrupters, or
- hall sensors.

Be careful to select reference switch resistors matching your switch requirements!
In case of long cables additional RC filtering might be required near the TMC5161 reference inputs. Adding an RC filter will also reduce the danger of destroying the logic level inputs by wiring faults, but it will add a certain delay which should be considered with respect to the application.

IMPLEMENTING A HOMING PROCEDURE

1. Make sure, that the home switch is not pressed, e.g. by moving away from the switch.
2. Activate position latching upon the desired switch event and activate motor (soft) stop upon active switch. stallGuard based homing requires using a hard stop (*en_softstop*=0).
3. Start a motion ramp into the direction of the switch. (Move to a more negative position for a left switch, to a more positive position for a right switch). You may timeout this motion by using a position ramping command.

4. As soon as the switch is hit, the position becomes latched and the motor is stopped. Wait until the motor is in standstill again by polling the actual velocity *VACTUAL* or checking *vzero* or the *standstill* flag.
5. Switch the ramp generator to hold mode and calculate the difference between the latched position and the actual position. For stallGuard based homing or when using hard stop, *XACTUAL* stops exactly at the home position, so there is no difference (0).
6. Write the calculated difference into the actual position register. Now, homing is finished. A move to position 0 will bring back the motor exactly to the switching point. In case stallGuard was used for homing, read and write back *RAMP_STAT* to clear the stallGuard stop event *event_stop_sg* and release the motor from the stop condition.

HOMING WITH A THIRD SWITCH

Some applications use an additional home switch, which operates independently of the mechanical limit switches. The encoder functionality of the TMC5161 provides an additional source for position latching. It allows using the N channel input to snapshot *XACTUAL* with a rising or falling edge event, or both. This function also provides an interrupt output.

1. Activate the latching function (*ENCMODE*: Set *ignoreAB*, *clr_cont*, *neg_edge* or *pos_edge* and *latch_x_act*). The latching function can then trigger the interrupt output (check by reading *n_event* in *ENC_STATUS* when interrupt is signaled at DIAG0).
2. Move to the direction, where the N channel switch should be. In case the motor hits a stop switch (REFL or REFR) before the home switch is detected, reverse the motion direction.
3. Read out *XLATCH* once the switch has been triggered. It gives the position of the switch event.
4. After detection of the switch event, stop the motor, and subtract *XLATCH* from the actual position. Read and write back *ENC_STAT* to clear the status flags. (A detailed description of the required steps is in the homing procedure above.)

13 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 13.1. 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.

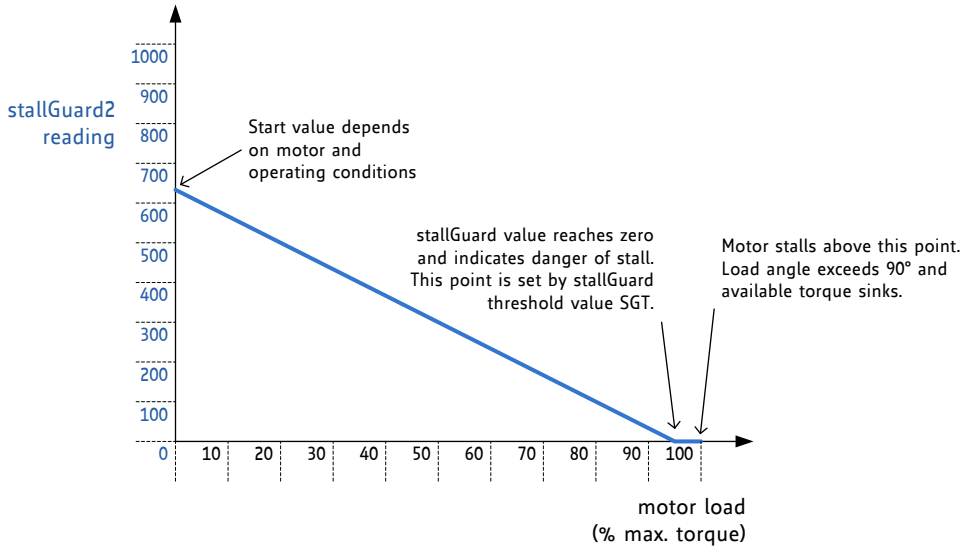


Figure 13.1 Function principle of stallGuard2

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!

13.1 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 enable *sg_stop* to enable the stop on stall feature. Make sure, that the motor is safely stopped whenever it is stalled. Increase *SGT* if the motor becomes stopped before a stall occurs. Restart the motor by disabling *sg_stop* or by reading and writing back the *RAMP_STAT* register (write+clear function).
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 ramp generator can be programmed to stop the motor upon a stall event by enabling *sg_stop* in *SW_MODE*. Set *TCOOLTHRS* to match the lower velocity threshold where stallGuard delivers a good result in order to use *sg_stop*.

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 22.
For detail procedure see Application Note AN002 - *Parameterization of stallGuard2 & coolStep*

13.1.1 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.

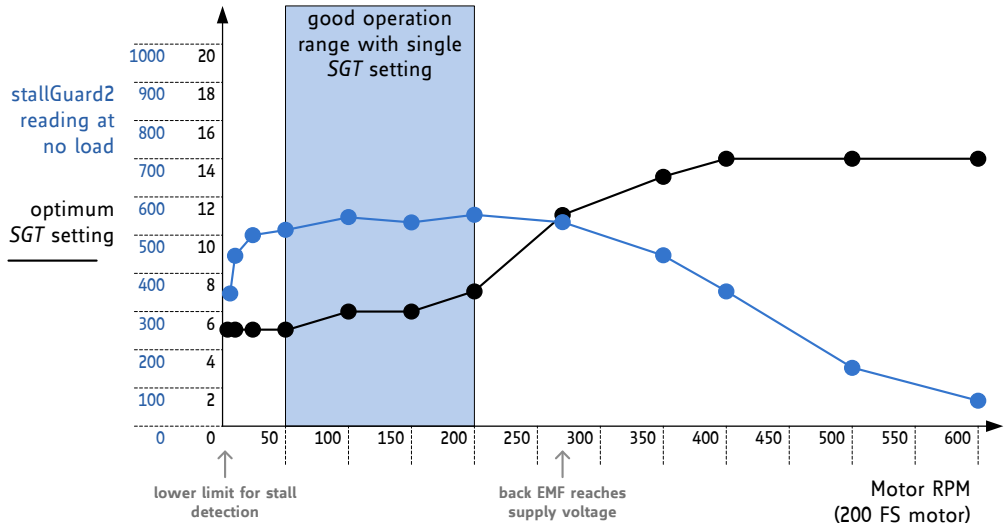


Figure 13.2 Example: optimum SGT setting and stallGuard2 reading with an example motor

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.

13.1.2 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.

13.1.3 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.

13.1.4 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:

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

13.2 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.

13.3 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.

13.4 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 (set *sg_stop* in *SW_MODE*). Once a stall is detected, the ramp generator stops motion and sets *VACTUAL* zero, stopping the motor. The stop condition also is indicated by the flag *stallGuard* in *DRV_STATUS*. After setting up new motion parameters in order to prevent the motor from restarting right away, stallGuard can be disabled, or the motor can be re-enabled by reading and writing back *RAMP_STAT*. The write and clear function of the *event_stop_sg* flag in *RAMP_STAT* restarts the motor after expiration of *TZEROWAIT* in case the motion parameters have not been modified. 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).

13.5 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.

14 coolStep Operation

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

14.1 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.

14.2 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 14.1 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.

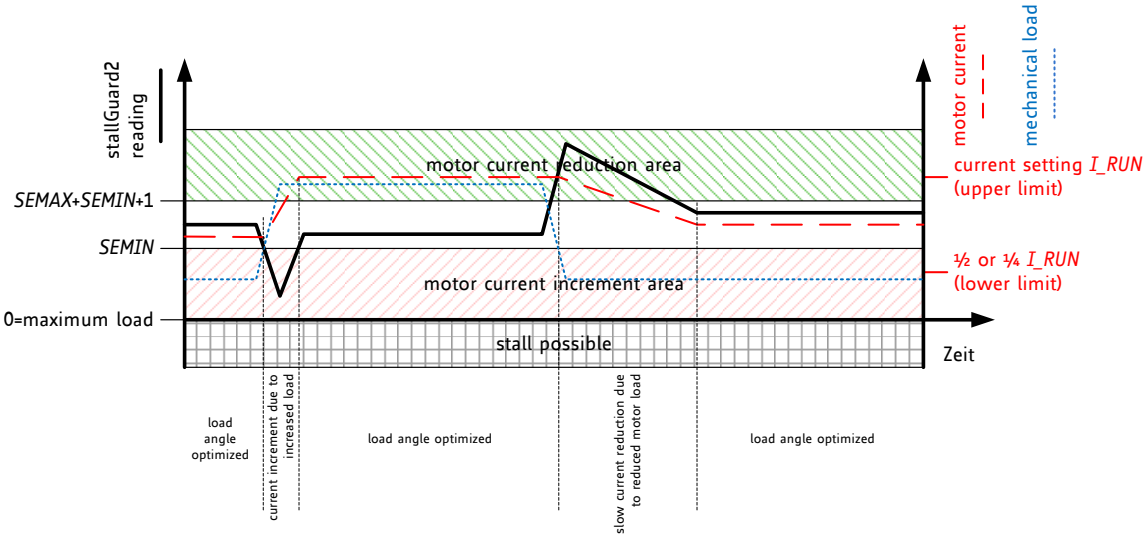


Figure 14.1 coolStep adapts motor current to the load

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	0: 1/2 of IRUN
		1	1: 1/4 of IRUN
TCOOLTHRS	Lower velocity threshold for switching on coolStep and stop on stall. 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 stop on stall. 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

14.3 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.

14.3.1 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.

14.3.2 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 in the ramp generator. 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.

15 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. The only motion controller registers remaining active in this case are the current settings in register *IHOLD_IRUN*.

15.1 Timing

Figure 15.1 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.

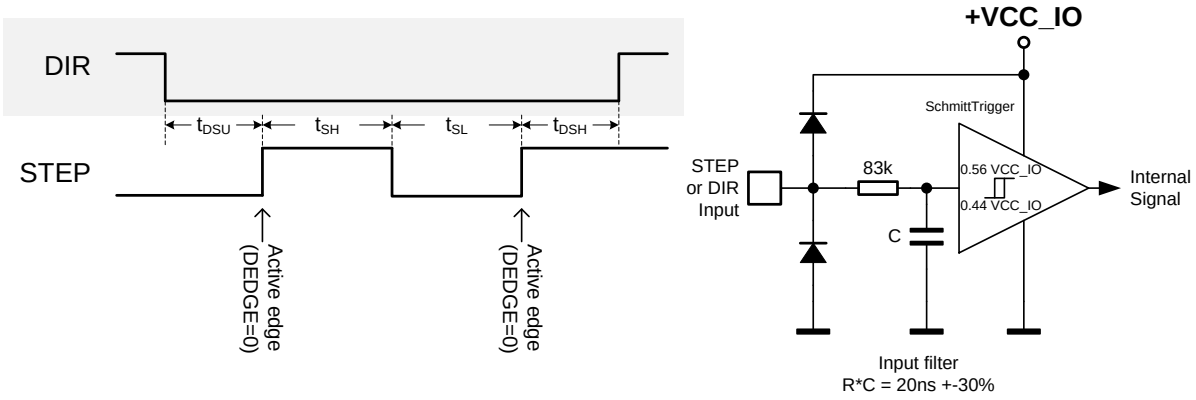


Figure 15.1 STEP and DIR timing, Input pin filter

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.

15.2 Changing Resolution

The TMC5161 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%

15.3 microPlyer and Stand Still Detection

For each active edge on STEP, microPlyer produces microsteps at 256x resolution, as shown in Figure 15.2. 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.

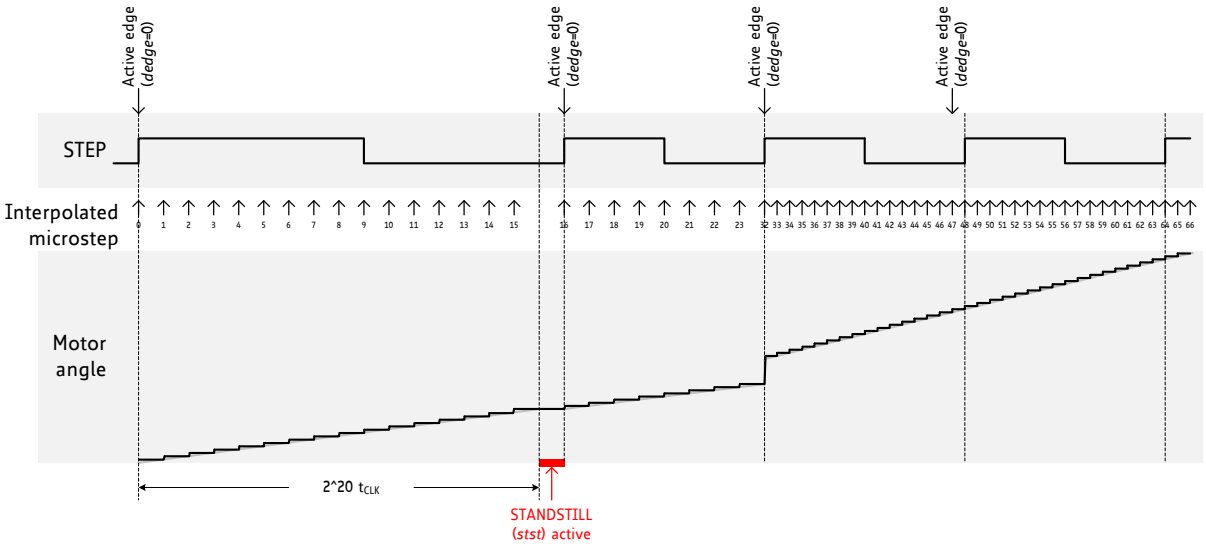


Figure 15.2 microPlyer microstep interpolation with rising STEP frequency (Example: 16 to 256)

In Figure 15.2, 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.

16 DIAG Outputs

16.1 STEP/DIR Mode

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 16.1 shows the available signals and control bits.

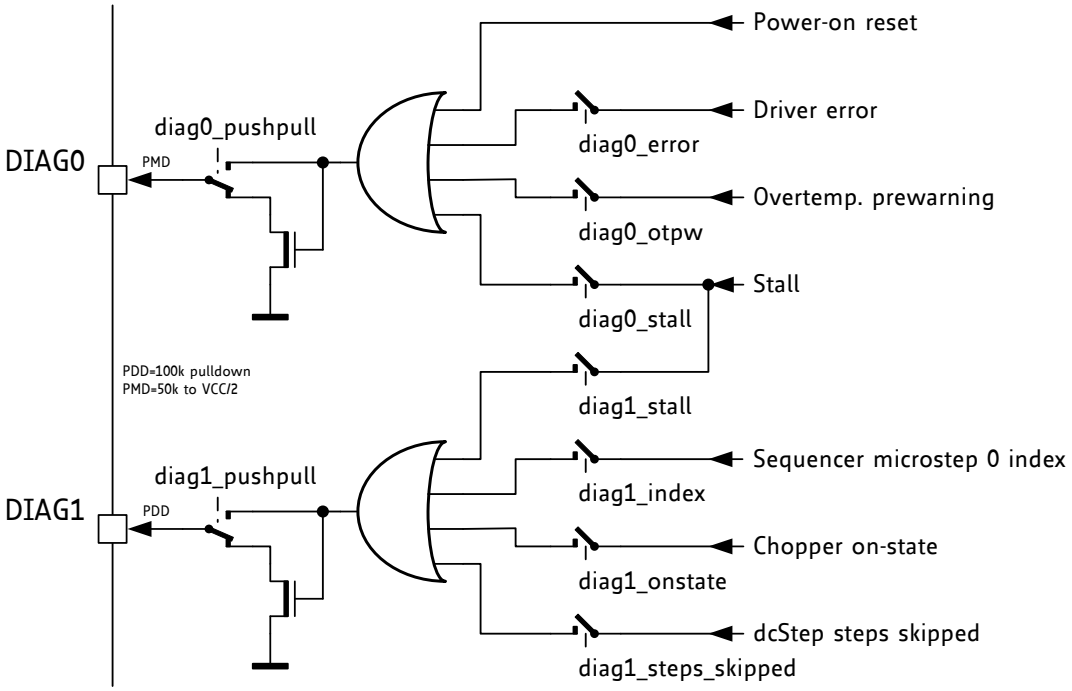


Figure 16.1 DIAG outputs in STEP/DIR mode

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 in order 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.

16.2 Motion Controller Mode

In motion controller mode, the DIAG outputs deliver a position compare signal to allow exact triggering of external logic, and an interrupt signal in order to trigger software to certain conditions within the motion ramp. Either an open drain (active low) output signal can be chosen (default), or an active high push-pull output signal. When using the open drain output, an external pull up resistor in the range 4.7kΩ to 33kΩ is required. DIAG0 also becomes driven low upon a reset condition. However,

the end of the reset condition cannot be determined by monitoring DIAG0 in this configuration, because *event_pos_reached* flag also becomes active upon reset and thus the pin stays actively low after the reset condition. In order to safely determine a reset condition, monitor the *reset* flag by SPI or read out any register to confirm that the chip is powered up.

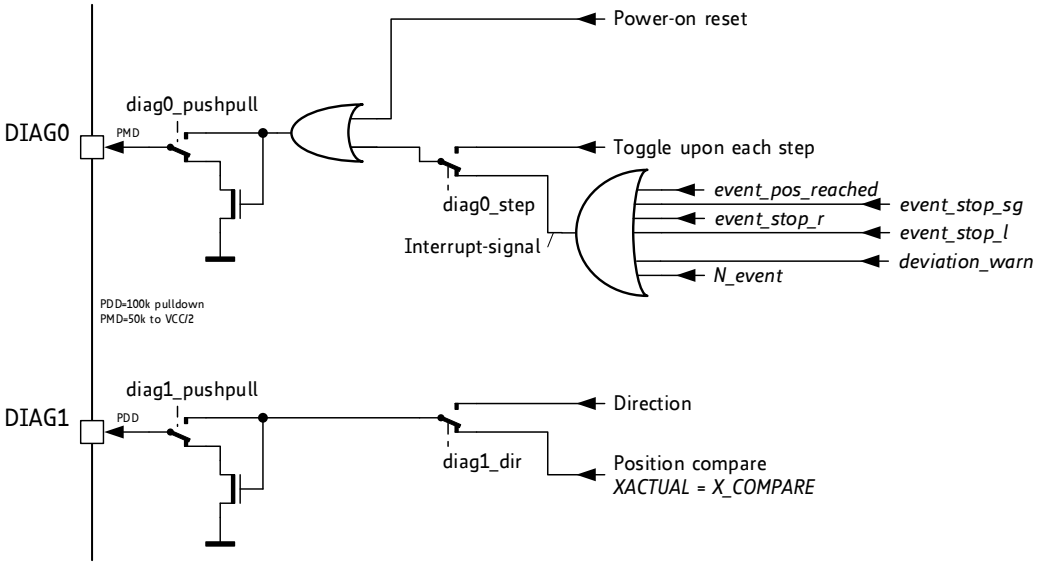


Figure 16.2 DIAG outputs with SD_MODE=0

17 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 ramp generator 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 allow compensating for mechanical torque peaks. In case the motor becomes completely blocked, the stall flag becomes set.

17.1 User Benefits

dcStep™

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!

17.2 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, in order 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.

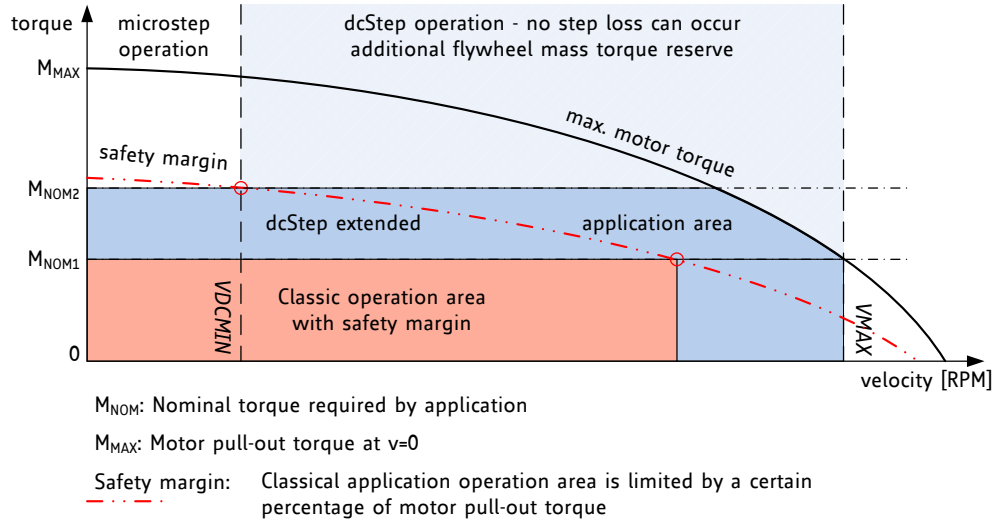


Figure 17.1 dcStep extended application operation area

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 22.
For detail configuration procedure see Application Note AN003 - *dcStep*

17.3 dcStep Integration with the Motion Controller

dcStep requires only a few settings. It directly feeds back motor motion to the ramp generator, so that it becomes seamlessly integrated into the motion ramp, even if the motor becomes overloaded with respect to the target velocity. dcStep operates the motor in fullstep mode at the ramp generator target velocity *VACTUAL* or at reduced velocity if the motor becomes overloaded. It requires setting the minimum operation velocity *VDCMIN*. *VDCMIN* shall be set to the lowest operating velocity where dcStep gives a reliable detection of motor operation. The motor never stalls unless it becomes braked to a velocity below *VDCMIN*. In case the velocity should fall below this value, the motor would restart once its load is released, unless the stall detection becomes enabled (set *sg_stop*). Stall detection is covered by stallGuard2.

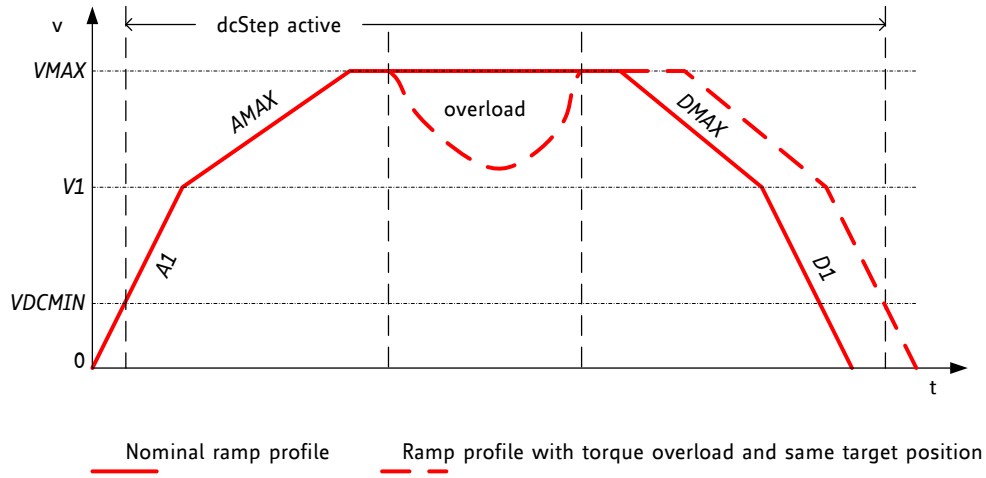


Figure 17.2 Velocity profile with impact by overload situation

Hint
dcStep requires that the phase polarity of the sine wave is positive within the *MSCNT* range 768 to 255 and negative within 256 to 767. The cosine polarity must be positive from 0 to 511 and negative from 512 to 1023. A phase shift by 1 would disturb dcStep operation. Therefore it is advised to work with the default wave. Please refer chapter 18.2 for an initialization with the default table.

17.4 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. In order to safely detect a step loss and avoid restarting of the motor, the stop on stall can be enabled (set flag *sg_stop*). In this case *VACTUAL* becomes set to zero once the motor is stalled. It remains stopped until reading the *RAMP_STAT* status flags. The flag *event_stop_sg* shows the active stop condition. 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. In order 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. Enable *sg_stop* flag for stopping the motor upon a stall event. This way the motor will be stopped once it stalls.	0… 255	Set slightly higher than *DC_TIME* / 16

17.5 Measuring Actual Motor Velocity in dcStep Operation

dcStep has the ability to reduce motor velocity in case the motor becomes slower than the target velocity due to mechanical load. *VACTUAL* shows the ramp generator target velocity. It is not influenced by dcStep. Measuring dcStep velocity is possible based on the position counter *XACTUAL*.

Therefore take two snapshots of the position counter with a known time difference:

$$VACTUAL_{DCSTEP} = \frac{XACTUAL(time2) - XACTUAL(time1)}{time2 - time1} * \frac{2^{24}}{f_{CLK}}$$

Example:

At 16.0 MHz clock frequency, a 0.954 second measurement delay would directly yield in the velocity value, a 9.54 ms delay would yield in 1/100 of the actual dcStep velocity.

To grasp the time interval as precisely as possible, snapshot a timer each time the transmission of *XACTUAL* from the IC starts or ends. The rising edge of NCS for SPI transmission provides the most exact time reference.

17.6 dcStep with STEP/DIR Interface

The TMC5161 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 TMC5161 automatically reduces the motor velocity or stops the motor upon overload. In order 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.

17.6.1 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 TMC5161 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 17.3 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.

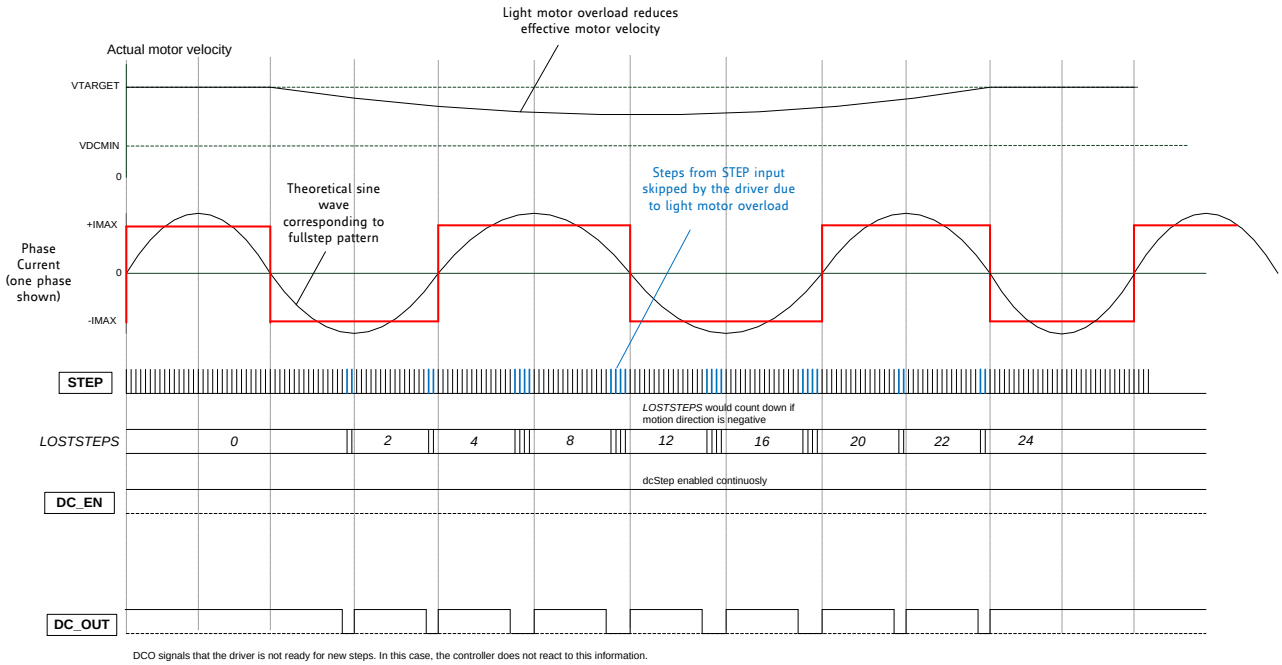


Figure 17.3 Motor moving slower than STEP input due to light overload. LOSTSTEPS incremented

17.6.2 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 17.5 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.

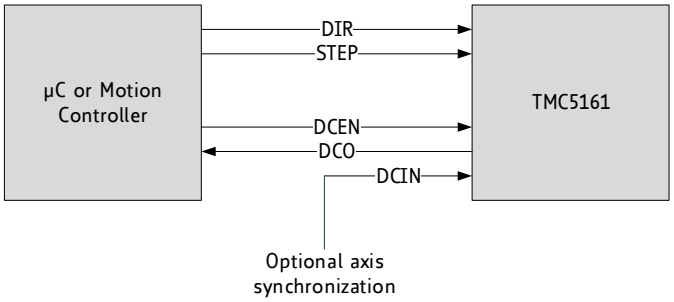


Figure 17.4 Full signal interconnection for dcStep

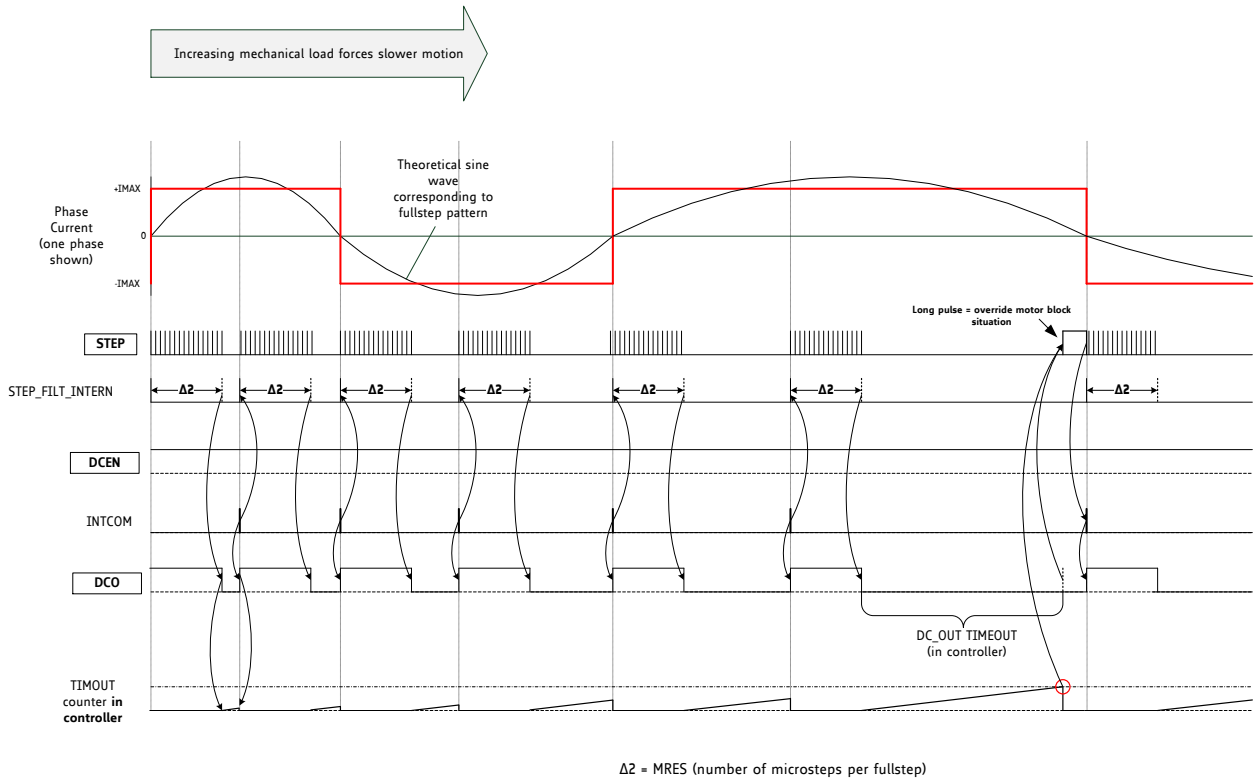


Figure 17.5 DCO Interface to motion controller – step generator stops when DCO is asserted

18 Sine-Wave Look-up Table

The TMC5161 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.

18.1 User Benefits

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

18.2 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, in order to give the best possible resolution while leaving headroom for the hysteresis based chopper to add an offset.

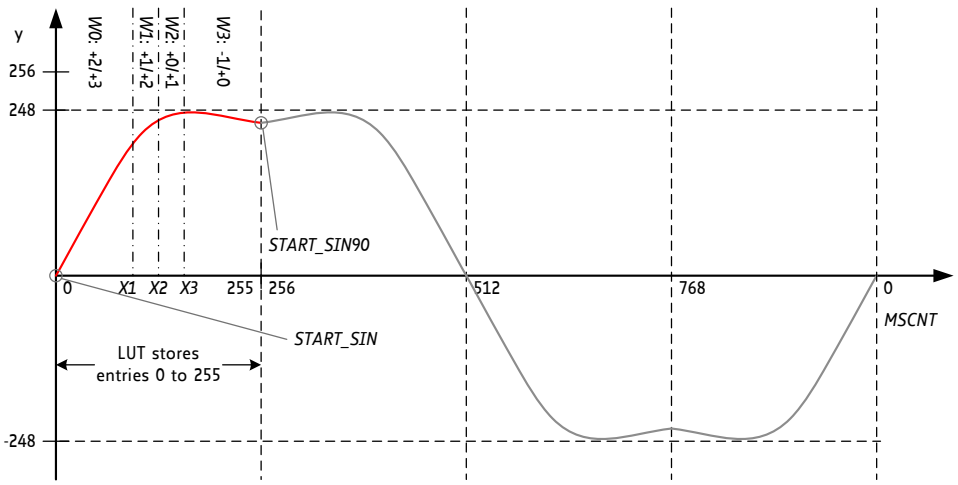


Figure 18.1 LUT programming example

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 6.5 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 TMC5161-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]= %00010000000010000010000010000100010 = 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

19 Emergency Stop

The driver provides a negative active enable pin 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 ENCA_DCIN to act as a step disable function. Set GCONF flag *stop_enable* to activate this option. Whenever ENCA_DCIN becomes pulled up, the motor will stop abruptly and go to the power down state, as configured via *IHOLD*, *IHOLD_DELAY* and stealthChop standstill options. Disabling the driver via ENN will require three clock cycles to safely switch off the driver.

20 ABN Incremental Encoder Interface

The TMC5161 is equipped with an incremental encoder interface for ABN encoders. The encoder inputs are multiplexed with other signals in order to keep the pin count of the device low. The basic selection of the peripheral configuration is set by the register *GCONF*. The use of the N channel is optional, as some applications might use a reference switch or stall detection rather than an encoder N channel for position referencing. The encoders give positions via digital incremental quadrature signals (usually named A and B) and a clear signal (usually named N for null or Z for zero).

N SIGNAL

The N signal can be used to clear the position counter or to take a snapshot. To continuously monitor the N channel and trigger clearing of the encoder position or latching of the position, where the N channel event has been detected, set the flag *clr_cont*. Alternatively it is possible to react to the next encoder N channel event only, and automatically disable the clearing or latching of the encoder position after the first N signal event (flag *clr_once*). This might be desired because the encoder gives this signal once for each revolution.

Some encoders require a validation of the N signal by a certain configuration of A and B polarity. This can be controlled by *pol_A* and *pol_B* flags in the *ENCMODE* register. For example, when both *pol_A* and *pol_B* are set, an active N-event is only accepted during a high polarity of both, A and B channel.

For clearing the encoder position *ENC_POS* with the next active N event set *clr_enc_x* = 1 and *clr_once* = 1 or *clr_cont* = 1.

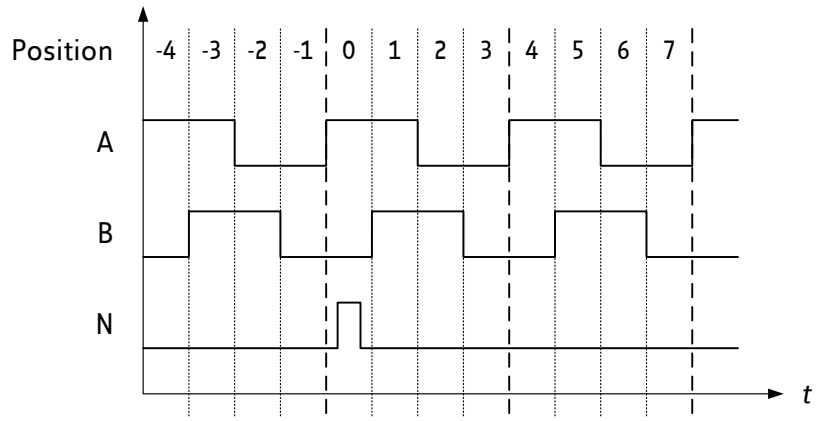


Figure 20.1 Outline of ABN signals of an incremental encoder

THE ENCODER CONSTANT *ENC_CONST*

The encoder constant *ENC_CONST* is added to or subtracted from the encoder counter on each polarity change of the quadrature signals AB of the incremental encoder. The encoder constant *ENC_CONST* represents a signed fixed point number (16.16) to facilitate the generic adaption between motors and encoders. In decimal mode, the lower 16 bits represent a number between 0 and 9999. For stepper motors equipped with incremental encoders the fixed number representation allows very comfortable parameterization. Additionally, mechanical gearing can easily be taken into account. Negating the sign of *ENC_CONST* allows inversion of the counting direction to match motor and encoder direction.

Examples:

- Encoder factor of 1.0: *ENC_CONST* = 0x0001.0x0000 = FACTOR.FRACTION
- Encoder factor of -1.0: *ENC_CONST* = 0xFFFF.0x0000. This is the two's complement of 0x00010000. It equals (2^16-(FACTOR+1)).(2^16-FRACTION)
- Decimal mode encoder factor 25.6: 00025.6000 = 0x0019.0x1770 = FACTOR.DECIMALS

- Decimal mode encoder factor -25.6: 0xFFE6.4000 = 0xFFE6.0x0FA0. This equals (2^16-(FACTOR+1)).(10000-DECIMALS)

THE ENCODER COUNTER *X_ENC*

The encoder counter *X_ENC* holds the current encoder position ready for read out. Different modes concerning handling of the signals A, B, and N take into account active low and active high signals found with different types of encoders. For more details please refer to the register mapping in section 6.4.

THE REGISTER *ENC_STATUS*

The register *ENC_STATUS* holds the status concerning the event of an encoder clear upon an N channel signals. The register *ENC_LATCH* stores the actual encoder position on an N signal event.

20.1 Encoder Timing

The encoder inputs use analog and digital filtering to ensure reliable operation even with increased cable length. The maximum continuous counting rate is limited by input filtering to 2/3 of f_{CLK}.

Encoder interface timing	**AC-Characteristics** clock period is t_{CLK}					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Encoder counting frequency	f_{CNT}			<2/3 f_{CLK}	f_{CLK}	
A/B/N input low time	t_{ABNL}		3 t_{CLK}+20			ns
A/B/N input high time	t_{ABNH}		3 t_{CLK}+20			ns
A/B/N spike filtering time	$t_{FILTABN}$	Rising and falling edge		3 t_{CLK}		

20.2 Setting the Encoder to Match Motor Resolution

Encoder example settings for motor parameters: USC=256 µsteps, 200 fullstep motor
Factor = FSC*USC / encoder resolution

ENCODER EXAMPLE SETTINGS FOR A 200 FULLSTEP MOTOR WITH 256 MICROSTEPS		
Encoder resolution	**Required encoder factor**	**Comment**
200	256	
360	142.2222 = 9320675.5555 / 2^16 = 1422222.2222 / 10000	No exact match possible!
500	102.4 = 6710886.4 / 2^16 = 1024000 / 10000	Exact match with decimal setting
1000	51.2	Exact match with decimal setting
1024	50	
4000	12.8	Exact match with decimal setting
4096	12.5	
16384	3.125	

Example:

The encoder constant register shall be programmed to 51.2 in decimal mode. Therefore, set $ENC\_CONST = 51 * 2^{16} + 0.2 * 10000$

20.3 Closing the Loop

Depending on the application, an encoder can be used for different purposes. Medical applications often require an additional and independent monitoring to detect hard or soft failure. Upon failure, the machine can be stopped and restarted manually. Use *ENC_DEVIATION* setting and interrupt to safely detect a step loss failure / mismatch between motor and encoder.

Less critical applications may use the encoder to detect failure, stop the motors upon step loss and restart automatically. A different use of the encoder allows increased positioning precision by positioning directly to encoder positions. The application can modify target positions based on the deviation, or even regularly update the actual position with the encoder position.

To realize a directly encoder based commutation, TRINAMIC offers the new S-ramp closed loop motion controller TMC4361.

21 DC Motor or Solenoid

The TMC5161 can drive one or two DC motors using one coil output per DC motor. Either a torque limited operation, or a voltage based velocity control with optional torque limit is possible.

CONFIGURATION AND CONTROL

Set the flag *direct_mode* in the *GCONF* register. In direct mode, the coil current polarity and coil current, respectively the PWM duty cycle become controlled by register *XTARGET* (0x2D). Bits 8..0 control motor A and Bits 24..16 control motor B PWM. Additionally to this setting, the current limit is scaled by *IHOLD*. The STEP/DIR inputs and the motion controller are not used in this mode.

PWM DUTY CYCLE VELOCITY CONTROL

In order to operate the motor at different velocities, use the stealthChop voltage PWM mode in the following configuration:
en_pwm_mode = 1, *pwm_autoscale* = 0, *PWM_OFS* = 255, *PWM_GRAD* = 4, *IHOLD* = 31
Set *TOFF* > 0 to enable the driver.
In this mode the driver behaves like a 4-quadrant power supply. The direct mode setting of PWM A and PWM B using *XTARGET* controls motor voltage, and thus the motor velocity. Setting the corresponding PWM bits between -255 and +255 (signed, two's complement numbers) will vary motor voltage from -100% to 100%. With *pwm_autoscale* = 0, current sensing is not used and the sense resistors should be eliminated or 150mΩ or less to avoid excessive voltage drop when the motor becomes heavily loaded up to 2.5A. Especially for higher current motors, make sure to slowly accelerate and decelerate the motor in order to avoid overcurrent or triggering driver overcurrent detection.

To activate optional motor freewheeling, set *IHOLD* = 0 and *FREEWHEEL* = %01.

ADDITIONAL TORQUE LIMIT

In order to additionally take advantage of the motor current limitation (and thus torque controlled operation) in stealthChop mode, use automatic current scaling (*pwm_autoscale* = 1). The actual current limit is given by *IHOLD* and scaled by the respective motor PWM amplitude, e.g. PWM = 128 yields in 50% motor velocity and 50% of the current limit set by *IHOLD*. In case two DC motors are driven in voltage PWM mode, note that the automatic current regulation will work only for the motor which has the higher absolute PWM setting. The PWM of the second motor also will be scaled down in case the motor with higher PWM setting reaches its current limitation.

PURELY TORQUE LIMITED OPERATION

For a purely torque limited operation of one or two motors, spread cycle chopper individually regulates motor current for both full bridge motor outputs. When using spreadCycle, the upper motor velocity is limited by the supply voltage only (or as determined by the load on the motor).

21.1 Solenoid Operation

The same way, one or two solenoids (i.e. magnetic coil actuators) can be operated using spreadCycle chopper. For solenoids, it is often desired to have an increased current for a short time after switching on, and reduce the current once the magnetic element has switched. This is automatically possible by taking advantage of the automatic current scaling (*IRUN*, *IHOLD*, *IHOLDDELAY* and *TPOWERDOWN*). The current scaling in *direct_mode* is still active, but will not be triggered if no step impulse is supplied. Therefore, a step impulse must be given to the STEP input whenever one of the coils shall be switched on. This will increase the current for both coils at the same time.

22 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

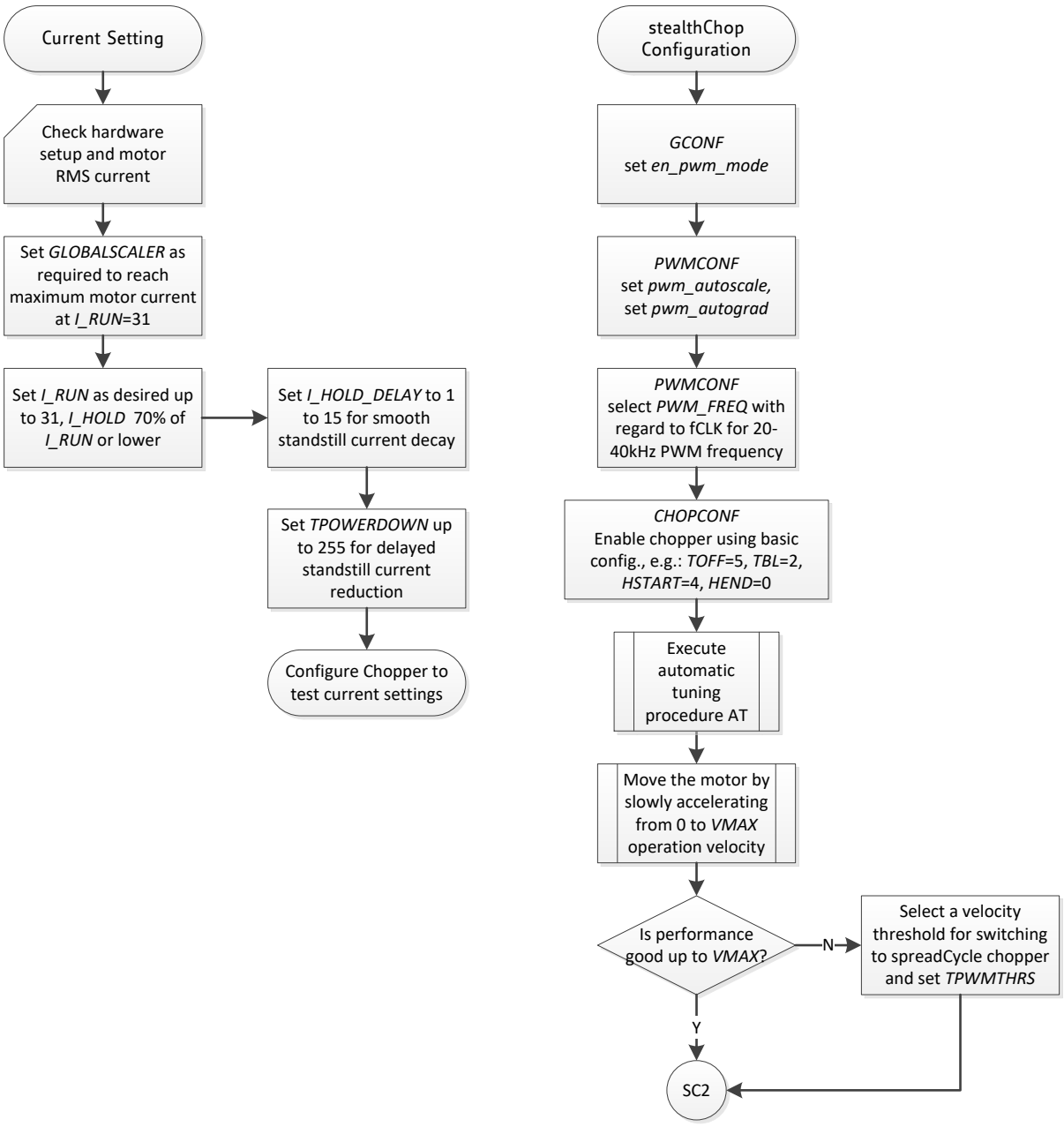


Figure 22.1 Current setting and first steps with stealthChop

TUNING STEALTHCHOP AND SPREADCYCLE

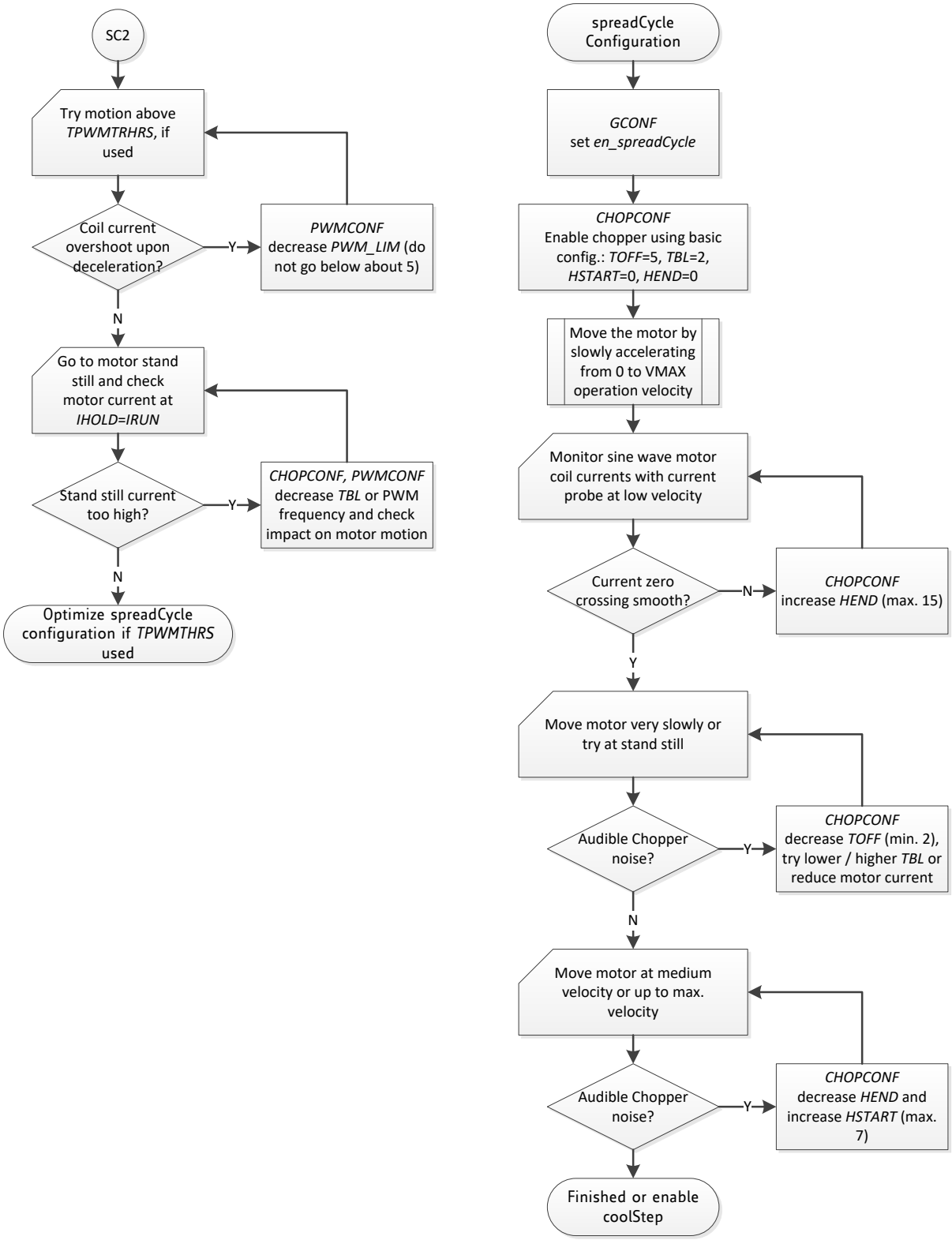


Figure 22.2 Tuning stealthChop and spreadCycle

MOVING THE MOTOR USING THE MOTION CONTROLLER

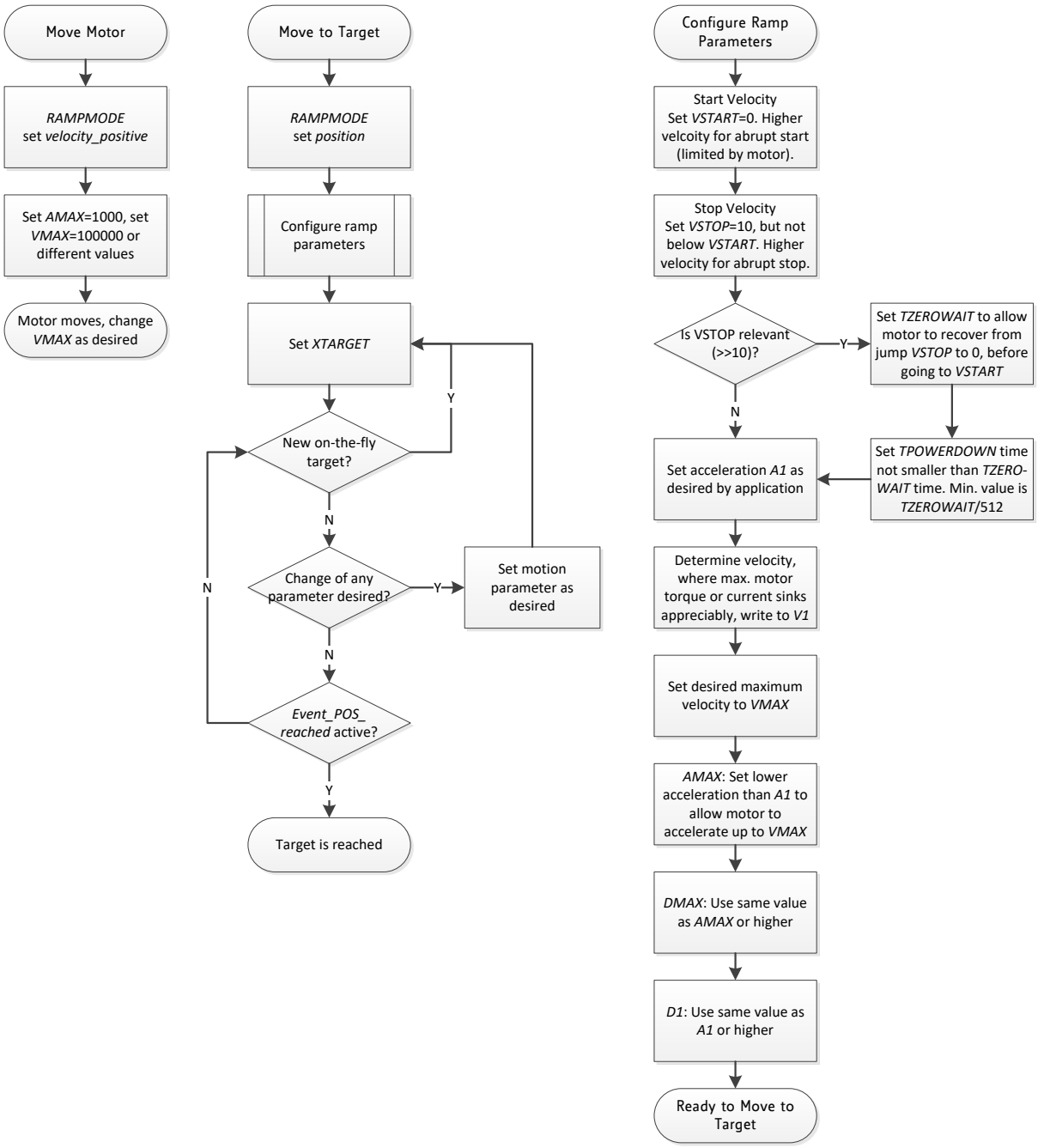


Figure 22.3 Moving the motor using the motion controller

ENABLING COOLSTEP (ONLY IN COMBINATION WITH SPREADCYCLE)

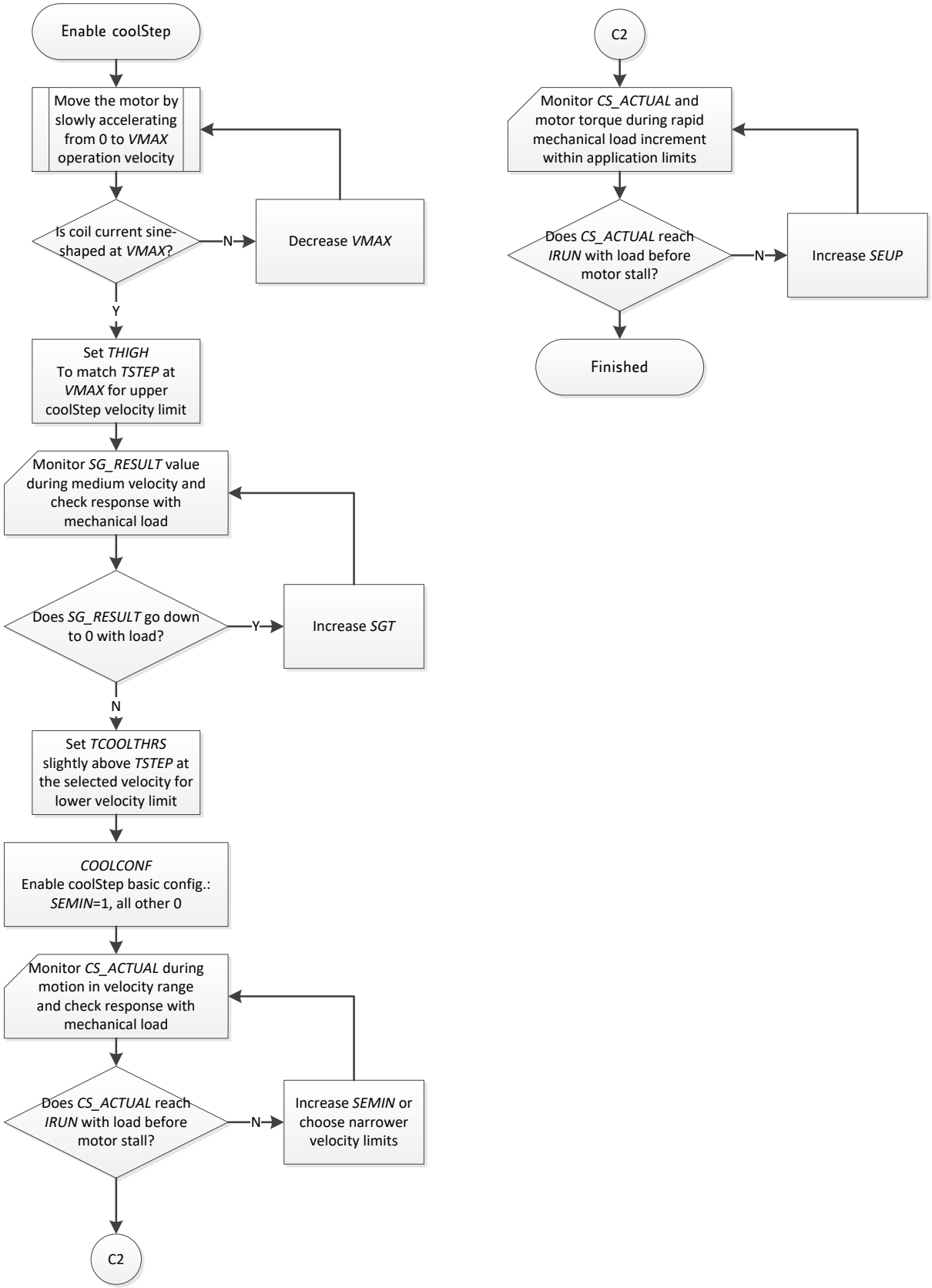


Figure 22.4 Enabling coolStep (only in combination with spreadCycle)

SETTING UP DCSTEP

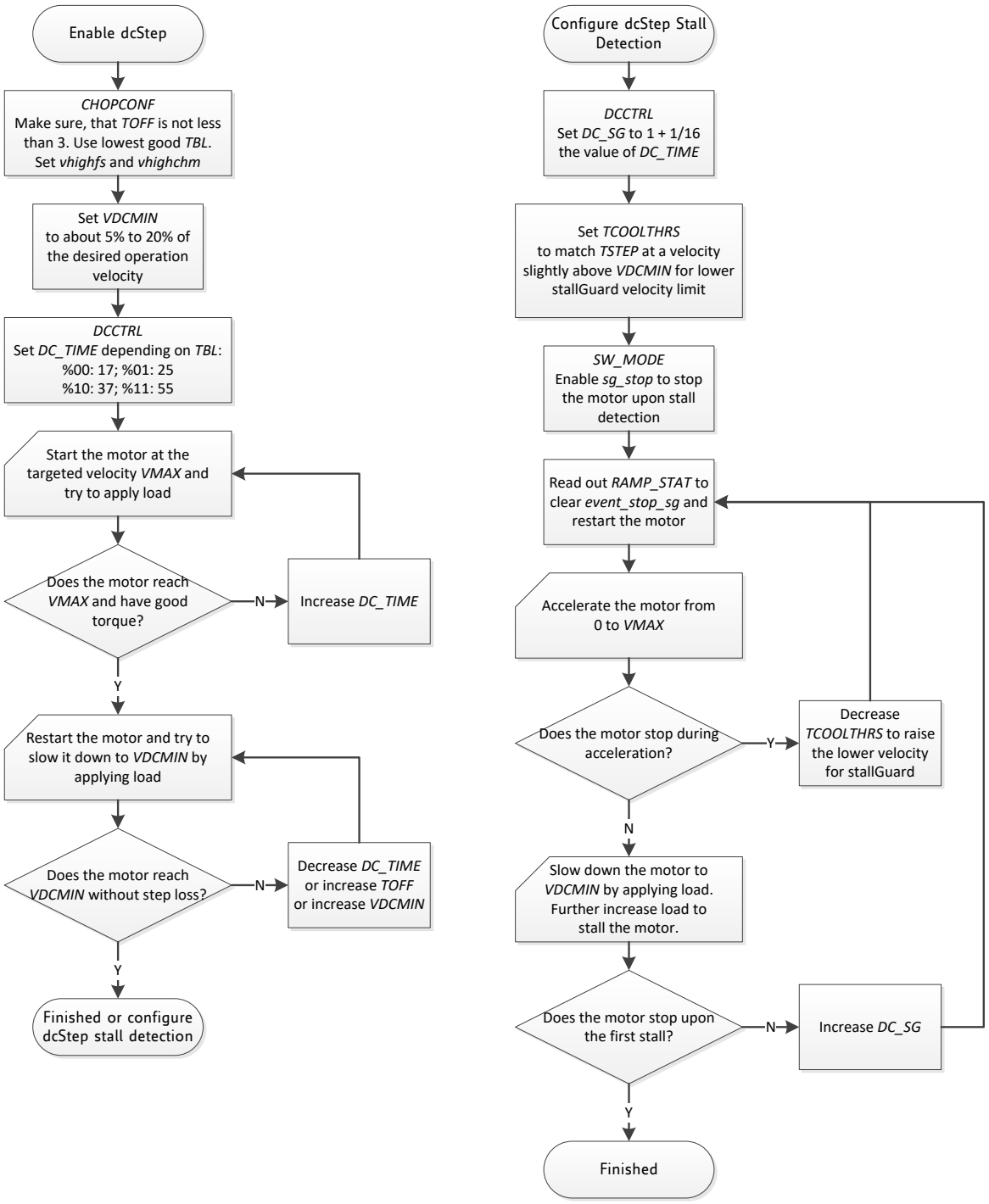


Figure 22.5 Setting up dcStep

23 Getting Started

Please refer to the TMC5161 evaluation board to allow a quick start with the device, and in order to allow interactive tuning of the device setup in your application. Chapter 22 will guide you through the process of correctly setting up all registers.

23.1 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 <60 RPM:

```
SPI send: 0x8A00020002;  // DRV_CONF: BBMTIME=2, BBMCLKS=0, OTSEL=2(136°C), DRVSTR.=0
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 PWM_CONF)
SPI send: 0x93000001F4;  // TPWM_THRS=500 yields a switching velocity about 35000 = ca. 30RPM
```

SPI sample sequence to enable and initialize the motion controller and move one rotation (51200 microsteps) using the ramp generator. A read access querying the actual position is also shown.

```
SPI send: 0xA4000003E8;   // A1      = 1 000 First acceleration
SPI send: 0xA50000C350;   // V1      = 50 000 Acceleration threshold velocity V1
SPI send: 0xA6000001F4;   // AMAX    = 500 Acceleration above V1
SPI send: 0xA700030D40;   // VMAX    = 200 000
SPI send: 0xA8000002BC;   // DMAX    = 700 Deceleration above V1
SPI send: 0xAA00000578;   // D1      = 1400 Deceleration below V1
SPI send: 0xAB0000000A;   // VSTOP   = 10 Stop velocity (Near to zero)
SPI send: 0xA000000000;   // RAMPMODE = 0 (Target position move)
// Ready to move!
SPI send: 0xADFFFF3800;   // XTARGET = -51200 (Move one rotation left (200*256 microsteps)
// Now motor 1 starts rotating
SPI send: 0x2100000000;   // Query XACTUAL – The next read access delivers XACTUAL
SPI read;                 // Read XACTUAL
```

For UART based operation it is important to make sure that the CRC byte is correct. The following example shows initialization for the driver with slave address 1 (NAI pin high). It programs the driver to spreadCycle mode and programs the motion controller for a constant velocity move and then read accesses the position and actual velocity registers:

```
UART write: 0x05 0x01 0xEC 0x00 0x01 0x00 0xC5 0xD3;  // TOFF=5, HEND=1, HSTR=4,
                                                      // TBL=2, MRES=0, CHM=0
UART write: 0x05 0x01 0x90 0x00 0x01 0x14 0x05 0xD8;  // IHOLD=5, IRUN=20, IHOLDDELAY=1
UART write: 0x05 0x01 0xA6 0x00 0x00 0x13 0x88 0xB4;  // AMAX=5000
UART write: 0x05 0x01 0xA7 0x00 0x00 0x4E 0x20 0x85;  // VMAX=20000
UART write: 0x05 0x01 0xA0 0x00 0x00 0x00 0x01 0xA3;  // RAMPMODE=1 (positive velocity)
// Now motor should start rotating
UART write: 0x05 0x01 0x21 0x6B;                      // Query XACTUAL
UART read 8 bytes;
UART write: 0x05 0x01 0x22 0x25;                      // Query VACTUAL
UART read 8 bytes;
```

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

24 Standalone Operation

For standalone operation, no SPI interface is required to configure the TMC5161. 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.

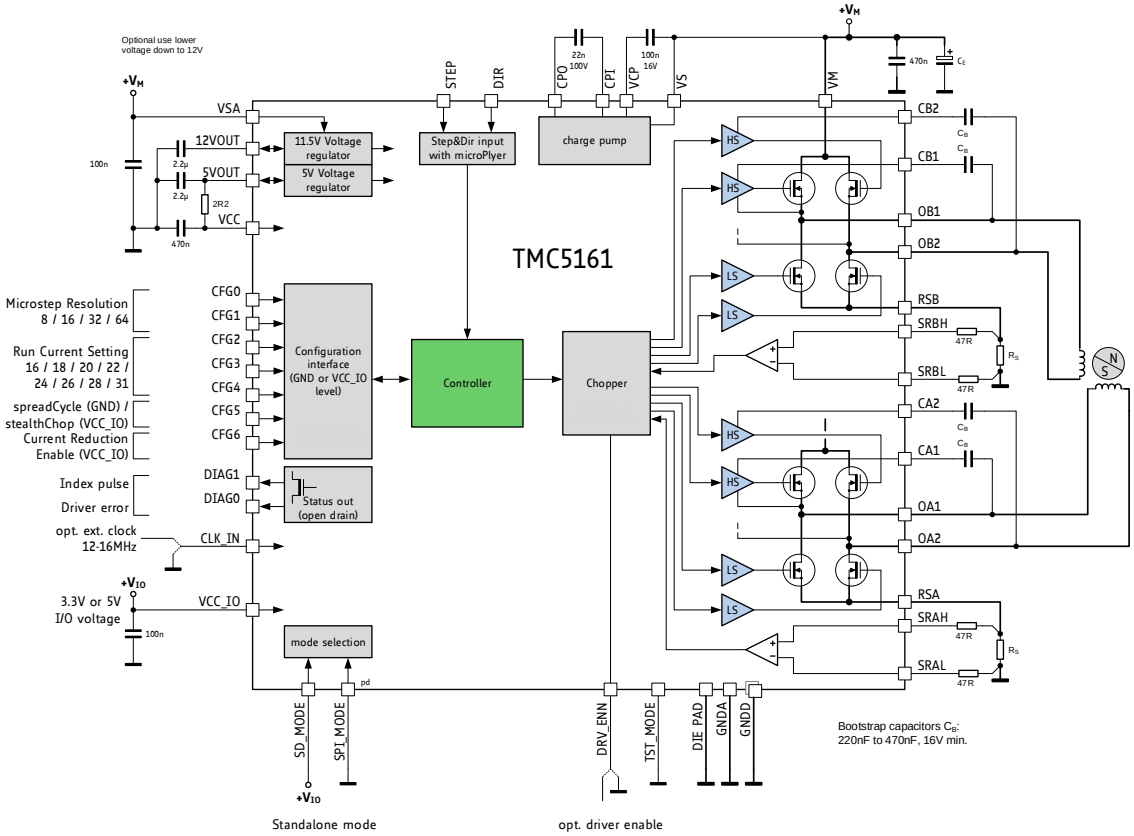


Figure 24.1 Standalone operation with TMC5161 (pins shown with their standalone mode names)

To activate standalone mode, tie pin SPI_MODE to GND and pin SD_MODE high. 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:

GCONF settings:
GCONF.diag0_error = 1: DIAG0 works in open drain mode and signals driver error.
GCONF.diag1_index = 1: DIAG1 works in open drain mode and signals microstep table index position.

The following settings are affected by the CFG pins in order 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.

25 External Reset

The chip is loaded with default values during power on via its internal power-on reset. In order to reset the chip to power on defaults, any of the supply voltages monitored by internal reset circuitry (VSA, +5VOUT or VCC_IO) must be cycled. VCC is not monitored. Therefore, VCC must not be switched off during operation of the chip. As +5VOUT is the output of the internal voltage regulator, it cannot be cycled via an external source except by cycling VSA. It is easiest and safest to cycle VCC_IO in order to completely reset the chip. Also, 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, all inputs must be driven low during this reset operation. When this is not possible, an input protection resistor may be used to limit current flowing into the related inputs.

In case, VCC becomes supplied by an external source, make sure that VCC is at a stable value above the lower operation limit once the reset ends. This normally is satisfied when generating a 3.3V VCC_IO from the +5V supply supplying the VCC pin, because it will then come up with a certain delay.

26 Clock Oscillator and Input

The clock is the timing reference for all functions: the chopper, the velocity, the acceleration control, 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 timing in case no external clock is easily available.

26.1 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.

26.2 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.

26.2.1 Considerations on the Frequency

A higher frequency allows faster step rates, faster SPI operation and higher chopper frequencies. On the other hand, it causes more power dissipation in the TMC5161 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.

27 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	55	V
Supply and bridge voltage short time peak	V_{VSMAX}		60	V
VSA when different from VS	V_{VSAMAX}	-0.5	60	V
Supply voltage V12	V_{12VOUT}	-0.5	14	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
Maximum steady state output current per MOSFET (RMS)			5	A
Peak MOSFET current (duty cycle thermally limited)			19	A
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 cont. 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}		300	V

*) 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.

28 Electrical Characteristics

28.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	T_J	-40	125	°C
Supply voltage for motor and bridge (max. 30V nominal supply voltage recommended for design guideline due to air gap distance on PCB / package)	V_{VS}	10	40	V
Supply voltage VSA	V_{VSA}	10	40	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)	V_{VS}	8		V
I/O supply voltage on VCC_IO	V_{VIO}	3.00	5.25	V
RMS motor coil current (short time)	I_{COIL}		4	A
RMS motor coil current (continuous rating)	I_{COIL}		3.5	A
Standstill current setting (RMS) for standstill at any microstep position (i.e. up to 1.41 times coil current)	I_{COIL}		3	A

28.2 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=25°C					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
RDS_{ON} Motor outputs highside / lowside	R_{ON}	I=1A		45	65	mΩ
Slope	t_{SLP0}	*DRVSTRENGTH=0 or 1*		10		ns
	t_{SLP2}	*DRVSTRENGTH=2*		5		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}	10			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	1.3	1.56	1.82	V
		S2G_LEVEL=15; VS<55V	1.0			V
		S2G_LEVEL=6; VS<52V	0.46	0.625	0.80	V
		S2G_LEVEL=6; *VS<55V*	0.20			V

Detector levels	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
V_{VSA} undervoltage threshold for RESET	V_{UV_VSA}	V_{VSA} rising	3.8	4.2	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 (control IC temperature)	t_{OTPW}	Temperature rising	100	120	140	°C
Overtemperature shutdown 136 °C (control IC temperature)	t_{OT136}	Temperature rising		136		°C
Overtemperature shutdown 143 °C (control IC temperature)	t_{OT143}	Temperature rising		143		°C
Overtemperature shutdown 150 °C (control IC temperature)	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 (low sensitivity) (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

28.3 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, number and position of vias for heat transfer, 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 overall power dissipation	P_D	stealthChop or spreadCycle, 30kHz chopper, 24V, 2.8A RMS	2.7	W
		stealthChop or spreadCycle, 30kHz chopper, 24V, 3.5A RMS	4.3	W
Thermal resistance junction to ambient on a multilayer board for output MOSFET measured on TMC6200-EVAL Dual signal, two internal power plane board (2s2p) (FR4, 35µm CU, 70mm x 133mm, d=1.5mm)	R_{TJA}	Single lowside MOSFET	50	K/W
		Two lowside MOSFETs OA1 & OA2 or OB1 & OB2	40	K/W
		Single highside MOSFET	40	K/W
		Two highside MOSFETs (OA1 & OA2 / OB1 & OB2)	30	K/W

Table 28.1 Thermal characteristics

The thermal performance in an actual layout can be tested by checking for the heat up, using a thermal camera. Heat distribution in stealthChop operation often is better and shows lower peak values than with spreadCycle. Carefully check heat up under different conditions and select a suiting overtemperature threshold, in case thermal protection is desired.

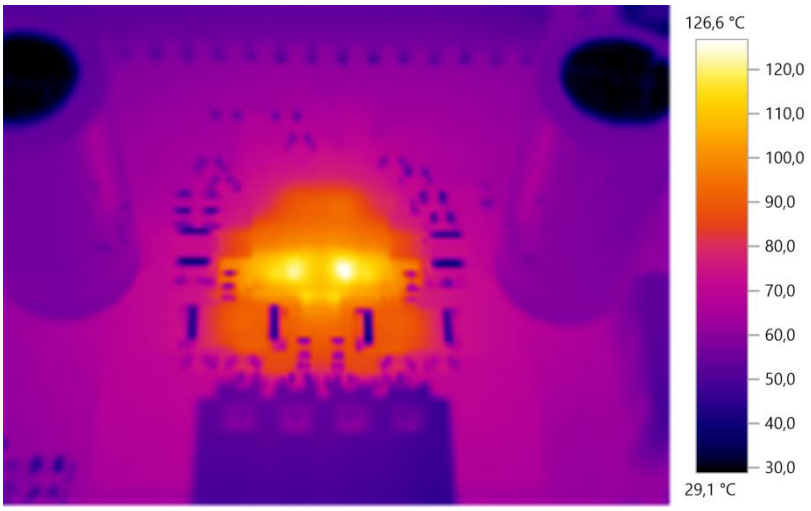


Figure 28.1 Thermal photo showing 3.5A RMS operation with spreadCycle on TMC6200-EVAL

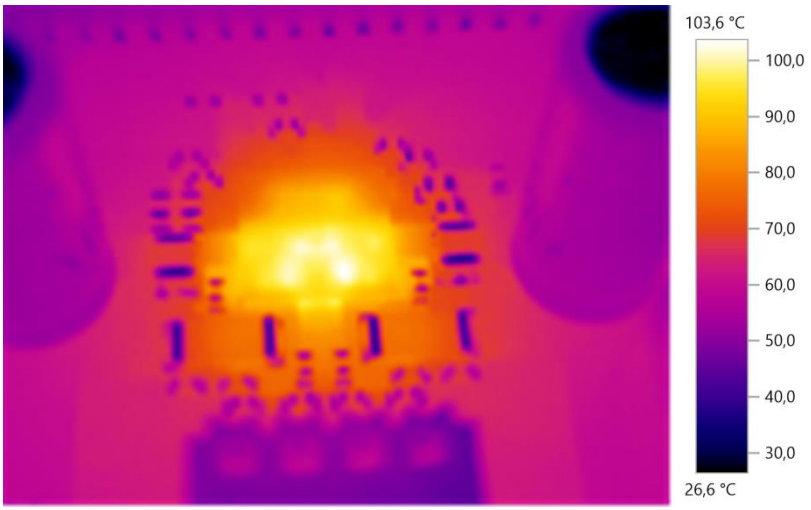


Figure 28.2 Thermal distribution 3.5A RMS operation running stealthChop

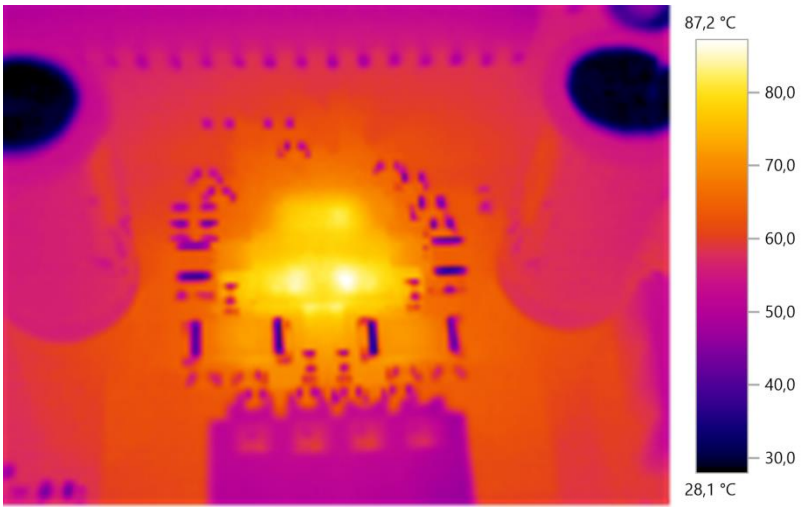


Figure 28.3 Thermal distribution 2.8A RMS operation running spreadCycle

Especially when device is to be operated near its maximum thermal limits, care has to be taken to provide a good thermal design of the PCB layout in order to avoid overheating of the power MOSFETs integrated into the TMC5161. As the TMC5161 uses discrete MOSFETs, power dissipation in each MOSFET needs to be looked over carefully. The actual values depend on the duty cycle and the die temperature. The thermal characteristics and the sample layout are intended as a guideline for your own board layout. In case, the driver is to be operated at high current levels, special care should be taken to spread the heat generated by the driver power bridges efficiently within the PCB.

Worst case power dissipation for the individual MOSFET is in standstill or at low velocity, with one coil operating at the maximum current, because one full bridge in this case takes over the full current. This scenario can be avoided with power down current reduction and current reduction in case slow movements (<1Hz fullstep frequency) are required. As single MOSFET temperatures cannot be monitored within the system, it is a good practice to react to the temperature pre-warning by reducing motor current, rather than relying on the overtemperature switch off.

The MOSFET (and bond wire) temperature should not exceed 150°C, despite temperatures up to 200°C will not immediately destroy the devices. But the package plastics will apply strain onto the bond wires, so that cyclic, repetitive exposure to temperatures above 150°C may damage the electrical contacts and increase contact resistance and eventually lead to contract break.

Check MOSFET temperature under worst case conditions not to exceed 150°C using a thermal camera to validate your layout. Please carefully check your layout against the sample layout or the layout of the TMC5161 evaluation board on the TRINAMIC website in order to ensure proper cooling of the IC!

29 Layout Considerations

29.1 Exposed Die Pads

The TMC5161 uses an exposed pad for each die to dissipate heat from the MOSFETs and the central control chip to the board. For best electrical and thermal performance, use wide traces on more than one PCB layer for all power signals, and interconnect them with a reasonable amount of solid, thermally conducting vias between each die attach pad and the supply plane, resp. the output traces.

The OAx and OBx outputs are directly connected electrically and thermally to the drain of the low side MOSFETs of the power stage in order to dissipate heat. A symmetrical, thermally optimized layout is required to ensure proper heat dissipation of all MOSFETs into the PCB. Use thick traces and areas for vertical heat transfer into the GND plane and provide enough vias for vertical heat transfer near the outputs. All high side MOSFETs are connected and cooled via the VM bar. Provide a solid, thermally conductive connection to the supply plane and additional areas and vias for cooling.

The printed circuit board should have a solid ground plane spreading heat horizontally into the board and providing for a stable GND reference. All signals of the TMC5161 are referenced to GND. Directly connect all GND pins to a common ground area.

The switching motor coil outputs have a high dV/dt, so stray capacitive coupling into high-impedance signals can occur, if the motor traces are parallel to other traces over long distances.

29.2 Power Supply Pins

Both, the VM pins and Oxx motor outputs, as well as the RSA and RSB pins conduct the full motor current for a limited amount of time during each chopper cycle. Due to the resistance of bond wires connected to these pins, the pins heat up. Therefore, it is essential to use a wide PCB trace for cooling and in order to avoid additional heat up of the pins caused by PCB trace resistance. Failure to do so might affect reliability; despite heat-up of bond wires might not be visible with a thermal camera.

29.3 Wiring GND

All signals of the TMC5161 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 sense resistors and their GND connection near to the TMC5161 using a common GND plane in order to avoid ringing leading to GND differences and to dangerous inductive peak voltages.

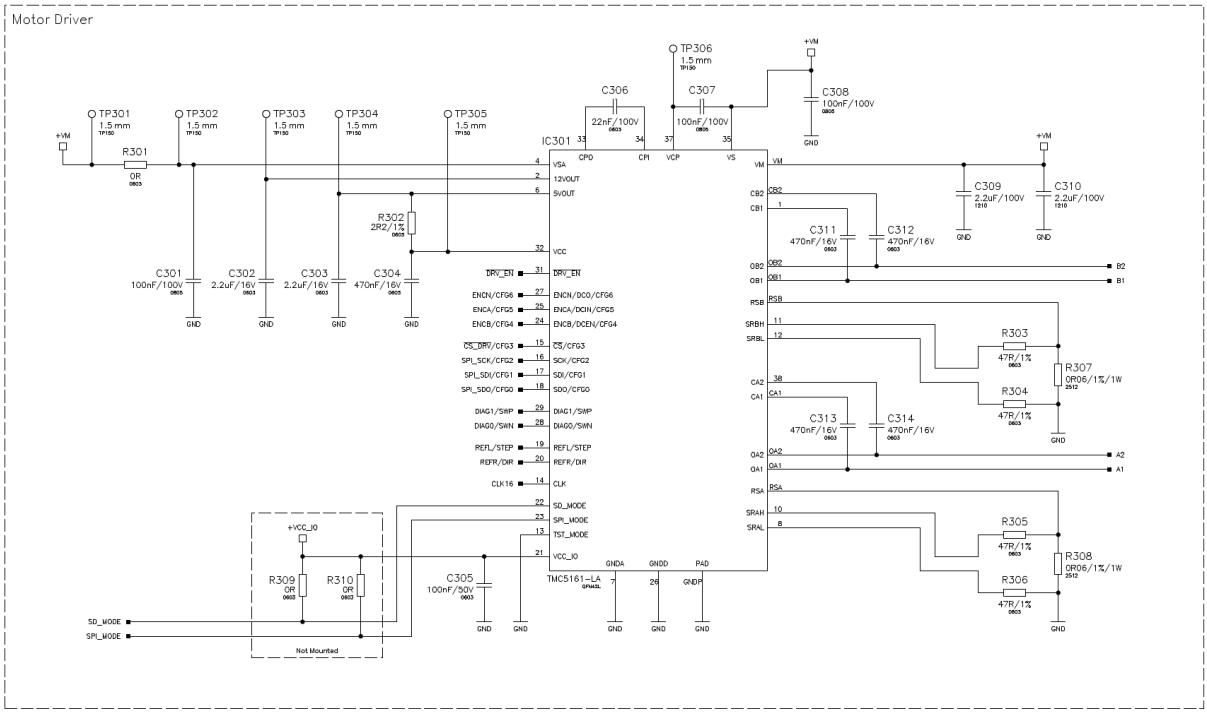
29.4 Supply Filtering

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 in order 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.

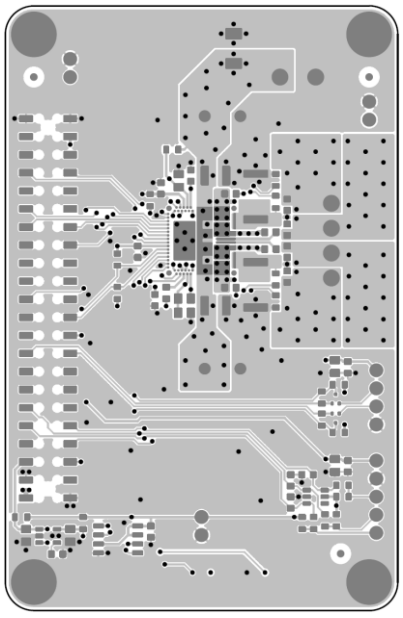
29.5 Layout Example

Layout Hints
- Take care for good thermal power dissipation of all VM, OAx and OBx pads into the PCB.
- Tune the power bridge layout for minimum loop inductivity. A compact layout is best.
- Minimize the distance between the sense resistors and BRA/BRB terminals and connect the sense resistors and the TMC5161 GND connections directly to the same GND plane using enough vias.
- Add MOSFET bridge output capacitors (470pF - 1nF from each output to GND) to reduce ringing of the outputs during switching events near the motor connector.

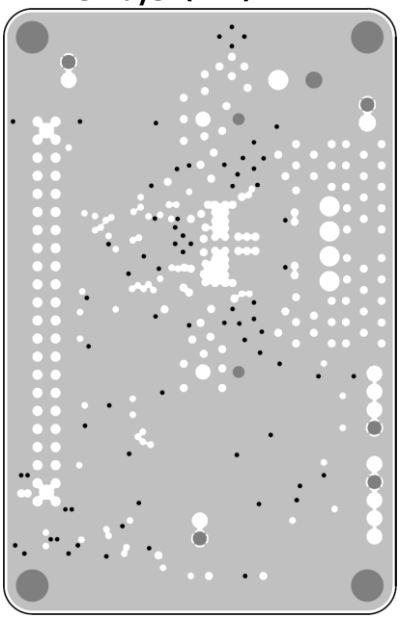
Schematic (TMC5161+sense resistors shown)



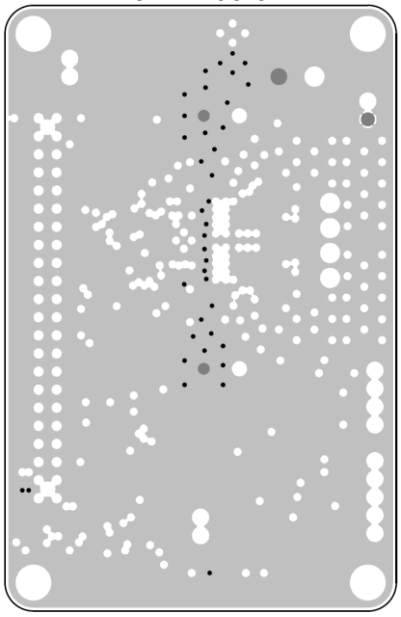
1- Top Layer (assembly side)



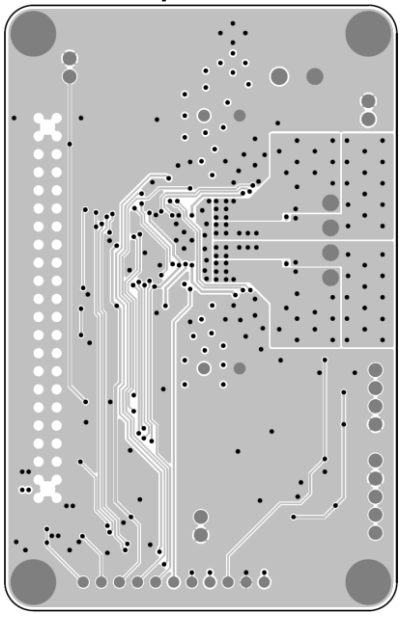
2- Inner Layer (GND)



3- Inner Layer (supply VS)



4- Bottom Layer



Components

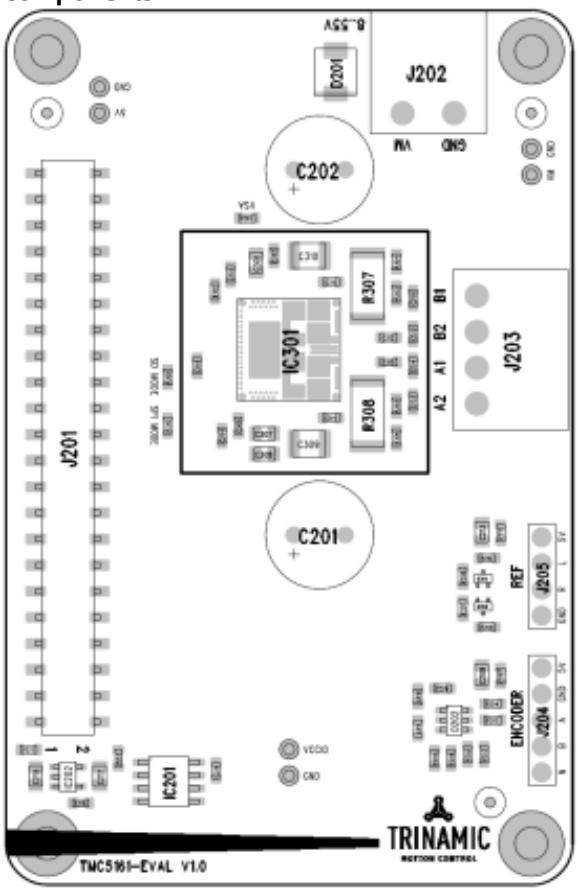


Figure 29.1 Layout example

30 Package Mechanical Data

30.1 Dimensional Drawings aQFN

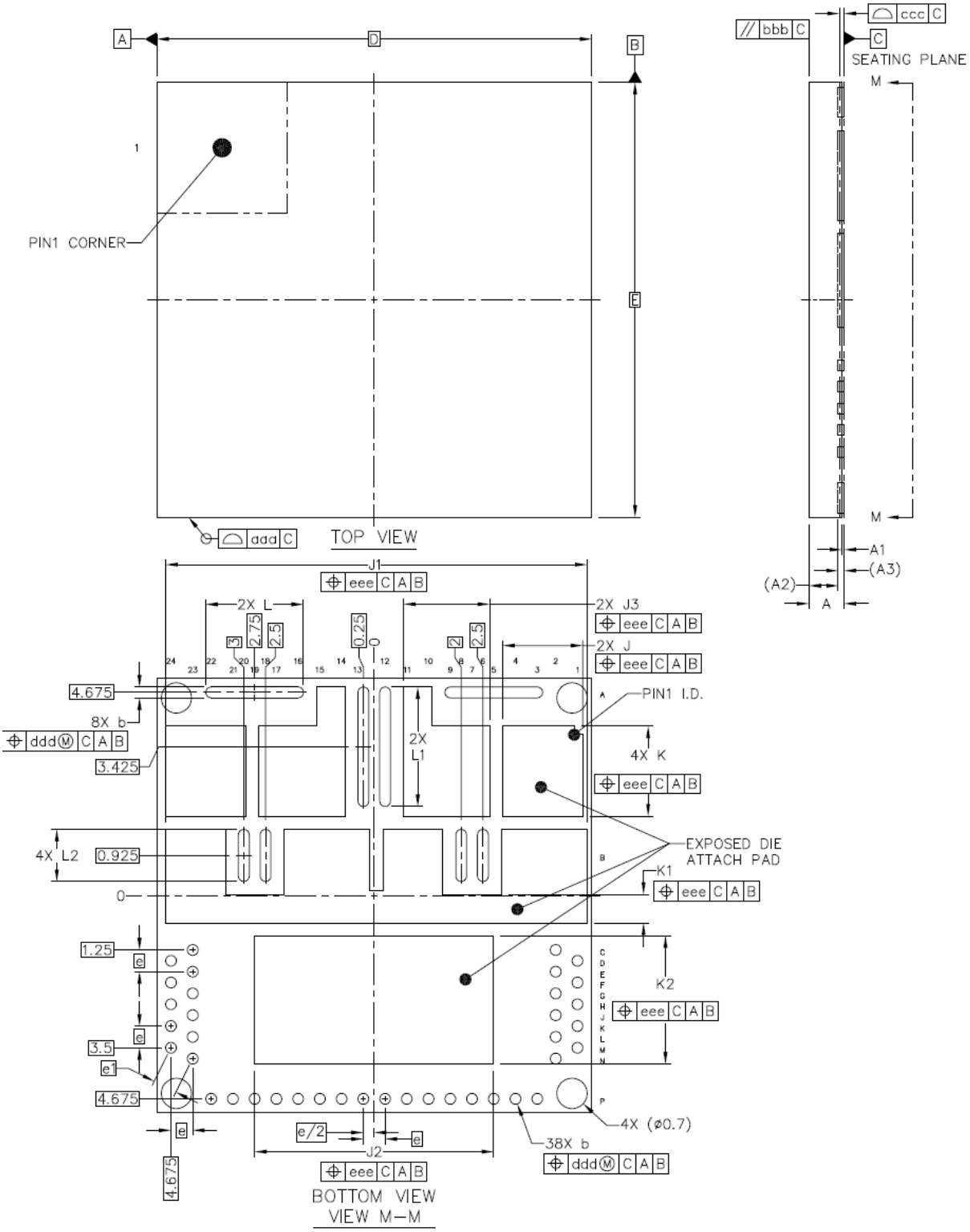


Figure 30.1 Dimensional drawings special aQFN

Parameter	Ref	Min	Nom	Max
total thickness	A	-	-	0.85
stand off	A1	0.02	0.05	0.08
mold thickness	A2		0.675	
lead frame thickness	A3		0.13	
lead width	b	0.2	0.25	0.3
body size X	D		10.0	
body size Y	E		10.0	
lead pitch	e		0.5	
lead pitch	e1		0.559	
gap around exposed die pads to neighbor pad	e2		0.3	
exposed die pad size X	J	1.745	1.845	1.945
exposed die pad size X	J1	9.6	9.7	9.8
exposed die pad size X	J2	5.4	5.5	5.6
exposed die pad size X	J3	1.88	1.98	2.08
exposed die pad size Y	K	1.975	2.075	2.175
exposed die pad size Y	K1	0.55	0.65	0.75
exposed die pad size Y	K2	2.85	2.95	3.05
lead length	L	2.2	2.25	2.3
lead length	L1	2.7	2.75	2.8
lead length	L2	1.15	1.2	1.25
package edge tolerance	aaa		0.1	
mold flatness	bbb		0.2	
coplanarity	ccc		0.1	
lead offset	ddd		0.08	
exposed pad offset	eee		0.1	

Attention
Due to the complex footprint, please use the footprint samples available from the TRINAMIC website for different PCB software.

30.2 Package Codes

Type	Package	Temperature range	Code & marking
TMC5161-AA	AQFN-10x10 (RoHS)	-40°C ... +125°C	TMC5161-AA

31 Design Philosophy

The TMC50XX and TMC51XX family brings premium functionality, reliability and coherence previously reserved to costly motion control units to smart applications. Integration at street level cost was possible by squeezing know-how into a few mm² of layout using one of the most modern smart power processes. The ICs comprise all the knowledge gained from designing motion controller and driver chips and complex motion control systems for more than 20 years. We are often asked if our motion controllers contain software – they definitely do not. The reason is that sharing resources in software leads to complex timing constraints and can create interrelations between parts which should not be related. This makes debugging of software so difficult. Therefore, the IC is completely designed as a hardware solution, i.e. each internal calculation uses a specially designed dedicated arithmetic unit. The basic philosophy is to integrate all real-time critical functionality in hardware, and to leave additional starting points for highest flexibility. Parts of the design go back to previous ICs, starting from the TMC453 motion controller developed in 1997. Our deep involvement, practical testing and the stable team ensure a high level of confidence and functional safety.

Bernhard Dwersteg, CTO and founder

32 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.

33 ESD Sensitive Device

The TMC5161 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.

Note: In a modern SMD manufacturing process, ESD voltages well below 100V are standard. A major source for ESD is hot-plugging the motor during operation. As the power MOSFETs are discrete devices, the device in fact is very rugged concerning any ESD event on the motor outputs. All other connections are typically protected due to external circuitry on the PCB.

34 Table of Figures

35 Revision History

Version	Date	Author BD= Bernhard Dwersteg	Description
V0.94	2018-JUN-30	BD	First version of datasheet based on datasheet TMC5160 V1.06.
V1.00	2018-AUG-03	BD	Added product pic, adapted RDSon, minor corrections

Table 35.1 Document Revisions

36 References

[TMC5161-EVAL] TMC5161-EVAL Manual
[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 TMC5161_Calculations.xlsx