POWER DRIVER FOR STEPPER MOTORS INTEGRATED CIRCUITS

TMC5130A-TA DATASHEET

Universal high voltage controller/driver for two-phase bipolar stepper motor. stealthChop™ for quiet movement. Integrated MOSFETs for up to 2 A motor current per coil. With Step/Dir Interface and SPI.

APPLICATIONS

Textile, Sewing Machines
Factory Automation
Lab Automation
Liquid Handling
Medical
Office Automation
CCTV, Security
ATM, Cash recycler
POS
Pumps and Valves
Heliostat Controller

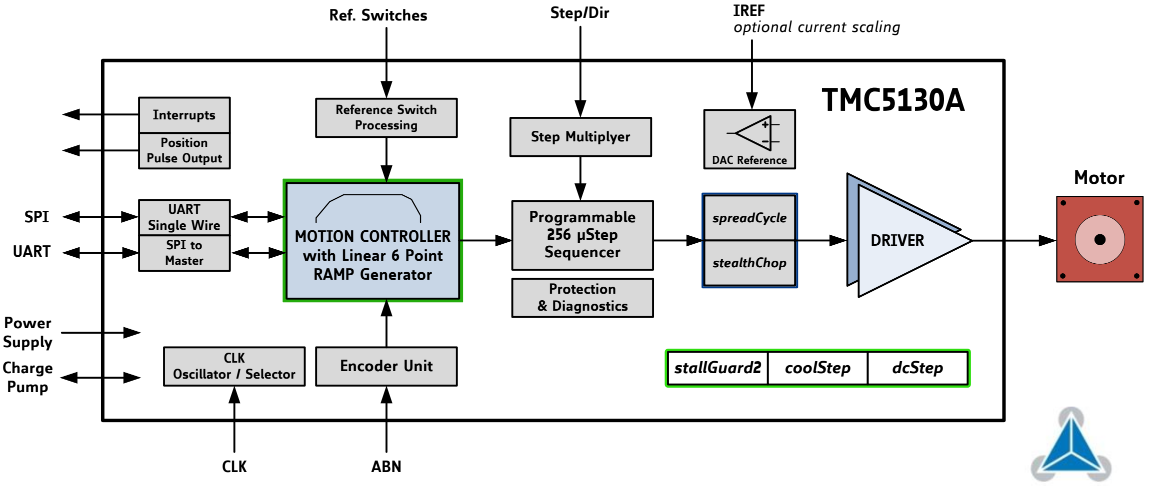
FEATURES AND BENEFITS

2-phase stepper motors
Drive Capability up to 2A coil current (2.5A peak)
Motion Controller with **sixPoint™** ramp
Step/Dir Interface with microstep interpolation **microPlyer™**
Voltage Range 4.75… 46V DC
SPI & Single Wire UART
Encoder Interface and **2x Ref.-Switch Input**
Highest Resolution 256 microsteps per full step
stealthChop™ for extremely quiet operation and smooth motion
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%
Integrated Current Sense Option
Passive Breaking and freewheeling mode
Full Protection & Diagnostics
Compact Size 9x9mm^2 TQFP48 package

DESCRIPTION

The TMC5130A is a high performance stepper motor controller and driver IC with serial communication interfaces. It combines a flexible ramp generator for automatic target positioning with industries' most advanced stepper motor driver. Based on TRINAMICs sophisticated stealthChop chopper, the driver ensures absolutely noiseless operation combined with maximum efficiency and best motor torque. High integration, high energy efficiency and a small form factor enable miniaturized and scalable systems for cost effective solutions. The complete solution reduces learning curve to a minimum while giving best performance in class.

BLOCK DIAGRAM

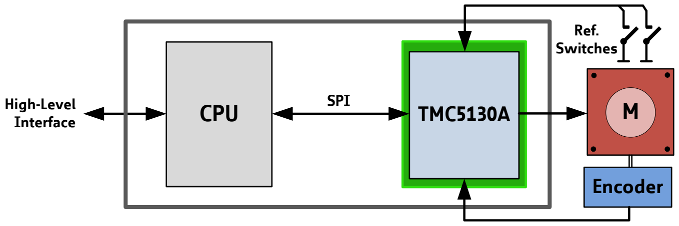


TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

APPLICATION EXAMPLES: HIGH VOLTAGE – MULTIPURPOSE USE

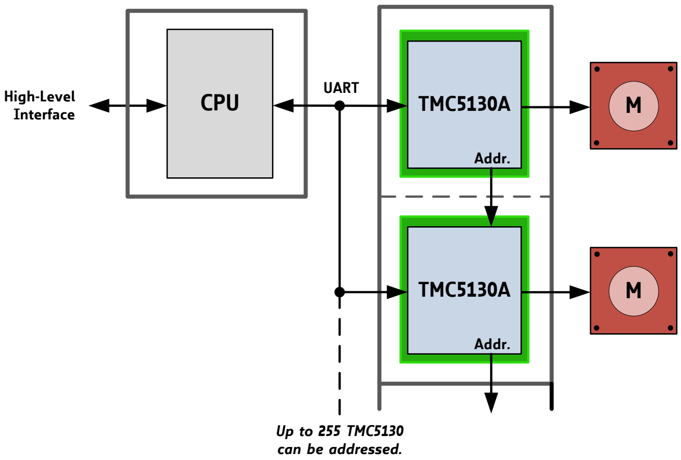
The TMC5130A scores with complete motion controlling features, integrated power stages, and power density. It offers a versatility that covers a wide spectrum of applications from battery powered systems up to embedded applications with 2A motor current per coil. The TMC5130A contains the complete intelligence which is required to drive a motor. Receiving target positions the TMC5130A manages motor movement. Based on TRINAMICs unique features stallGuard2, coolStep, dcStep, spreadCycle, and stealthChop, the TMC5130A 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 TMC5130A 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.

MINIATURIZED DESIGN FOR ONE STEPPER MOTOR



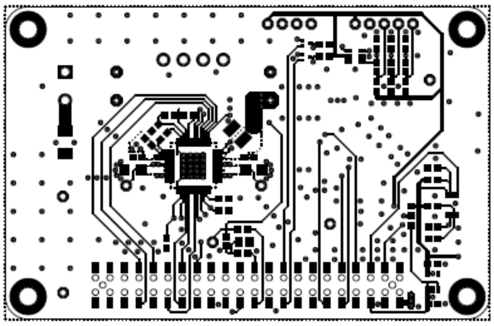
An ABN encoder interface with scaler unit and two reference switch inputs are used to control motor movement.

COMPACT DESIGN FOR UP TO 255 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. The CPU-board and the controller / driver boards are highly economical and space saving.

TMC5130-EVAL EVALUATION BOARD



The TMC5130-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 TMC5130 evaluation board system consists of three parts: STARTRAMPE (base board), ESELSBRÜCKE (connector board including several test points), and TMC5130-EVAL.

ORDER CODES

Order code	Description	Size [mm²]
TMC5130A-TA	1-axis dcStep, coolStep, and stealthChop controller/driver; TQFP48	9 x 9
TMC5130-EVAL	Evaluation board for TMC5130A two phase stepper motor controller/driver	85 x 55
STARTRAMPE	Baseboard for TMC5130-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 TMC5130A motion controller and driver chip is an intelligent power component interfacing between CPU and stepper motor. All stepper motor logic is completely within the TMC5130A. No software is required to control the motor – just provide target positions. The TMC5130A 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 TMC5130A uses stealthChop, dcStep, coolStep, and stallGuard2 automatically to optimize every motor movement.

THE TMC5130A OFFERS THREE BASIC MODES OF OPERATION:

MODE 1: Full Featured Motion Controller & Driver

All stepper motor logic is completely within the TMC5130A. 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 TMC5130A takes care of intelligent current and mode control and delivers feedback on the state of the motor. The microPlyer automatically smoothens motion. Leave open SD_MODE and SPI_MODE.

MODE 3: Simple Step & Direction Driver

The TMC5130A 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 TMC5130A. Optional feedback signals allow error detection and synchronization. Enable this mode by tying low pin SPI_MODE.

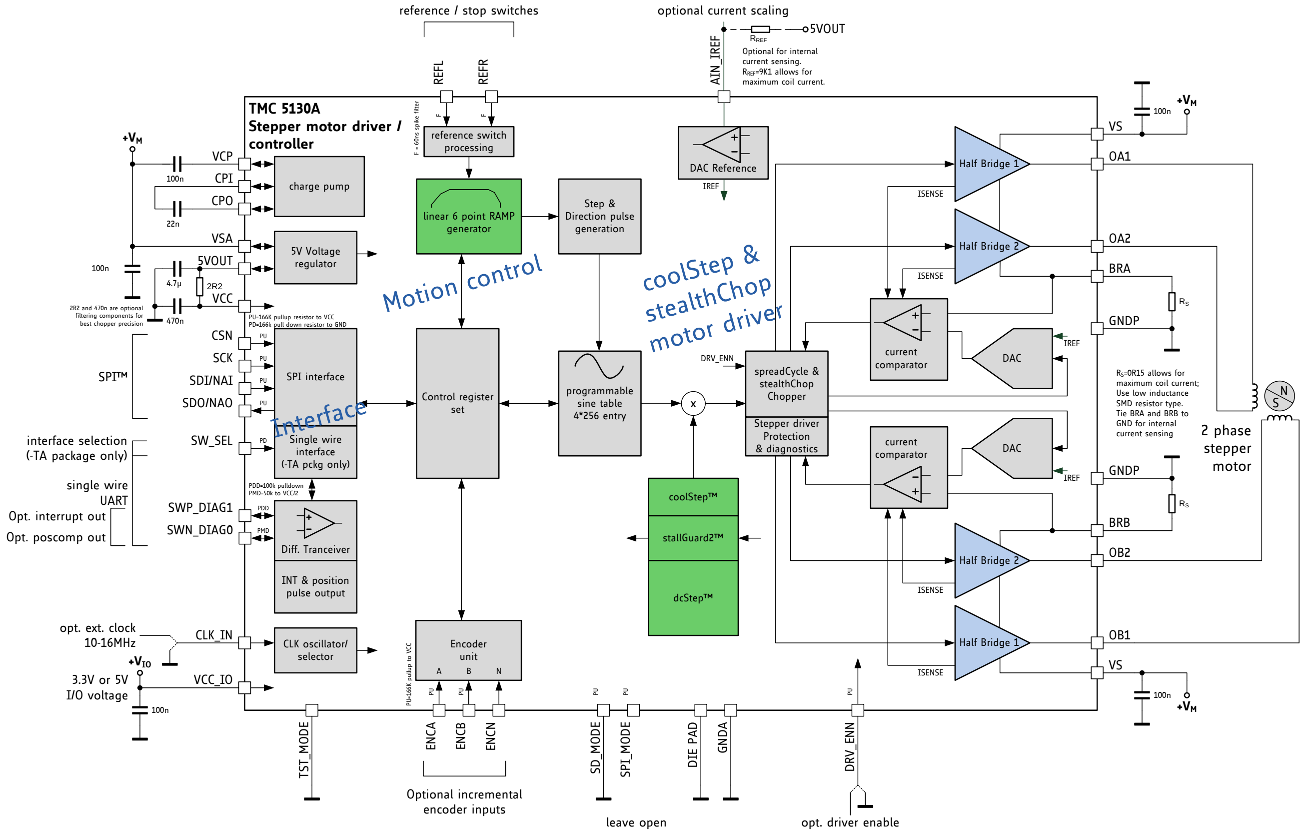


Figure 1.1 TMC5130A basic application block diagram with motion controller

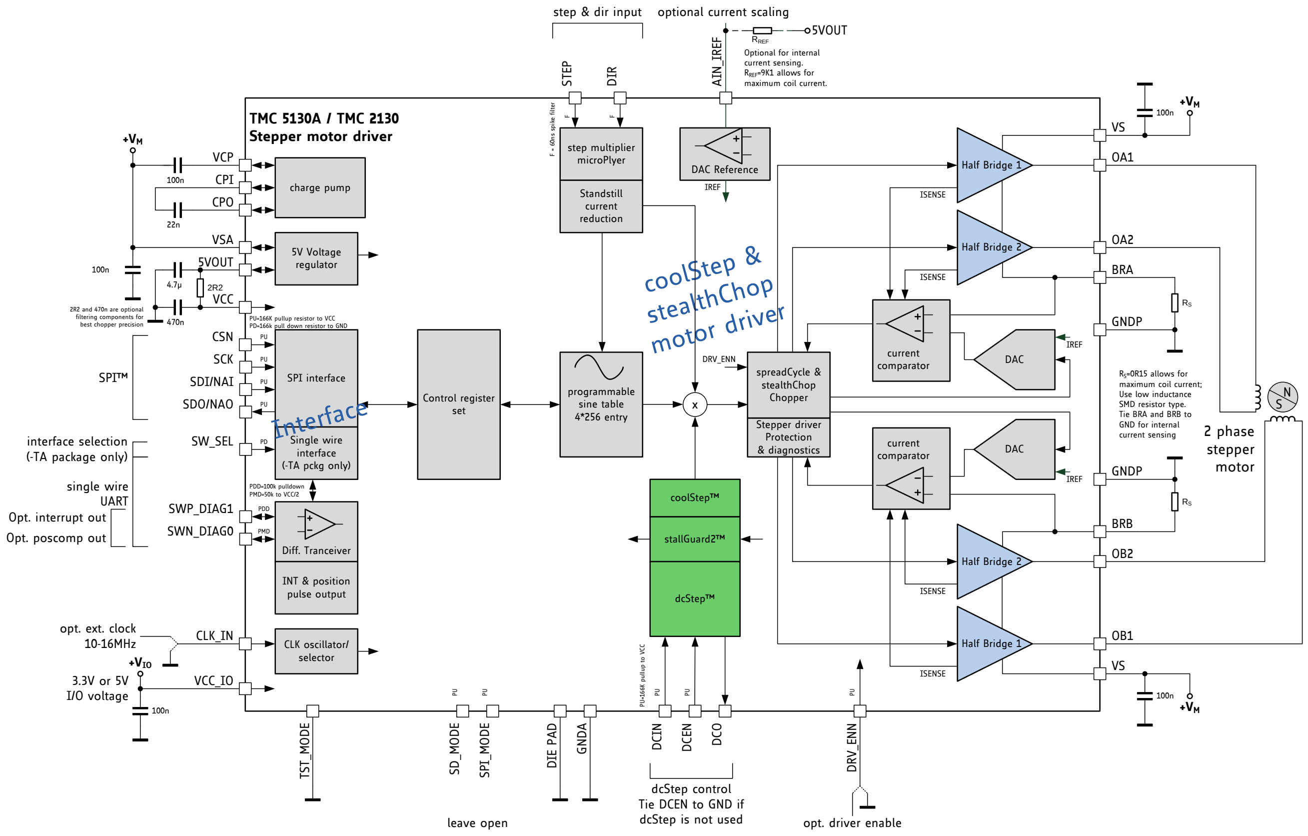


Figure 1.2 TMC5130A STEP/DIR application diagram

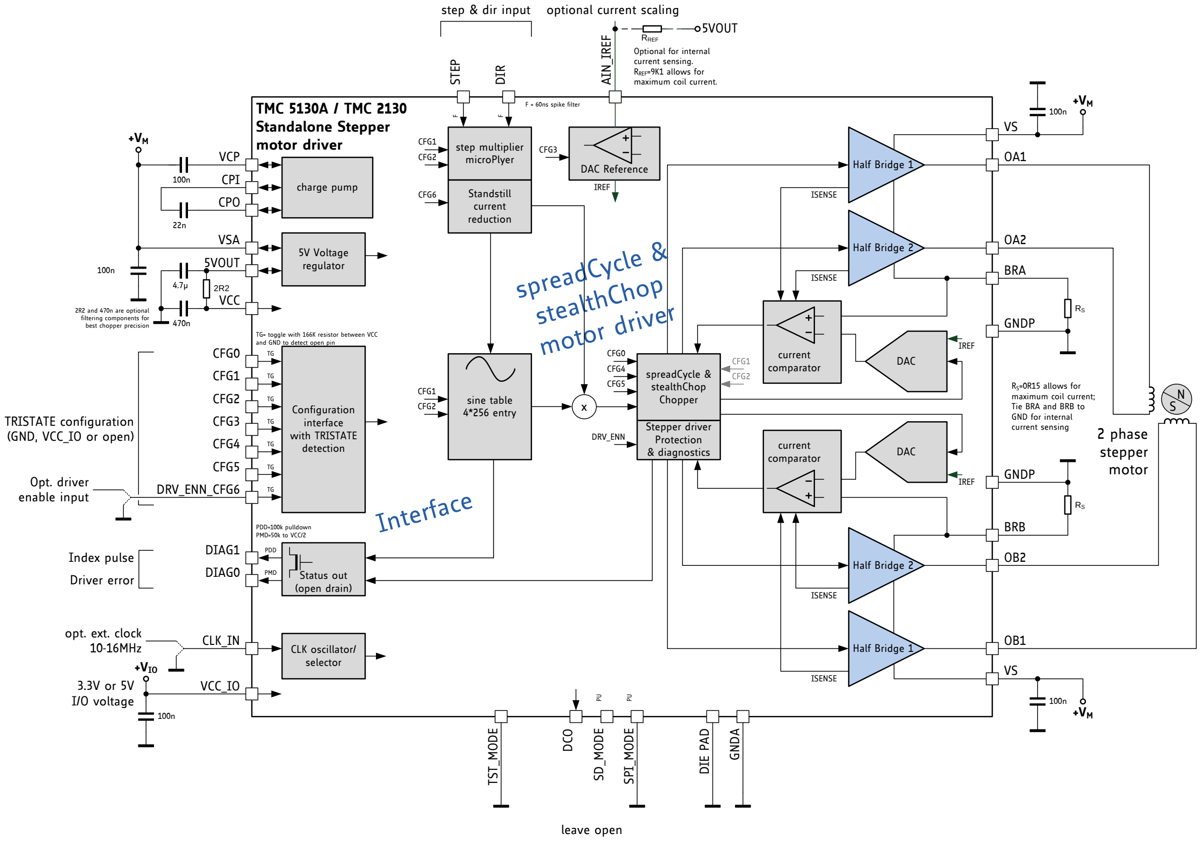


Figure 1.3 TMC5130A standalone driver application diagram

1.1 Key Concepts

The TMC5130A 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.

stealthChop™	No-noise, high-precision chopper algorithm for inaudible motion and inaudible standstill of the motor.
spreadCycle™	High-precision chopper algorithm available as an alternative to the traditional constant off-time algorithm.
dcStep™	Load dependent speed control. The motor moves as fast as possible and never loses a step.
stallGuard2™	High-precision load measurement using the back EMF on the motor coils.
coolStep™	Load-adaptive current control which reduces energy consumption by as much as 75%.
microPlyer™	Microstep interpolator for obtaining increased smoothness of microstepping when using the STEP/DIR interface.

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 TMC5130A 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 TMC5130A 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 SWIOP and SWION) or single wire interfacing (leaving open SWION). It can be driven by any standard UART. No baud rate configuration is required.

1.3 Software

From a software point of view the TMC5130A 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. During microstepping, 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 stealthChop Driver with Programmable Microstepping Wave

Current into the motor coils is controlled using a cycle-by-cycle chopper mode. Up to three chopper modes are available: a traditional constant off-time mode and the spreadCycle mode as well as the unique stealthChop. The constant off-time mode provides higher torque at highest velocity, while spreadCycle mode offers smoother operation and greater power efficiency over a wide range of speed and load. The spreadCycle chopper scheme automatically integrates a fast decay cycle and guarantees smooth zero crossing performance. In contrast to the other chopper modes, stealthChop is a voltage chopper based principle. It guarantees that the motor is absolutely quiet in standstill and in slow motion, except for noise generated by ball bearings. The extremely smooth motion is beneficial for many applications.

Programmable microstep shapes allow optimizing the motor performance.

Benefits of using stealthChop:

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

1.6 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.

1.7 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 1.4 shows the efficiency gain of a 42mm stepper motor when using coolStep compared to standard operation with 50% of torque reserve. coolStep is enabled above 60RPM in the example.

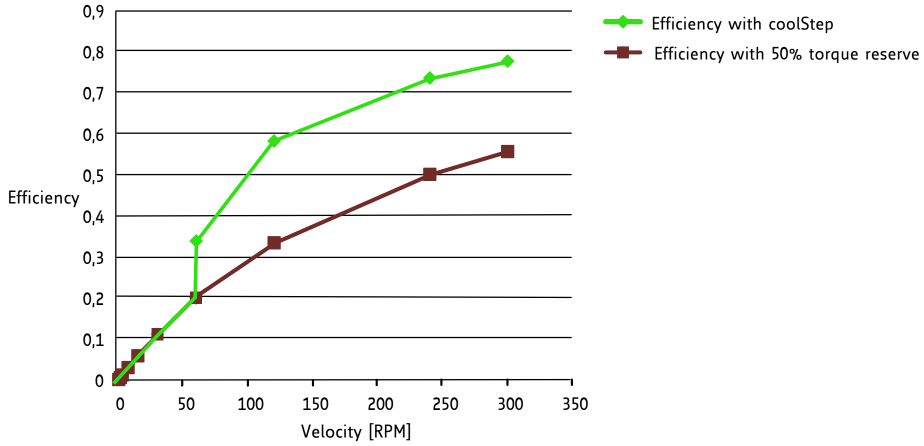


Figure 1.4 Energy efficiency with coolStep (example)

1.8 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

1.9 Encoder Interface

The TMC5130A provides an encoder interface for external incremental encoders. The encoder can be used for homing of the motion controller (alternatively to reference switches) and for consistency checks on-the-fly between encoder position and ramp generator position. 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

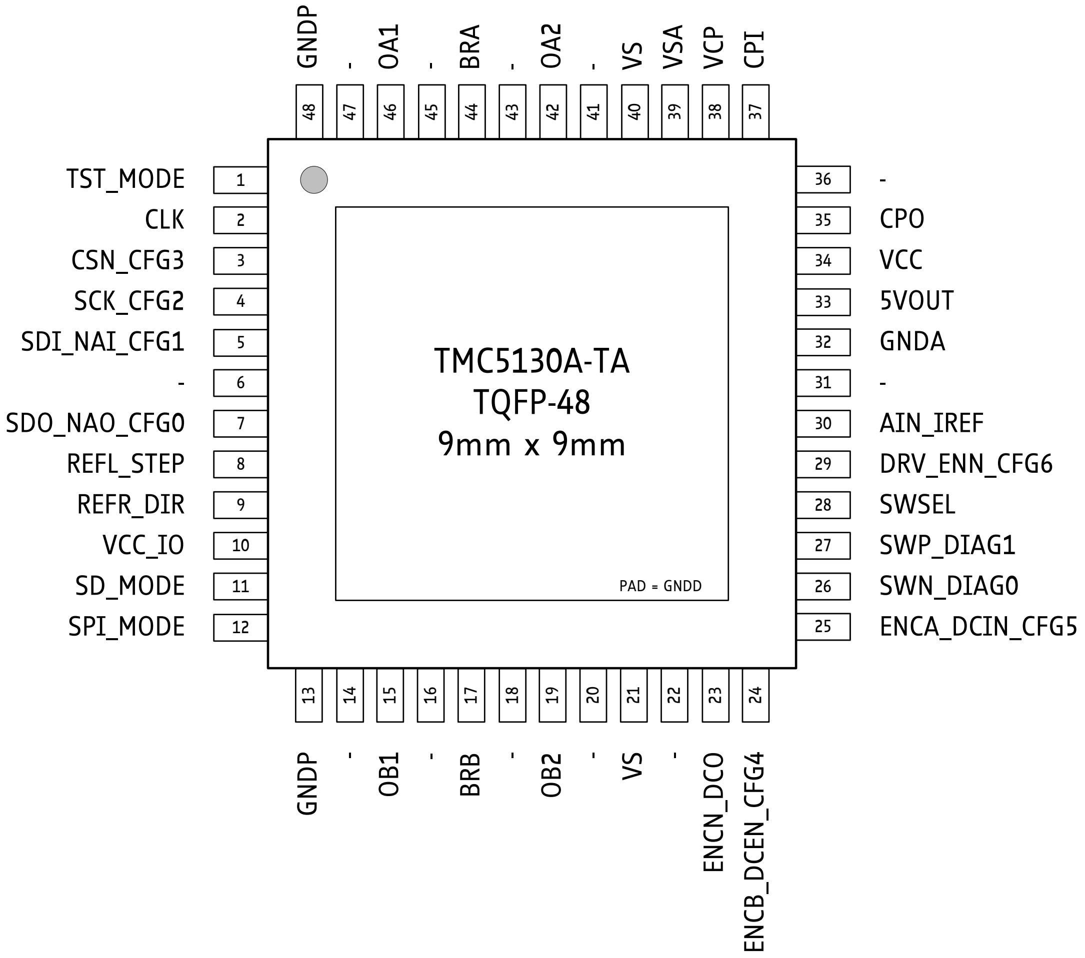


Figure 2.1 TMC5130A-TA package and pinning TQFP-EP 48 (7x7mm body, 9x9mm with leads)

2.2 Signal Descriptions

Pin	Number	Type	Function
TST_MODE	1	DI	Test mode input. Tie to GND using short wire.
CLK	2	DI	CLK input. Tie to GND using short wire for internal clock or supply external clock.
CSN_CFG3	3	DI	SPI chip select input (negative active) or configuration input
SCK_CFG2	4	DI	SPI serial clock input and configuration input
SDI_NAI_ CFG1	5	DI	SPI data input and configuration input and next address input for single wire interface
N.C.	6, 31, 36		Unused pins; connect to GND for compatibility to future versions.
SDO_NAO_ CFG0	7	DIO	SPI data output (tristate) or configuration input or next address output for single wire interface
REFL_STEP	8	DI	Left reference input when SPI_MODE=1 and SD_MODE=0 STEP input when SD_MODE=1 or SPI_MODE=0

Pin	Number	Type	Function
REFR_DIR	9	DI	Right reference input when SPI_MODE=1 and SD_MODE=0 DIR input when SD_MODE=1 or SPI_MODE=0
VCC_IO	10		3.3V to 5V IO supply voltage for all digital pins.
SD_MODE	11	DI (pu)	Mode selection input with pullup resistor. When tied low, the internal ramp generator generates step pulses. When tied high, the STEP/DIR inputs control the driver. Integrated pullup resistor.
SPI_MODE	12	DI (pu)	Mode selection input with pullup resistor. When tied low, the chip is in standalone mode and pins have their CFG functions. When tied high, the SPI and UART interface are available for control. Integrated pullup resistor.
GNDP	13, 48		Power GND. Connect to GND plane near pin.
DNC.	14, 16, 18, 20, 22, 41, 43, 45, 47		Do not connect these pins. Provided to increase creeping distance on PCB in order to allow higher supply voltage without coating.
OB1	15		Motor coil B output 1
BRB	17		Sense resistor connection for coil B. Place sense resistor to GND near pin. An additional 100nF capacitor to GND (GND plane) is recommended for best performance.
OB2	19		Motor coil B output 2
VS	21, 40		Motor supply voltage. Provide filtering capacity near pin with short loop to nearest GNDP pin (respectively via GND plane).
ENCN_DCO	23	DIO	Encoder N-channel or dcStep ready output when SD_MODE=1. With SD_MODE=0, pull to GND or VCC_IO, if the pin is not used.
ENCB_DCEN_ CFG4	24	DI	Encoder B-channel or dcStep enable input when SD_MODE=1 or configuration input. With SD_MODE=1, tie to GND for normal operation (no dcStep).
ENCA_DCIN_ CFG5	25	DI	Encoder A-channel or dcStep gating input for axis synchronization when SD_MODE=1 or configuration input
SWN_DIAG0	26	DIO	Single wire I/O (negative) when SWSEL=1, otherwise diagnostics output DIAG0. Interrupt or STEP output in motion controller mode. Use external pullup resistor with 47k or less in open drain mode.
SWP_DIAG1	27	DIO	Single wire I/O (positive) when SWSEL=1, otherwise diagnostics output DIAG1. Position compare or DIR output in motion controller mode. Use external pullup resistor with 47k or less in open drain mode.
SWSEL	28	DI (pd)	Single wire interface select input, tie high for use of single wire interface (only when SPI_MODE=1). Integrated pull-down resistor.
DRV_ENN_ CFG6	29	DI	Enable input or configuration / Enable input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.
AIN_IREF	30	AI	Analog reference voltage for current scaling (optional mode) or reference current for use of internal sense resistors
GNDA	32		Analog GND. Tie to GND plane.
5VOUT	33		Output of internal 5V regulator. Attach 2.2µF or larger ceramic capacitor to GNDA near to pin for best performance. May be used to supply VCC of chip.
VCC	34		5V supply input for digital circuitry within chip and charge pump. Attach 470nF capacitor to GND (GND plane). May 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	35		Charge pump capacitor output.

Pin	Number	Type	Function
CPI	37		Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.
VCP	38		Charge pump voltage. Tie to VS using 100nF capacitor.
VSA	39		Analog supply voltage for 5V regulator. Normally tied to VS. Provide a 100nF filtering capacitor.
OA2	42		Motor coil A output 2
BRA	44		Sense resistor connection for coil A. Place sense resistor to GND near pin. An additional 100nF capacitor to GND (GND plane) is recommended for best performance.
OA1	46		Motor coil A output 1
Exposed die pad	-		Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for digital circuitry.

3 Sample Circuits

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

3.1 Standard Application Circuit

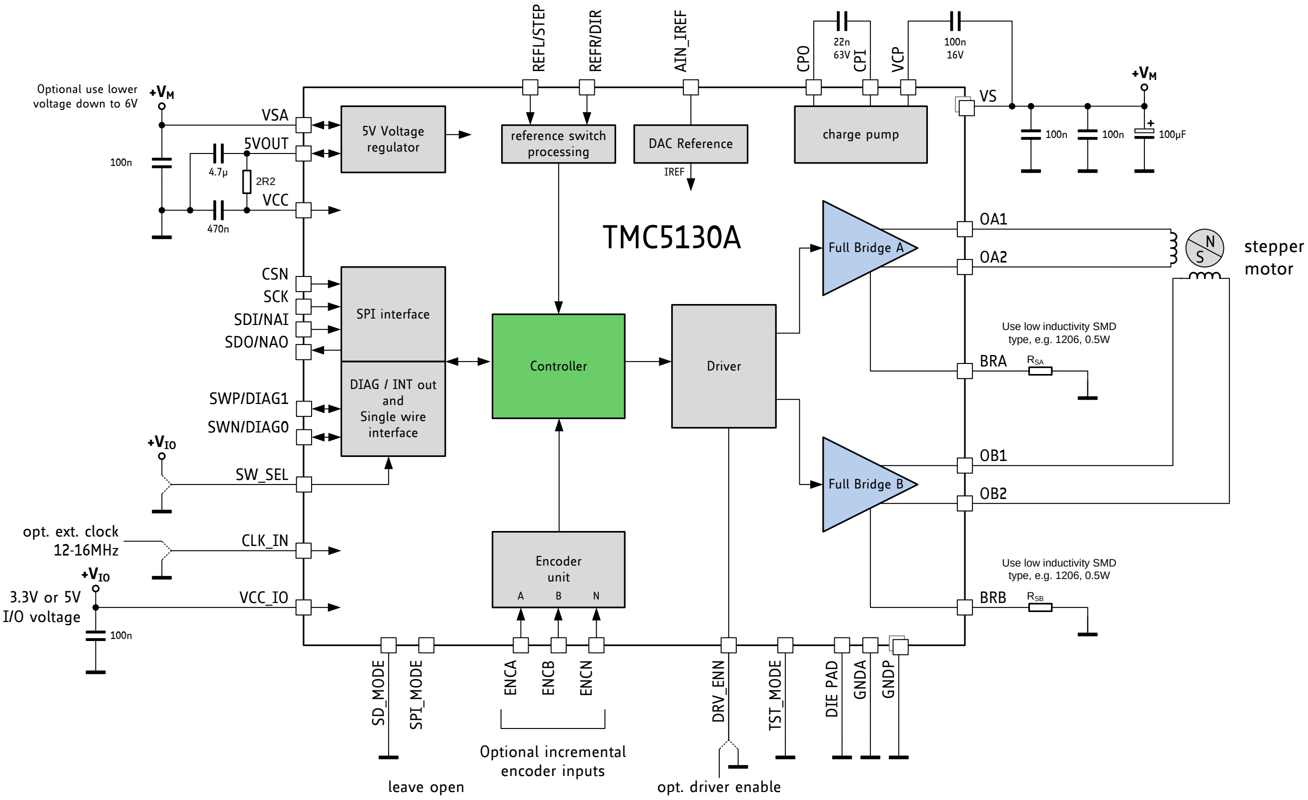


Figure 3.1 Standard application circuit

The standard application circuit uses a minimum set of additional components in order to operate the motor. Use low ESR capacitors for filtering the power supply which are capable to cope with the current ripple. The current ripple often depends on the power supply and cable length. The VCC_IO voltage can be supplied from 5VOUT, or from an external source, e.g. a low drop 3.3V regulator. In order to minimize linear voltage regulator power dissipation of the internal 5V voltage regulator in applications where VM is high, a different (lower) supply voltage can be used for VSA, if available. For example, many applications provide a 12V supply in addition to a higher supply voltage, like 24V or 36V. Using the 12V supply for VSA will reduce the power dissipation of the internal 5V regulator to about 37% resp. 23% of the dissipation caused by supply with the full motor voltage.

Basic layout hints
Place sense resistors and all filter capacitors as close as possible to the related IC pins. Use a solid common GND for all GND 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 exceed VS by more than one diode drop upon power up or power down.

3.2 Reduced Number of Components

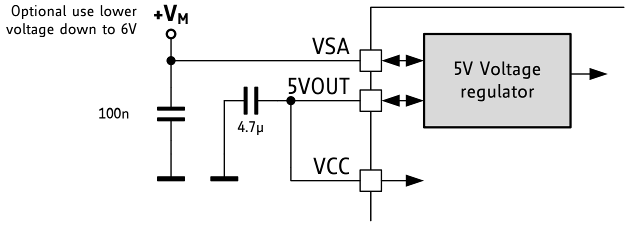


Figure 3.2 Reduced number of filtering components

The standard application circuit uses RC filtering to de-couple the output of the internal linear regulator from high frequency ripple caused by digital circuitry supplied by the VCC input. For cost sensitive applications, the RC-Filtering on VCC can be eliminated. This leads to more noise on 5VOUT caused by operation of the charge pump and the internal digital circuitry. There is a slight impact on microstep vibration and chopper noise performance.

3.3 Internal RDSon Sensing

For cost critical or space limited applications, it may be desired to eliminate the sense resistors. Further, this slightly reduces power dissipation, because the effective resistance of the driver bridge is reduced. In this application, a reference current set by a tiny external resistor programs the output current. For calculation of the reference resistor, refer chapter 11.

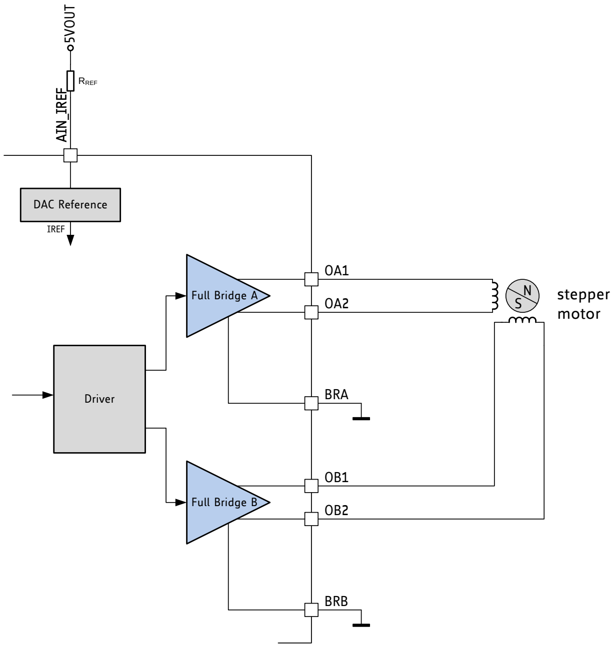


Figure 3.3 RDSon based sensing eliminates high current sense resistors

3.4 External 5V Power Supply

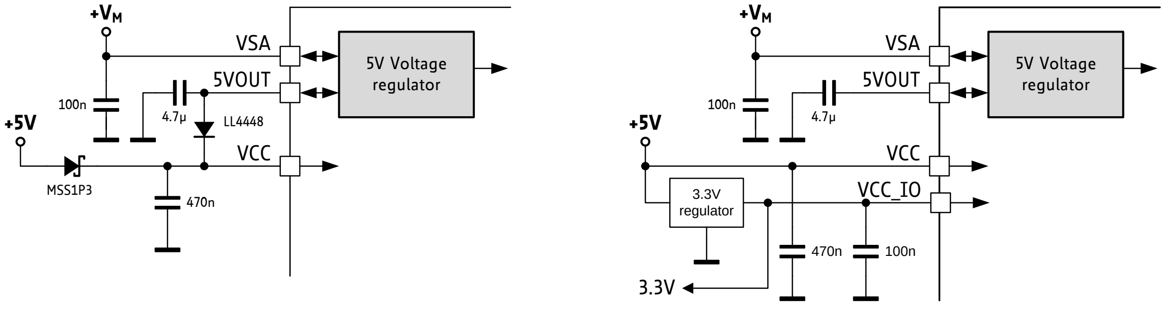
When an external 5V power supply is available, the power dissipation caused by the internal linear regulator can be eliminated. This especially is beneficial in high voltage applications, and when thermal conditions are critical. There are two options for using this external 5V source: either the external 5V source is used to support the digital supply of the driver by supplying the VCC pin or the complete internal voltage regulator becomes bridged and is replaced by the external supply voltage.

3.4.1 Support for the VCC Supply

This scheme uses an external supply for all digital circuitry within the driver (Figure 3.4). As the digital circuitry makes up for most of the power dissipation, this way the internal 5V regulator sees only low remaining load. The precisely regulated voltage of the internal regulator is still used as the reference for the motor current regulation as well as for supplying internal analog circuitry.

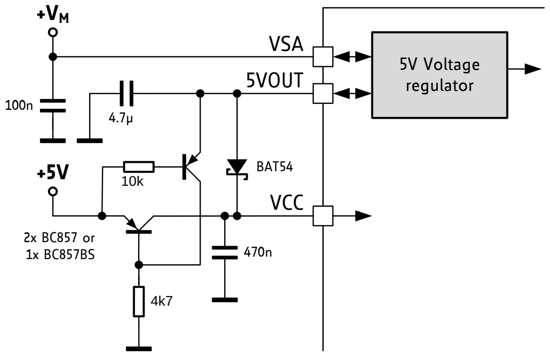
When cutting pin VCC from 5VOUT, make sure that the VCC supply comes up before or synchronously with the 5VOUT supply, because otherwise the power-up reset event may be missed by the internal logic. A simple schematic uses two diodes forming an OR of the internal and the external power supplies to VCC to ensure this. In order to prevent the chip from drawing part of the power from its internal regulator, a low drop 1A Schottky diode is used for the external 5V supply path, while a silicon diode is used for the 5VOUT path. An enhanced solution uses a dual PNP transistor as an active switch. It minimizes voltage drop and thus gives best performance.

Using a 3.3V VCC_IO additional switching can be eliminated given that a safe reset condition is ensured by the 3.3V VCC_IO coming up synchronously with or delayed to VCC. This is true, when a linear regulator is used to generate a 3.3V VCC_IO from the external 5V VCC source. This 3.3V regulator will cause a certain voltage drop. A voltage drop in the regulator of 0.9V or more (e.g. LD1117-3.3) ensures that the 5V supply already has reached a lower limit of more than about 3.0V once the reset conditions ends. The reset condition ends earliest, when VCC_IO exceeds the undervoltage limit of minimum 2.1V. Make sure that the power-down sequence also is safe. Undefined states can result when VCC drops well below 4V without safely triggering a reset condition. Triggering a reset upon power-down can be ensured when VSA goes down synchronously with or before VCC.



VCC supplied from external 5V. 5V or 3.3V IO voltage.

VCC supplied from external 5V. 3.3V IO voltage generated from same source.



VCC supplied from external 5V using active switch. 5V or 3.3V IO voltage.

Figure 3.4 Using an external 5V supply for digital circuitry of driver (different options)

3.4.2 Internal Regulator Bridged

In case a clean external 5V supply is available, it can be used for complete supply of analog and digital part (Figure 3.5). The circuit will benefit from a well regulated supply, e.g. when using a +/-1% regulator. A precise supply guarantees increased motor current precision, because the voltage at 5VOUT directly is the reference voltage for all internal units of the driver, especially for motor current control. For best performance, the power supply should have low ripple to give a precise and stable supply at 5VOUT pin with remaining ripple well below 5mV. Some switching regulators have a higher remaining ripple, or different loads on the supply may cause lower frequency ripple. In this case, increase capacity attached to 5VOUT. In case the external supply voltage has poor stability or low frequency ripple, this would affect the precision of the motor current regulation as well as add chopper noise.

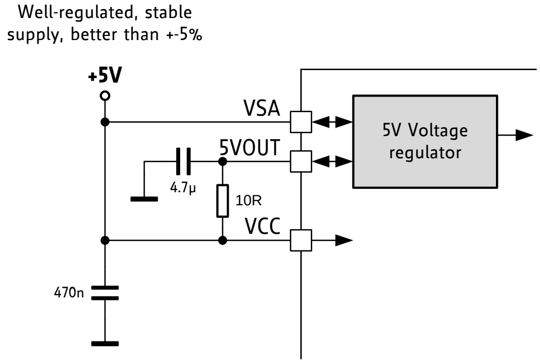


Figure 3.5 Using an external 5V supply to bypass internal regulator

3.5 Pre-Regulator for Reduced Power Dissipation

When operating at supply voltages up to 46V for VS and VSA, the internal linear regulator will contribute with up to 1W to the power dissipation of the driver. This will reduce the capability of the chip to continuously drive high motor current, especially at high environment temperatures. When no external power supply in the range 5V to 24V is available, an external pre-regulator can be built with a few inexpensive components in order to dissipate most of the voltage drop in external components. Figure 3.6 shows different examples. In case a well-defined supply voltage is available, a single 1W or higher power zener diode also does the job.

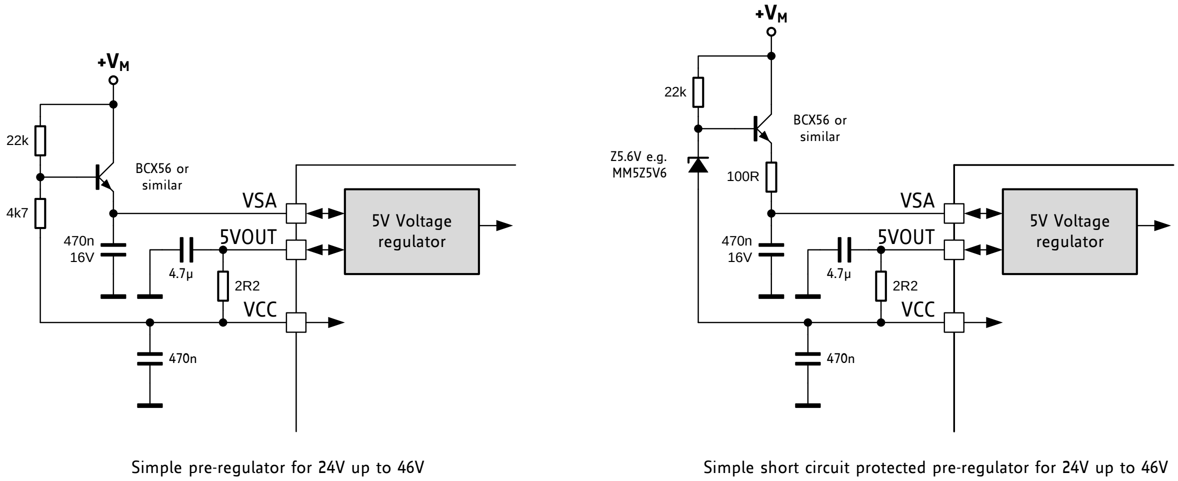


Figure 3.6 Examples for simple pre-regulators

3.6 5V Only Supply

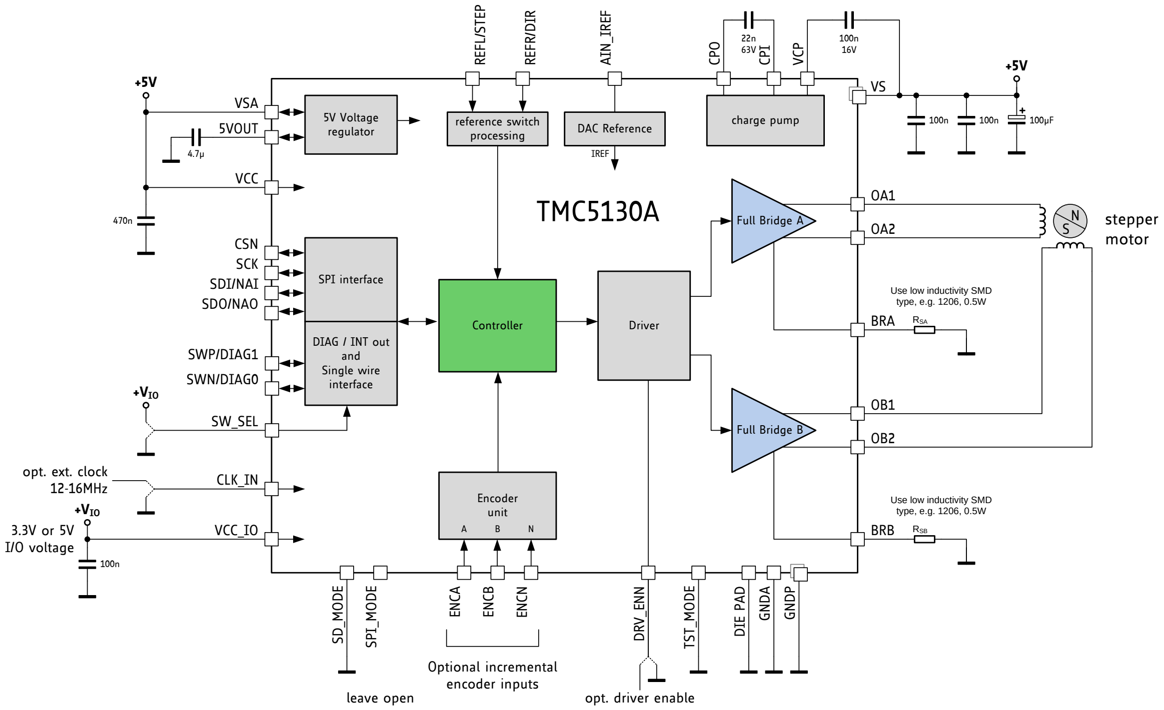


Figure 3.7 5V only operation

While the standard application circuit is limited to roughly 5.5V lower supply voltage, a 5V only application lets the IC run from a normal 5V +/-10% supply. In this application, linear regulator drop must be minimized. Therefore, the major 5V load is removed by supplying VCC directly from the external supply. In order to keep supply ripple away from the analog voltage reference, 5VOUT should have an own filtering capacity and the 5VOUT pin does not become bridged to the 5V supply.

3.7 High Motor Current

When operating at a high motor current, the driver power dissipation due to MOSFET switch on-resistance significantly heats up the driver. This power dissipation will heat up the PCB cooling infrastructure also, if operated at an increased duty cycle. This in turn leads to a further increase of driver temperature. An increase of temperature by about 100°C increases MOSFET resistance by roughly 50%. This is a typical behavior of MOSFET switches. Therefore, under high duty cycle, high load conditions, thermal characteristics have to be carefully taken into account, especially when increased environment temperatures are to be supported. Refer the thermal characteristics and the layout hints for more information. As a thumb rule, thermal properties of the PCB design become critical for the TQFP-48 at or above 1.2A RMS motor current for increased periods of time. Keep in mind that resistive power dissipation raises with the square of the motor current. On the other hand, this means that a small reduction of motor current significantly saves heat dissipation and energy.

An effect which might be perceived at medium motor velocities and motor sine wave peak currents above roughly 1.2A peak is an increasing negative impact of increased internal diode conduction on the duration of the fast decay cycle of the spreadCycle chopper. This is, because the current measurement does not see the full coil current during this phase of the sine wave, because an increasing part of the current flows directly from the power MOSFETs' drain to GND and does not flow through the sense resistor. This in turn under some conditions may lead to a slight sine distortion of the current wave when using spreadCycle. This effect with most motors does not negatively influence the smoothness of operation, as it does not impact the critical current zero transition. It does not occur with stealthChop and with classic chopper.

3.7.1 Reduce Linear Regulator Power Dissipation

When operating at high supply voltages, as a first step the power dissipation of the integrated 5V linear regulator can be reduced, e.g. by using an external 5V source for supply. This will reduce overall heating. It is advised to reduce motor stand still current in order to decrease overall power dissipation. If applicable, also use coolStep. A decreased clock frequency will reduce power dissipation of the internal logic. Further a decreased chopper frequency also can reduce power dissipation.

3.7.2 Operation near to / above 2A Peak Current

The driver can deliver up to 2.5A motor peak current. Considering thermal characteristics, this only is possible in duty cycle limited operation. When a peak current up to 2.5A is to be driven, the driver chip temperature is to be kept at a maximum of 105°C. Linearly derate the design peak temperature from 125°C to 105°C in the range 2A to 2.5A output current (see Figure 3.8). Exceeding this may lead to triggering the short circuit detection.

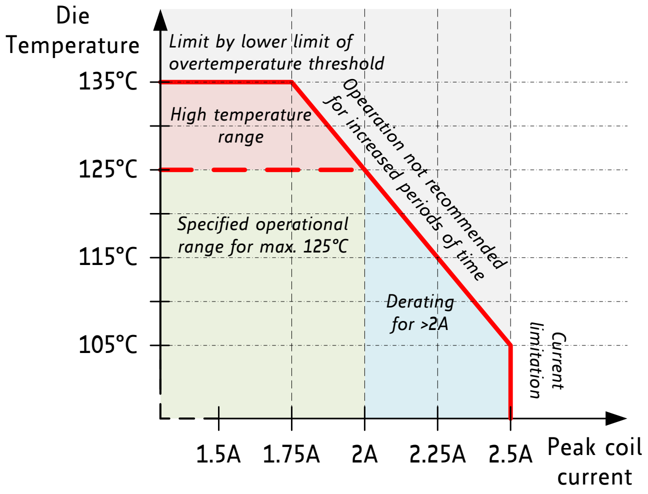


Figure 3.8 Derating of maximum sine wave peak current at increased die temperature

3.7.3 Reduction of Resistive Losses by Adding Schottky Diodes

Schottky Diodes can be added to the circuit to reduce driver power dissipation when driving high motor currents (see Figure 3.9). The Schottky diodes have a conduction voltage of about 0.5V and will take over more than half of the motor current during the negative half wave of each output in slow decay and fast decay phases, thus leading to a cooler motor driver. This effect starts from a few percent at 1.2A and increases with higher motor current rating up to roughly 20%. As a 30V Schottky diode has a lower forward voltage than a 50V or 60V diode, it makes sense to use a 30V diode when the supply voltage is below 30V. The diodes will have less effect when working with stealthChop due to lower times of diode conduction in the chopper cycle. At current levels below 1.2A coil current, the effect of the diodes is negligible.

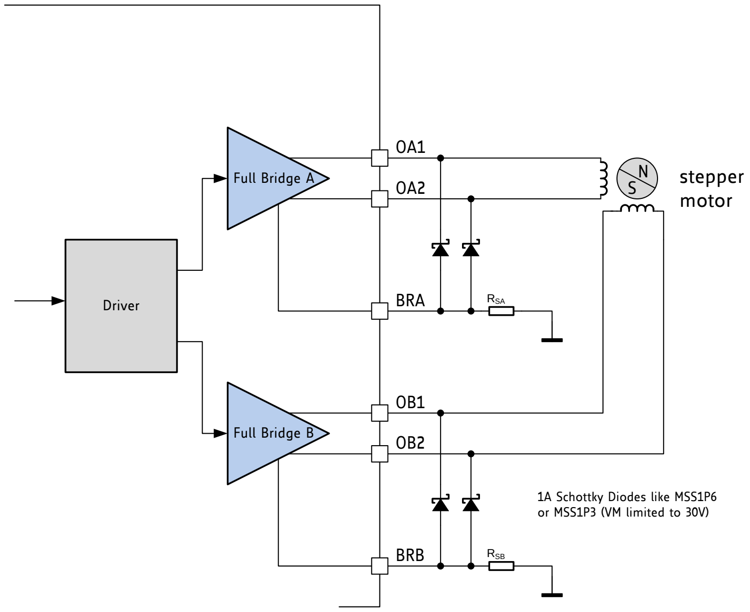
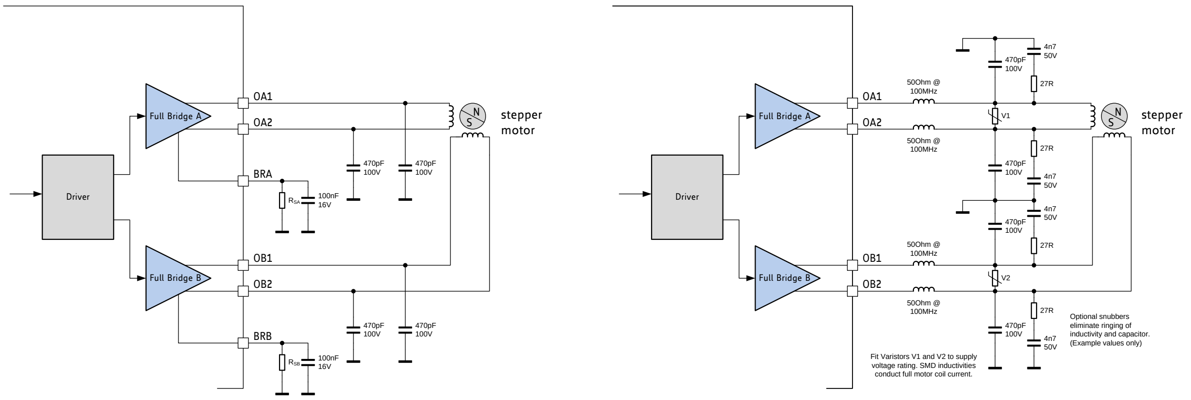


Figure 3.9 Schottky diodes reduce power dissipation at high peak currents up to 2A (2.5A)

3.8 Driver Protection and EME Circuitry

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. 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 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 also dampen high frequency noise injected from digital parts of the circuit and thus reduce electromagnetic emission. A more elaborate scheme uses LC filters to de-couple the driver outputs from the motor connector. Varistors eliminate coil overvoltage caused by live plugging. As LC filters tend to oscillate, additional snubber elements have been added. The drawback of this scheme is that it increases power dissipation significantly, especially at high supply voltages. A dampening resistor in parallel to the ferrite inductivity would be an option to the snubbers.



Fit Varistors V1 and V2 to supply voltage rating. SMD inductivities conduct full motor coil current.

Optional snubbers eliminate ringing of inductivity and capacitor. (Example values only)

Figure 3.10 Simple ESD enhancement and more elaborate motor output protection

4 SPI Interface

4.1 SPI Datagram Structure

The TMC5130A uses 40 bit SPI™ (Serial Peripheral Interface, SPI is Trademark of Motorola) datagrams for communication with a microcontroller. Microcontrollers which are equipped with hardware SPI are typically able to communicate using integer multiples of 8 bit. The NCS line of the device must be handled in a way, that it stays active (low) for the complete duration of the datagram transmission.

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

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

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

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

<table>
<tr><th colspan="40">SPI DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="12">MSB (transmitted first)</td><td colspan="16">40 bit</td><td colspan="12">LSB (transmitted last)</td></tr>
<tr><td colspan="20">39 ...</td><td colspan="20">... 0</td></tr>
<tr><td colspan="8">**→ 8 bit address**
← 8 bit SPI status</td><td colspan="32">**← → 32 bit data**</td></tr>
<tr><td colspan="8">39 ... 32</td><td colspan="32">31 ... 0</td></tr>
<tr><td colspan="8">→ to TMC5130A
RW + 7 bit address
← from TMC5130A
8 bit SPI status</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td></tr>
<tr><td colspan="8">39 / 38 ... 32</td><td colspan="8">31 ... 24</td><td colspan="8">23 ... 16</td><td colspan="8">15 ... 8</td><td colspan="8">7 ... 0</td></tr>
<tr><td>W</td><td colspan="7">38...32</td><td colspan="4">31...28</td><td colspan="4">27...24</td><td colspan="4">23...20</td><td colspan="4">19...16</td><td colspan="4">15...12</td><td colspan="4">11...8</td><td colspan="4">7...4</td><td colspan="4">3...0</td></tr>
<tr><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

4.1.1 Selection of Write / Read (WRITE_notREAD)

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

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

Whenever data is read from or written to the TMC5130A, 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 TMC5130A	data received from TMC5130A
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 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 TMC5130A 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 TMC5130A.

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.

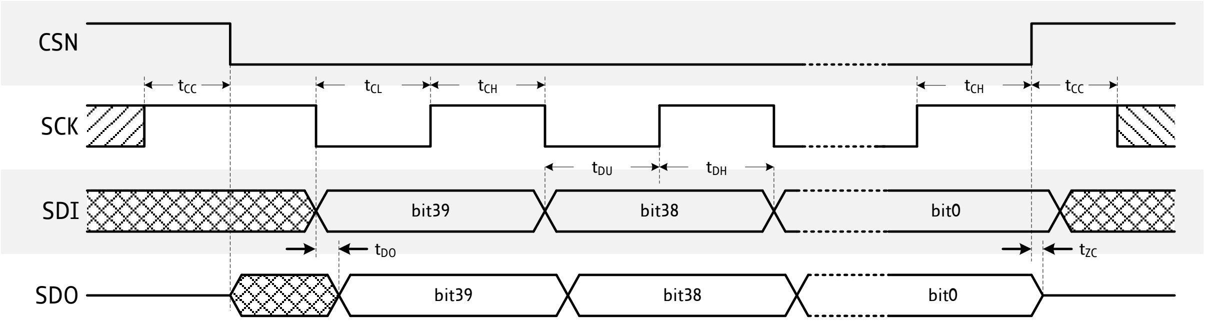


Figure 4.1 SPI timing

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 TMC5130A-TA 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 TMC5130A 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 SWIOP) and ends with a stop bit (logic 1, high level on SWIOP). 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 TMC5130A 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 TMC5130A 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.

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.

ERRATA IN READ ACCESS

A known bug in the UART interface implementation affects read access to registers that change during the access. While the SPI interface takes a snapshot of the read register before transmission, the UART interface transfers the register directly MSB to LSB without taking a snapshot. This may lead to inconsistent data when reading out a register that changes during the transmission. Further, the CRC sent from the driver may be incorrect in this case (but must not), which will lead to the master repeating the read access. As a workaround, it is advised not to read out quickly changing registers like *XACTUAL*, *MSCNT* or *X_ENC* during a motion, but instead first stop the motor or check the *position_reached* flag to become active, and read out these values afterwards. If possible, use *X_LATCH* and *ENC_LATCH* for a safe readout during motion (e.g. for homing). As the encoder cannot be guaranteed to stand still during motor stop, only a dual read access and check for identical result ensures correct *X_ENC* read data. Use the *vzero* and *velocity_reached* flag rather than reading *VACTUAL*.

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 TMC5130A responds only to correctly transmitted datagrams containing its own slave address. It increases its datagram counter for each correctly received write access datagram.

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

Hint:
The CRC can be calculated within a CPU using a bit-wise cyclic XOR calculation of incoming and outgoing bits accumulated to an 8 bit CRC register. You find the algorithm in the TMC5130A-EVAL evaluation board firmware.

```
CRC = (CRC << 1) OR (CRC.7 XOR CRC.1 XOR CRC.0 XOR [new incoming bit])
-- CRC.n is meant to extract bit n from the 8 bit CRC register
```

For a parallel 8 bit calculation of CRC in your CPU, you can use a look-up table. Additional algorithms can be found in literature.

5.3 UART Signals

The UART interface on the TMC5130A-TA comprises four signals:

TMC5130A UART INTERFACE SIGNALS	
SWIOP	Non-inverted data input and output
SWION	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.
NAI	Address increment pin for chained sequential addressing scheme
NAO	Next address output pin for chained sequential addressing scheme (reset default= high)

In UART mode (SW_SEL high) the slave checks the single wire SWIOP and SWION 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 SWIOP and SWION and sends its response using the same baud rate.

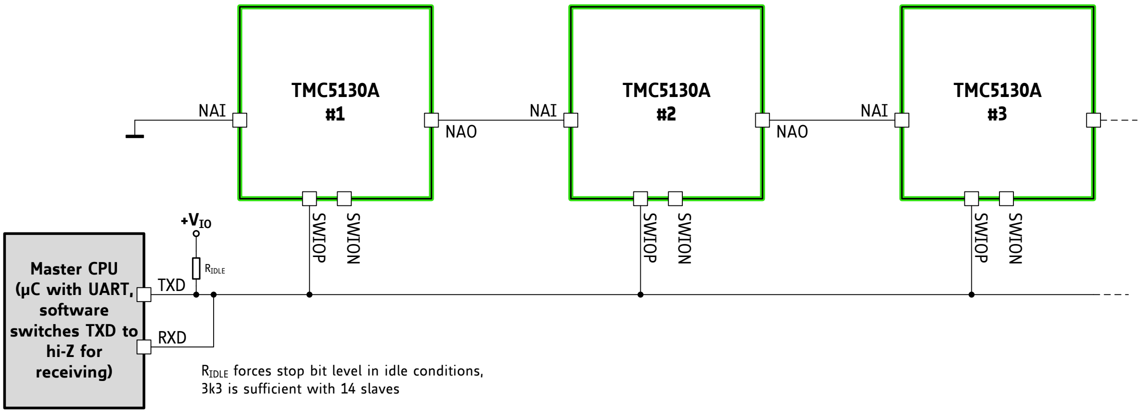
5.4 Addressing Multiple Slaves

ADDRESSING ONE OR TWO SLAVES

If only one or two TMC5130A 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.



R_IDLE forces stop bit level in idle conditions,
3k3 is sufficient with 14 slaves

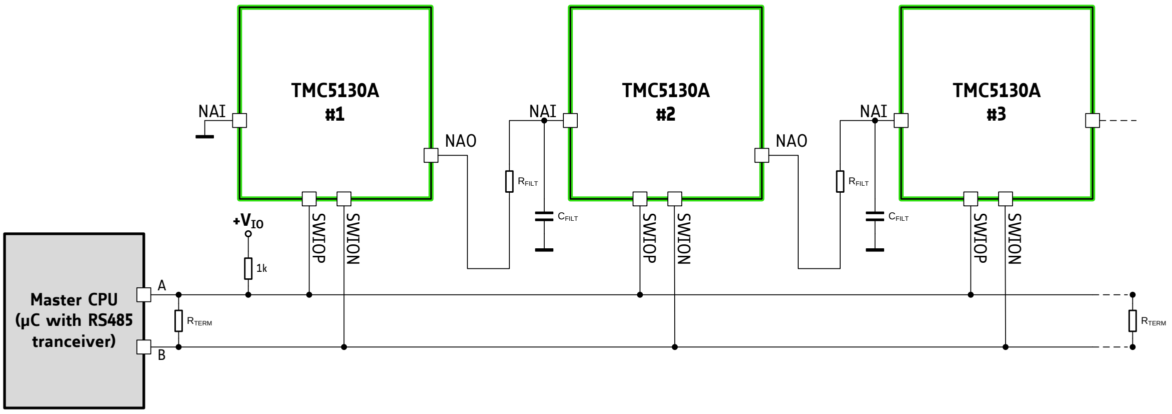
EXAMPLE FOR ADDRESSING UP TO 255 TMC5130A

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 TMC5130A via single wire interface using chaining

PROCEED AS FOLLOWS:

- Tie the NAI pin of your first TMC5130A to GND.
- Interconnect NAO output of the first TMC5130A 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 TMC5130A

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 multiple TMC5130A via the 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 TMC5130A 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	*I_scale_analog* 0: Normal operation, use internal reference voltage 1: Use voltage supplied to AIN as current reference
				1	*internal_Rsense* 0: Normal operation 1: Internal sense resistors. Use current supplied into AIN as reference for internal sense resistor
				2	*en_pwm_mode* 1: stealthChop voltage PWM mode enabled (depending on velocity thresholds). Switch from off to on state while in stand still, only.
				3	*enc_commutation* 1: Enable commutation by full step encoder (DCIN_CFG5 = ENC_A, DCEN_CFG4 = ENC_B)
				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 voltage PWM is not available in this mode. The automatic voltage PWM 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 *Attention: Not for user, set to 0 for normal operation!*
R+C	0x01	3	*GSTAT*	**Bit**	***GSTAT* – Global status 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 reset when all error conditions are cleared.
				2	*uv_cp* 1: Indicates an undervoltage on the charge pump. The driver is disabled in this case.
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, 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_CFG6
				5	ENC_N_DCO
				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*: 0x11=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. *Attention: 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.	

6.2 Velocity Dependent Driver Feature Control Register Set

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

R/W	Addr	n	Register	Bit	Description / bit names
W	0x10	5 + 5 + 4	*IHOLD_IRUN*	**Bit**	***IHOLD_IRUN* – Driver current control**
				4..0	*IHOLD* Standstill current (0=1/32…31=32/32) In combination with stealthChop mode, setting *IHOLD*=0 allows to choose freewheeling or coil short circuit for motor stand still.
				12..8	*IRUN* Motor run current (0=1/32…31=32/32) *Hint:* Choose sense resistors in a way, that normal *IRUN* is 16 to 31 for best microstep performance.
				19..16	*IHOLDDELAY* Controls the number of clock cycles for motor power down after a motion as soon as standstill is detected (*stst*=1) and *TPOWERDOWN* has expired. The smooth transition avoids a motor jerk upon power down. 0: instant power down 1..15: Delay per current reduction step in multiple of 2^18 clocks
W	0x11	8	*TPOWER DOWN*		*TPOWERDOWN* sets the delay time after stand still (*stst*) of the motor to motor current power down. Time range is about 0 to 4 seconds. *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.
W	0x13	20	*TPWMTHRS*		This is the upper velocity for stealthChop voltage PWM mode. *TSTEP* ≥ *TPWMTHRS* - stealthChop PWM mode is enabled, if configured - dcStep is disabled

VELOCITY DEPENDENT DRIVER FEATURE CONTROL REGISTER SET (0X10…0X1F)				
R/W	**Addr**	**n**	***Register***	**Description / *bit names***
W	0x14	20	*TCOOLTHRS*	This is the lower threshold velocity for switching on smart energy coolStep and stallGuard feature. (unsigned) Set this parameter to disable coolStep at low speeds, where it cannot work reliably. The 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 and stall output signal 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 target velocity (unsigned) 0: Disables *A1* and *D1* phase, use *AMAX*, *VMAX* 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*	Second acceleration phase target velocity *VMAX* > *V1*, *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) *Attention: Set VSTOP ≥ VSTART!* *Attention: Do not set 0 in positioning mode, minimum 10 recommend!*	1…(2^18)-1 [µsteps / t]
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 sensor-less 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	11	***SW_MODE***	Switch mode configuration *See separate table!*
R+C	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 ta² 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.*
10	*sg_stop*	1: Enable stop by stallGuard2 (also available in dcStep mode). Disable to release motor after stop event. *Attention: 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
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+C	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 (Flag is cleared upon reading)
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+C	7	*event_pos_reached*	1: Signals, that the target position has been reached (*position_reached* becoming active). (Flag and interrupt condition are cleared upon reading) This bit is ORed to the *interrupt output* signal.
R+C	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. (Flag and interrupt condition are cleared upon reading) 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+C	3	*status_latch_r*	1: Latch right ready (enable position latching using *SWITCH_MODE* settings *latch_r_active* or *latch_r_inactive*) (Flag is cleared upon reading)
	2	*status_latch_l*	1: Latch left ready (enable position latching using *SWITCH_MODE* settings *latch_l_active* or *latch_l_inactive*) (Flag is cleared upon reading)
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 / *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+C	0x3B	1	*ENC_STATUS*	bit 0: *n_event* 1: Encoder N event detected. Status bit is cleared on read: Read (R) + clear (C) This bit is ORed to the *interrupt output* signal.	
R	0x3C	32	*ENC_LATCH*	Encoder position *X_ENC* latched on N event	

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 / *bit names*	Range [Unit]
W	0x60	32	*MSLUT[0]* microstep table entries 0…31	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224*	32x 0 or 1 *reset default= sine wave table*
W	0x61 … 0x67	7 x 32	*MSLUT[1…7]* microstep table entries 32…255		7x 32x 0 or 1 *reset default= sine wave table*
W	0x68	32	***MSLUTSEL***	This register defines four segments within each quarter *MSLUT* wave. Four 2 bit entries determine the meaning of a 0 and a 1 bit in the corresponding segment of *MSLUT*. *See separate table!*	*0<X1<X2<X3* *reset default= sine wave table*
W	0x69	8 + 8	*MSLUTSTART*	bit 7… 0: *START_SIN* bit 23… 16: *START_SIN90* *START_SIN* gives the absolute current at microstep table entry 0. *START_SIN90* gives the absolute current for microstep table entry at positions 256. Start values are transferred to the microstep registers *CUR_A* and *CUR_B*, whenever the reference position *MSCNT*=0 is passed.	*START_SIN* *reset default =0* *START_SIN90* *reset default =247*
R	0x6A	10	*MSCNT*	Microstep counter. Indicates actual position in the microstep table for *CUR_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!*	
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 *Attention: 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= 0x00050480*
R	0x71	8	*PWM_SCALE*	Actual PWM amplitude scaler (255=max. Voltage) In voltage mode PWM, this value allows to detect a motor stall.	0…255
W	0x72	2	*ENCM_CTRL*	Encoder mode configuration Bit 0: *inv*: Invert encoder inputs Bit 1: *maxspeed*: Ignore Step input. If set, the hold current *IHOLD* determines the motor current, unless a step source is activated. The direction in this mode is determined by the *shaft* bit in *GCONF* or by the *inv* bit.	
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	-	-	Reserved, set to 0
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	*sync3*	*SYNC* PWM synchronization clock	This register allows synchronization of the chopper for both phases of a two phase motor in order to avoid the occurrence of a beat, especially at low motor velocities. It is automatically switched off above *VHIGH*. %0000: Chopper sync function chopSync off %0001 … %1111: Synchronization with $f_{SYNC} = f_{CLK}/(sync*64)$ *Hint:* Set *TOFF* to a low value, so that the chopper cycle is ended, before the next sync clock pulse occurs. Set for the double desired chopper frequency for *chm*=0, for the desired base chopper frequency for *chm*=1.
22	*sync2*		
21	*sync1*		
20	*sync0*		
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	*vsense*	sense resistor voltage based current scaling	0: Low sensitivity, high sense resistor voltage 1: High sensitivity, low sense resistor voltage
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.

0x6C: *CHOPCONF* – CHOPPER CONFIGURATION

Bit	Name	Function	Comment	
13	*rndtf*	random *TOFF* time	0	Chopper off time is fixed as set by *TOFF*
			1	Random mode, *TOFF* is random modulated by dN_{CLK}= -12 … +3 clocks.
12	*disfdcc*	fast decay mode	*chm*=1: *disfdcc*=1 disables current comparator usage for termination of the fast decay cycle	
11	*fd3*	*TFD [3]*	*chm*=1: MSB of fast decay time setting *TFD*	
10	*hend3*	*HEND* hysteresis low value *OFFSET* sine wave offset	*chm*=0	%0000 … %1111: Hysteresis is -3, -2, -1, 0, 1, …, 12 (1/512 of this setting adds to current setting) This is the hysteresis value which becomes used for the hysteresis chopper.
9	*hend2*			
8	*hend1*			
7	*hend0*		*chm*=1	%0000 … %1111: Offset is -3, -2, -1, 0, 1, …, 12 This is the sine wave offset and 1/512 of the value becomes added to the absolute value of each sine wave entry.
6	*hstrt2*	*HSTRT* hysteresis start value added to *HEND*	*chm*=0	%000 … %111: Add 1, 2, …, 8 to hysteresis low value *HEND* (1/512 of this setting adds to current setting) *Attention: Effective HEND+HSTRT ≤ 16.* *Hint:* Hysteresis decrement is done each 16 clocks
5	*hstrt1*			
4	*hstrt0*	*TFD [2..0]* fast decay time setting	*chm*=1	Fast decay time setting (MSB: *fd3*): %0000 … %1111: Fast decay time setting *TFD* with N_{CLK}= 32**HSTRT* (%0000: slow decay only)
3	*toff3*	*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	
2	*toff2*			
1	*toff1*			
0	*toff0*			

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	
…	-	reserved	set to 0	
21	*freewheel1*	Allows different standstill modes	Stand still option when motor current setting is zero (*I_HOLD*=0). %00: Normal operation %01: Freewheeling %10: Coil shorted using LS drivers %11: Coil shorted using HS drivers	
20	*freewheel0*			
19	*pwm_ symmetric*	Force symmetric PWM	0	The PWM value may change within each PWM cycle (standard mode)
			1	A symmetric PWM cycle is enforced
18	*pwm_ autoscale*	PWM automatic amplitude scaling	0	User defined PWM amplitude. The current settings have no influence.
			1	Enable automatic current control *Attention: When using a user defined sine wave table, the amplitude of this sine wave table should not be less than 244. Best results are obtained with 247 to 252 as peak values.*
17	*pwm_freq1*	PWM frequency selection	%00: f_{PWM}=1/1024 f_{CLK} %01: f_{PWM}=1/683 f_{CLK} %10: f_{PWM}=1/512 f_{CLK} %11: f_{PWM}=1/410 f_{CLK}	
16	*pwm_freq0*			
15	*PWM_ GRAD*	User defined amplitude (gradient) or regulation loop gradient	pwm_ autoscale=0	Velocity dependent gradient for PWM amplitude: *PWM_GRAD* * 256 / *TSTEP* is added to *PWM_AMPL*
14				
13				
12				
11			pwm_ autoscale=1	User defined maximum PWM amplitude change per half wave (1 to 15)
10				
9				
8				
7	*PWM_ AMPL*	User defined amplitude (offset)	pwm_ autoscale=0	User defined PWM amplitude offset (0-255) The resulting amplitude (limited to 0…255) is: *PWM_AMPL* + *PWM_GRAD* * 256 / *TSTEP*
6				
5				
4				
3			pwm_ autoscale=1	User defined maximum PWM amplitude when switching back from current chopper mode to voltage PWM mode (switch over velocity defined by TPWMTHRS). Do not set too low values, as the regulation cannot measure the current when the actual PWM value goes below a setting specific value. Settings above 0x40 recommended.
2				
1				
0				

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	-	reserved	Ignore these bits
13			
12			
11			
10			
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. *SG_RESULT* is ALSO applicable when dcStep is active. stallGuard2 works best with microstep operation. Temperature measurement: In standstill, no stallGuard2 result can be obtained. *SG_RESULT* shows the chopper on-time for motor coil A instead. If the motor is moved to a determined microstep position at a certain current setting, a comparison of the chopper on-time can help to get a rough estimation of motor temperature. 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 low and medium velocities. 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. There are no more configurations required except for the regulation of the PWM voltage to yield the motor target current. Two algorithms are provided, a manual and an automatic mode.

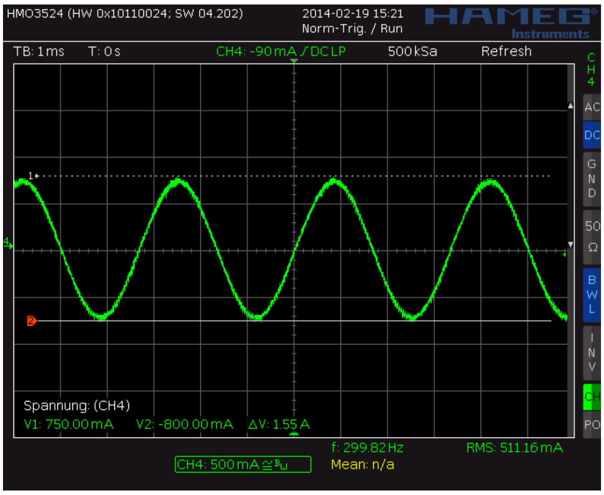


Figure 7.1 Motor coil sine wave current with stealthChop (measured with current probe)

7.1 Two Modes for Current Regulation

In order to match the motor current to a certain level, the voltage mode PWM voltage must be 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 actual level of the supply voltage. For the ease of use, two modes of PWM regulation are provided: An automatic mode using current feedback (*pwm_autoscale* = 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. Since this mode does not measure the actual current, it will not respond to modification of the current setting, like stand still current reduction. Therefore we recommend the automatic mode, unless current regulation is not satisfying in the given operating conditions.

The PWM frequency can be chosen in a range in four steps in order to adapt the frequency divider to the frequency of the clock source. An optimum setting is slightly above the audible frequency range. A slightly higher value might bring a benefit for some applications.

Choice of PWM frequency for stealthChop				
Clock frequency f_{CLK}	**PWM_FREQ=%00** f_{PWM}=1/1024 f_{CLK}	**PWM_FREQ=%01** f_{PWM}=1/683 f_{CLK}	**PWM_FREQ=%10** f_{PWM}=1/512 f_{CLK}	**PWM_FREQ=%11** f_{PWM}=1/410 f_{CLK}
18MHz	17.6kHz	26.3kHz	35.2kHz	
16MHz	15.6kHz	23.4kHz	31.2kHz	39.0kHz
(internal)	~13kHz	~19kHz	~26kHz	~32kHz
12MHz	11.7kHz	17.6kHz	23.4kHz	29.3kHz
10MHz		14.6kHz	19.5kHz	24.4kHz
8MHz		11.7kHz	15.6kHz	19.5kHz

Table 7.1 Choice of PWM frequency – green: recommended

7.2 Automatic Scaling

In stealthChop voltage PWM mode, the autoscaling function (*pwm_autoscale* = 1) regulates the motor current to the desired current setting. The driver measures the motor current during the chopper on time and uses a proportional regulator to regulate the *PWM_SCALE* in order match the motor current to the target current. *PWM_GRAD* 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, the motor velocity or effects resulting from changes of the supply voltage. As the supply voltage level and motor temperature normally change only slowly, a minimum setting of the regulation gradient often is sufficient (*PWM_GRAD*=1). If stealthChop operation is desired for a higher velocity range, variations of the motor back EMF caused by motor acceleration and deceleration may require a quicker regulation. *PWM_GRAD* setting should be optimized for the fastest required acceleration and deceleration ramp (see Figure 7.4). The quality of a given setting can be examined when monitoring *PWM_SCALE* and motor velocity. Just as in the acceleration phase, during a deceleration phase the voltage PWM amplitude must be adapted in order to keep the motor coil current constant. When the upper acceleration and the upper deceleration used in the application are identical, the value determined for the acceleration phase will already be optimum for both.

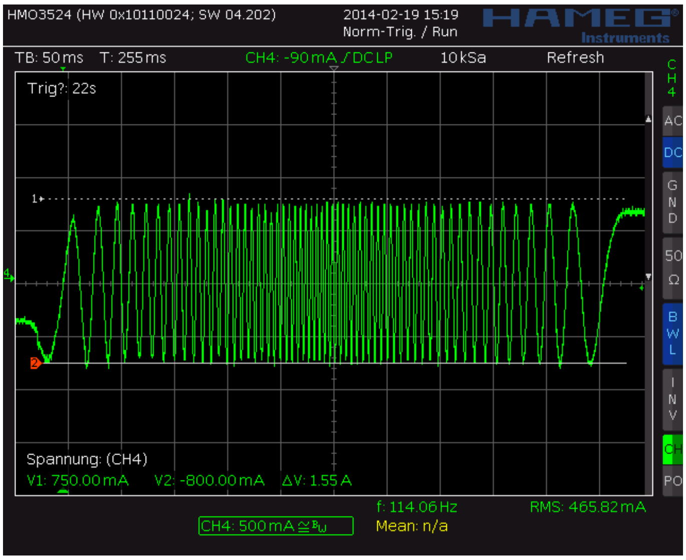


Figure 7.2 Scope shot: good setting for PWM_GRAD

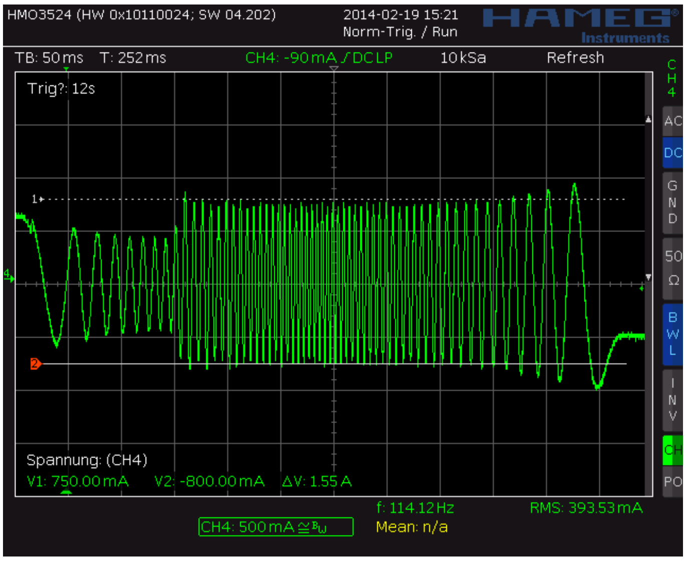
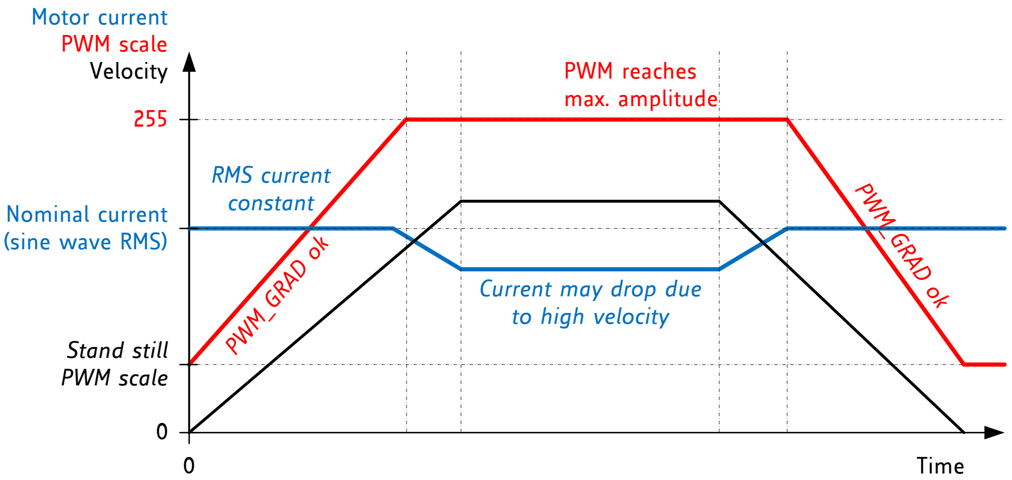
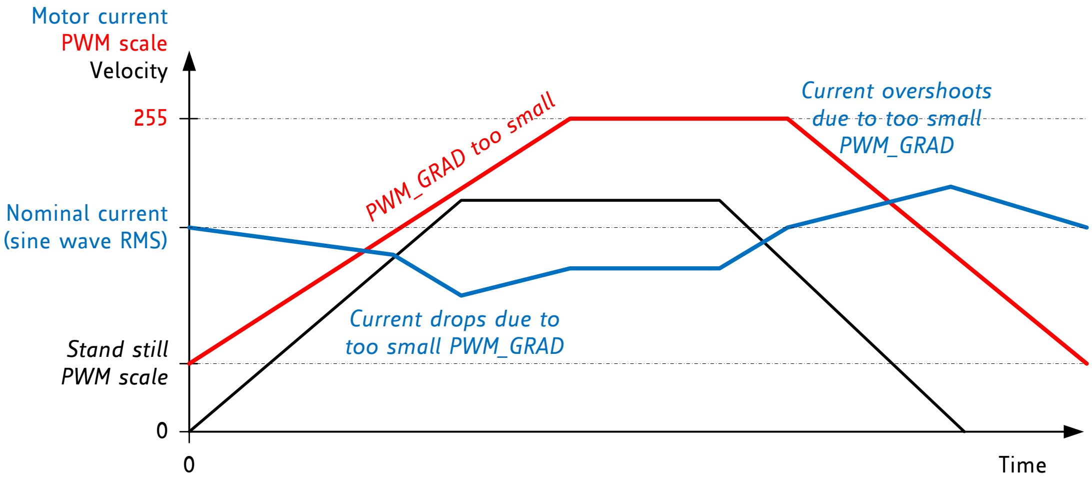


Figure 7.3 Scope shot: too small setting for PWM_GRAD



Setting for PWM_GRAD ok.



Setting for PWM_GRAD slightly too small.

Figure 7.4 Good and too small setting for PWM_GRAD

However, the autoscaling function cannot measure low RMS current settings with high supply voltages and low inductive motors, because chopper on time may become too short for successfully measuring the motor current. Especially at higher PWM frequency settings the lower current limit rises. This happens when the PWM on time required for reaching the necessary coil current falls below the comparator blank time. Try a lower blanking time in order to reduce the lower current limit. Extremely low currents (e.g. for standstill power down) can be realized with the non-automatic current scaling or with the freewheeling option, only. In case the *PWM_SCALE* drops to a too low value, e.g. because the current scale was too low, the regulator may not be able to recover. The regulator will recover once the motor is in standstill.

Be sure to use a symmetrical sense resistor layout and sense resistor traces of identical length and well matching sense resistors for best performance.

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 24.

7.2.1 PWM_AMPL for Using stealthChop and spreadCycle

When combining stealthChop with spreadCycle or constant off time classic PWM, a switching velocity can be chosen using *TPWMTHRS*. With this, stealthChop is only active at low velocities. Often, a very low velocity in the range of 1 to a few 10 RPM fits best. In case a high switching velocity is chosen, special care should be taken for switching back to stealthChop during deceleration, because the phase jerk can produce a short time overcurrent. (Refer to chapter 7.4 for more details about combining stealthChop with other chopper modes.)

To avoid a short time overcurrent and to minimize the jerk, the initial amplitude for switching back to stealthChop at sinking velocity can be determined using the setting *PWM_AMPL*. Tune *PWM_AMPL* to a value which gives a smooth and safe transition back to stealthChop within the application. As a thumb rule, ½ to ¾ of the last *PWM_SCALE* value which was valid after the switching event at rising velocity can be used. For high resistive steppers as well as for low transfer velocities (as set by *TPWMTHRS*), *PWM_AMPL* can be set to 255 as most universal setting.

Note

The autoscaling function only starts up regulation during motor standstill. After enabling stealthChop and setting all parameters, be sure to wait until *PWM_SCALE* has reached a stable state before starting a motion. Failure to do so will result in zero motor current!

In case the automatic scaling regulation is instable at your desired motion velocity, try modifying the chopper frequency divider *PWM_FREQ*. Also adapt the blank time *TBL* and motor current for best result.

7.2.2 Acceleration

In automatic current regulation mode (*pwm_autoscale* = 1), the *PWM_GRAD* setting should be optimized for the fastest required acceleration ramp. Use a current probe and check the motor current during (quick) acceleration. A setting of 1 may result in a too slow regulation, while a setting of 15 responds very quickly to velocity changes, but might produce regulation instabilities in some constellations. A setting of 4 is a good starting value.

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.

7.3 Velocity Based Scaling

Velocity based scaling scales the stealthChop amplitude based on the time between each two steps, i.e. based on *TSTEP*. 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_AMPL can be calculated:

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

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 TMC5130A provides a second velocity dependent factor (*PWM_GRAD*) to compensate for this. The overall effective PWM amplitude (*PWM_SCALE*) in this mode automatically is calculated in dependence of the microstep frequency as:

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

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.

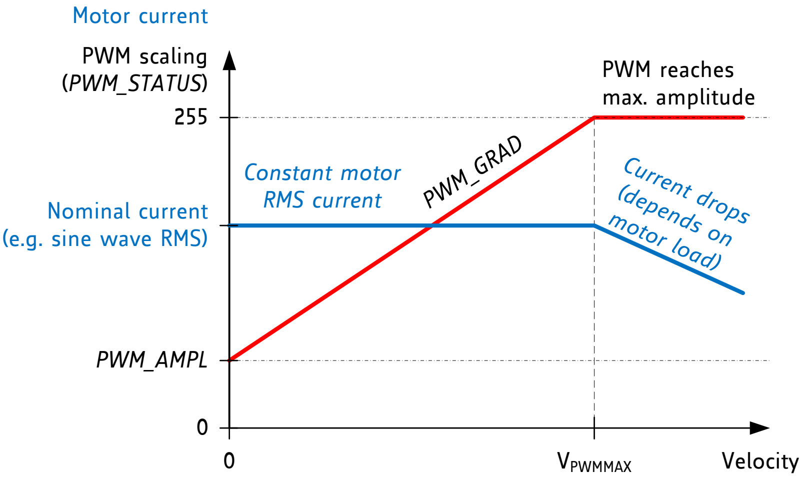


Figure 7.5 Velocity based PWM scaling (pwm_autoscale=0)

Hint
The values for *PWM_AMPL* and *PWM_GRAD* can easily be optimized by tracing the motor current with a current probe on the oscilloscope. It is not even necessary to calculate the formulas if you carefully start with a low setting for both.

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.

7.4 Combining stealthChop with other Chopper Modes

The TMC5130A allows combining stealthChop and different chopper modes based on velocity thresholds. This way, the optimum chopper principle can be chosen for different velocity ranges. 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. 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. At low velocities (e.g. 1 to a few 10 RPM), it can be completely neglected for most motors. Therefore, the *TPWMTHRS* should be set to a low velocity, in order to eliminate any jerk in case an automatic switching between two chopper modes is desired. 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 a different chopper mode 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.

7.5 Flags in stealthChop

7.5.1 Open Load Flags

In stealthChop mode, status information is different from the cycle-by-cycle regulated chopper modes. 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 tiny 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.
- 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 acceleration required a quick action of the current regulator).
  With automatic scaling and *PWM_GRAD* > 1, the current regulation tries to increase the current quickly to reach the target current in the interrupted motor coil. At the same time but a bit slower the current regulation tries to decrease the motor current due to the other motor coil seeing too high current.

Therefore it is recommended to do an on-demand open load test using the spreadCycle or classic chopper prior to operation in stealthChop, and not to switch on stealthChop in case of open load failure. Alternatively, PWM_SCALE can be checked for plausible values.

7.5.2 PWM_SCALE Informs about the Motor State

Information about the motor state is available with automatic scaling by reading out *PWM_SCALE*. 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* value allows seeing the motor load (similar to stallGuard2) and finding out if the target current can be reached. It even gives an idea on the motor temperature (evaluate at a well-known state of operation).

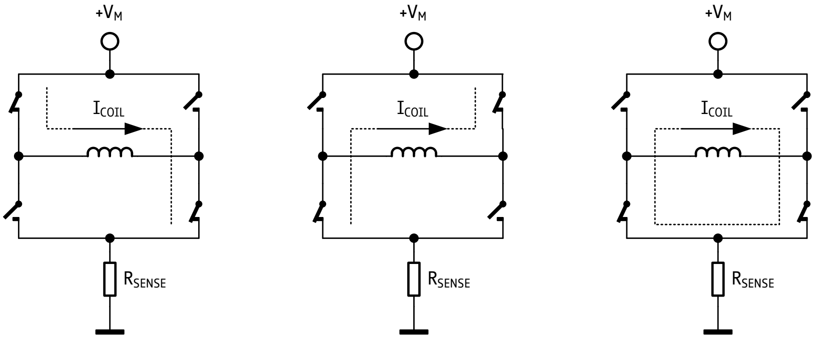
7.6 Freewheeling and Passive Braking

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*. The *PWM_SCALE* regulation becomes frozen once the motor target current is at zero current in order to ensure a quick startup.

Parameter	**Description**	**Setting**	**Comment**
en_pwm_ mode	General enable for use of stealthChop (register *GCONF*)	0	Do not use stealthChop
		1	stealthChop enabled
TPWMTHRS	Specifies the upper velocity for operation in stealthChop voltage PWM mode. Entry the *TSTEP* reading (time between two microsteps) when operating at the desired threshold velocity.	0 … 1048575	stealthChop also is disabled if *TSTEP* falls below *TCOOLTHRS* or *THIGH*
pwm_ autoscale	Enable automatic current scaling using current measurement or use forward controlled velocity based mode.	0	Forward controlled mode
		1	Automatic scaling with current regulator
PWM_FREQ	PWM frequency selection. stealthChop uses a fixed PWM frequency by dividing the system clock frequency using a programmable divider. Use the lowest setting giving good results.	0	f_{PWM}=1/1024 f_{CLK}
		1	f_{PWM}=1/683 f_{CLK}
		2	f_{PWM}=1/512 f_{CLK}
		3	f_{PWM}=1/410 f_{CLK}
PWM_GRAD	User defined PWM amplitude (gradient) for velocity based scaling or regulation loop gradient when *pwm_autoscale*=1.	1 … 15	With *pwm_autoscale*=1
		0 … 255	With *pwm_autoscale*=0
PWM_AMPL	User defined PWM amplitude (offset) for velocity based scaling or amplitude limit for re-entry into stealthChop mode when *pwm_autoscale*=1.	0 … 255	
pwm_ symmetric	Activate to force a symmetric PWM for each cycle. Reduces the number of updates to the PWM cycle. Special use only.	0	Normal operation
		1	A symmetric PWM cycle is enforced
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. Mode 2 will brake more intensely than mode 3, because low side drivers (LS) have lower resistance than high side drivers.	0	Normal operation
		1	Freewheeling
		2	Coil shorted using LS drivers
		3	Coil shorted using HS drivers
PWM_SCALE	Read back of the actual stealthChop voltage PWM scaling as determined by the current regulation. Can be used to detect motor load and stall when *autoscale*=1.	0 … 255 (read only)	The scaling value becomes frozen when operating in a different chopper mode
TOFF	General enable for the motor driver, the actual value does not influence stealthChop	0	Driver off
		1 … 15	Driver enabled
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. A lower setting allows 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}
IRUN *IHOLD*	Run and hold current setting for stealth Chop operation – only used with *pwm_autoscale*=1		See chapter on current setting for details

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.



On Phase:
current flows in
direction of target
current

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

Slow Decay Phase:
current re-circulation

Figure 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. It is advised to check the resulting frequency and to work below 50 kHz.

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

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*:

Assumptions:
Target Chopper frequency: 30kHz
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.

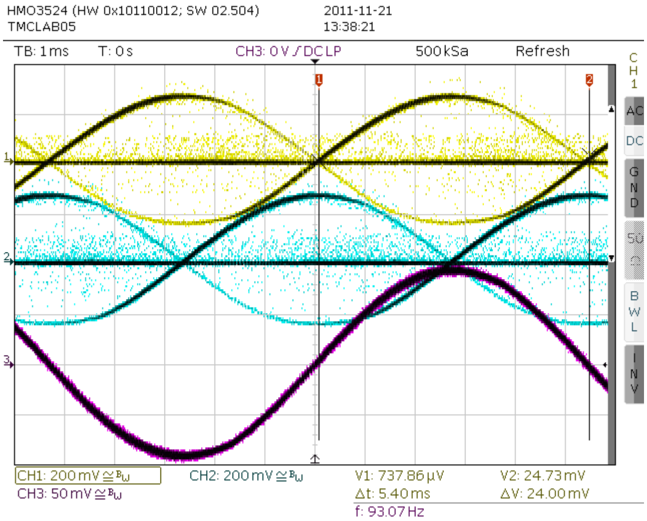


Figure 8.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)

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 24.
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.

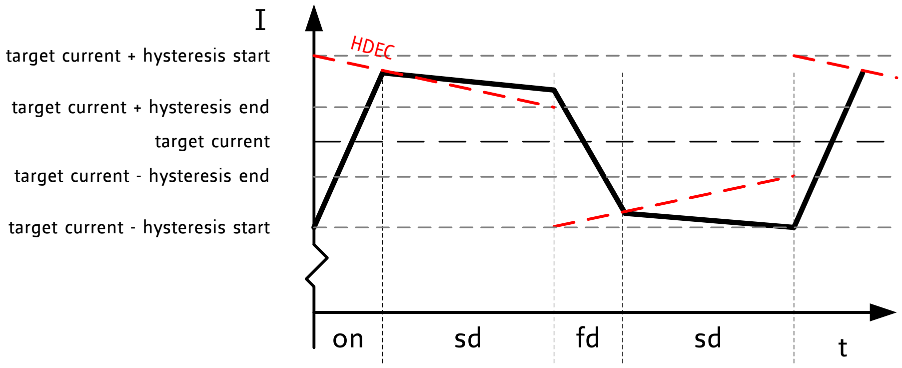


Figure 8.3 spreadCycle chopper scheme showing coil current during a chopper cycle

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

Even at HSTRT=0 and HEND=0, the TMC5130A sets a minimum hysteresis via analog circuitry.

Example:

In the 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 1 or 2 and a short TBL of 1 or 0.

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. In combination with RDSon current sensing without external sense resistors, this chopper mode can bring a benefit with regard to audible high-pitch chopper noise. 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.

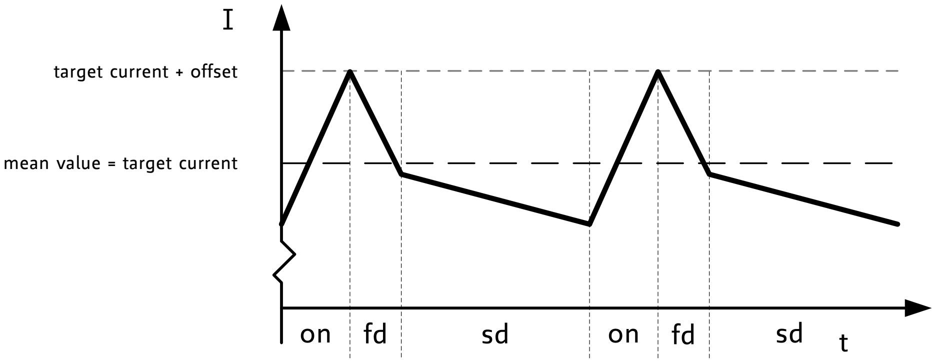


Figure 8.4 Classic const. off time chopper with offset showing coil current

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.

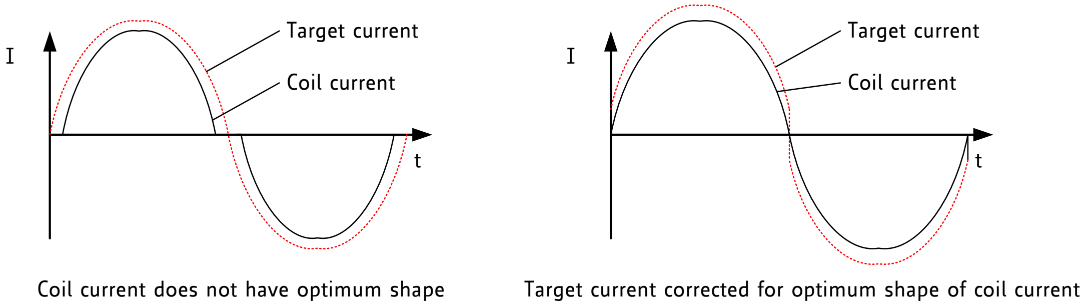


Figure 8.5 Zero crossing with classic chopper and correction using sine wave offset

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

8.3 Random Off Time

In the constant off-time chopper mode, both coil choppers run freely without synchronization. The frequency of each chopper mainly depends on the coil current and the motor coil inductance. The inductance varies with the microstep position. With some motors, a slightly audible beat can occur between the chopper frequencies when they are close together. This typically occurs at a few microstep positions within each quarter wave. This effect is usually not audible when compared to mechanical noise generated by ball bearings, etc. Another factor which can cause a similar effect is a poor layout of the sense resistor GND connections.

Hint
A common factor, which can cause motor noise, is a bad PCB layout causing coupling of both sense resistor voltages (please refer layout hints in chapter 31).

To minimize the effect of a beat between both chopper frequencies, an internal random generator is provided. It modulates the slow decay time setting when switched on by the *rndtf* bit. The *rndtf* feature further spreads the chopper spectrum, reducing electromagnetic emission on single frequencies.

Parameter	Description	Setting	Comment
rndtf	This bit switches on a *random off time* generator, which slightly modulates the off time *TOFF* using a random polynomial.	0	disable
		1	random modulation enable

8.4 chopSync2 for Quiet 2-Phase Motor

chopSync2 is an alternative add-on concept for spreadCycle chopper and constant off time chopper to optimize motor noise at low velocities. When using stealthChop for low velocity operation, chopSync2 is not applicable.

While a frequency adaptive chopper like spreadCycle provides excellent high velocity operation, in some applications, a constant frequency chopper is preferred rather than a frequency adaptive chopper. This may be due to chopper noise in motor standstill, or due to electro-magnetic emission. chopSync2 provides a means to synchronize the choppers for both coils with a common clock, by extending the off time of the coils. It integrates with both chopper principles. However, a careful set up of the chopper is necessary, because chopSync2 can just increment the off times, but not reduce the duration of the chopper cycles themselves. Therefore, it is necessary to test successful operation best with an oscilloscope. Set up the chopper as detailed above, but take care to have chopper frequency higher than the chopSync2 frequency. As high motor velocities take advantage of the normal, adaptive chopper style, chopSync2 becomes automatically switched off using the *VHIGH* velocity limit programmed within the motion controller.

A suitable chopSync2 *SYNC* value can be calculated as follows:

$$SYNC = \left\lfloor \frac{f_{CLK}}{64 * f_{SYNC}} \right\rfloor$$

Example:

The motor is operated in spreadCycle mode (*chm*=0). The minimum chopper frequency for standstill and slow motion (up to *VHIGH*) has been determined to be 25 kHz under worst case operation conditions (hot motor, low supply voltage). The standstill noise needs to be minimized by using chopSync. The IC uses an external 16 MHz clock.

Considering the chopper mode 0, *SYNC* has to be set for the closest value resulting in or below the double frequency, e.g. 50 kHz. Using above formula, a value of 5 results exactly and can be used. Trying a value of 6, a frequency of 41.7 kHz results, which still gives an effective chopper frequency of slightly above 20 kHz, and thus would also be a valid solution. A value of 7 might still be good, but could already give high frequency noise.

In chopper mode 1, *SYNC* could be set to any value between 10 and 13 to be within the chopper frequency range of 19.8 kHz to 25 kHz.

Parameter	Description	Setting	Comment
SYNC	This register allows synchronization of the chopper for both phases of a two phase motor in order to avoid the occurrence of a beat, especially at low motor velocities. It is automatically switched off above *VHIGH*. *Hint:* Set *TOFF* to a low value, so that the chopper cycle is ended, before the next sync clock pulse occurs. Set *SYNC* for the double desired chopper frequency for *chm*=0, for the desired base chopper frequency for *chm*=1.	0	chopSync off
		1…15	f_{CLK}/64 … f_{CLK}/(15*64)

9 Analog Current Control AIN

When a high flexibility of the output current scaling is desired, the analog input of the driver can be enabled for current control, rather than choosing a different set of sense resistors or scaling down the run current via *IRUN* parameter. This way, a simple voltage divider can be used for the adaptation of a board to different motors.

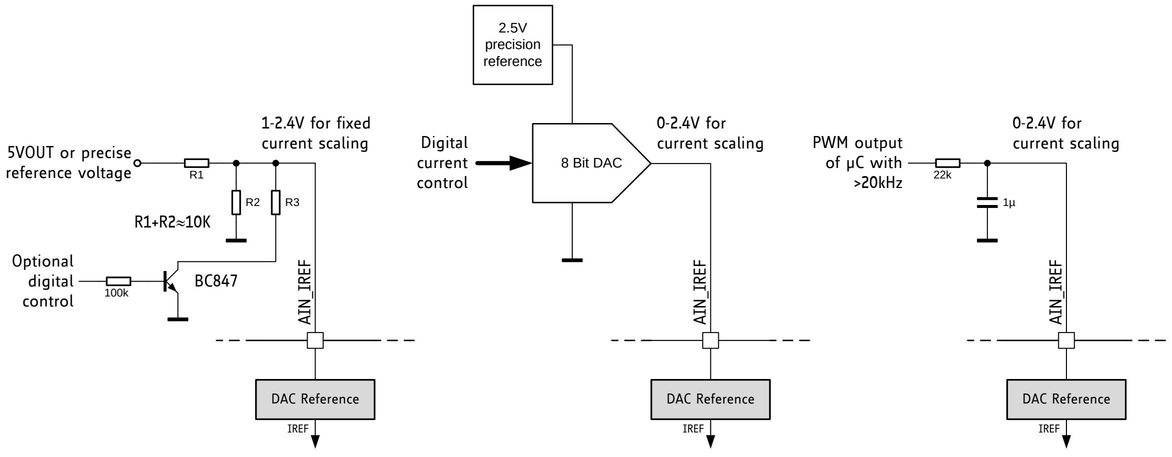
AIN SCALES THE MOTOR CURRENT

The TMC5130A provides an internal reference voltage for current control, directly derived from the 5VOUT supply output. Alternatively, an external reference voltage can be used. This reference voltage becomes scaled down for the chopper comparators. The chopper comparators compare the voltages on BRA and BRB to the scaled reference voltage for current regulation. When *I_scale_analog* in *GCONF* is enabled, the external voltage on AIN is amplified and filtered and becomes used as reference voltage. A voltage of 2.5V (or any voltage between 2.5V and 5V) gives the same current scaling as the internal reference voltage. A voltage between 0V and 2.5V linearly scales the current between 0 and the current scaling defined by the sense resistor setting. It is not advised to work with reference voltages below about 0.5V to 1V, because relative analog noise caused by digital circuitry has an increased impact on the chopper precision at low AIN voltages. For best precision, choose the sense resistors in a way that the desired maximum current is reached with AIN in the range 2V to 2.4V. Be sure to optimize the chopper settings for the normal run current of the motor.

DRIVING AIN

The easiest way to provide a voltage to AIN is to use a voltage divider from a stable supply voltage or a microcontroller's DAC output. A PWM signal can also be used for current control. The PWM becomes transformed to an analog voltage using an additional R/C low-pass at the AIN pin. The PWM duty cycle controls the analog voltage. Choose the R and C values to form a low pass with a corner frequency of several milliseconds while using PWM frequencies well above 10 kHz. AIN additionally provides an internal low-pass filter with 3.5kHz bandwidth. The integration of an NTC into the voltage divider feeding AIN allows the realization of temperature dependent motor current scaling. When a precise reference voltage is available (e.g. from TL431A), the precision of the motor current regulation can be improved when compared to the internal voltage reference.

Using a low reference voltage (e.g. below 1V), for adaptation of a high current driver to a low current motor will lead to reduced analog performance. Adapting the sense resistors to fit the desired motor current gives a better result.



Fixed resistor divider to set current scale
(use external reference for enhanced precision)

Precision current scaler

Simple PWM based current scaler

Figure 9.1 Scaling the motor current using the analog input

10 Current Setting

The internal 5V supply voltage available at the pin 5VOUT is used as a reference for the coil current regulation based on the sense resistor voltage measurement. The desired maximum motor current is set by selecting an appropriate value for the sense resistor. The sense resistor voltage range can be selected by the *vsense* bit in *CHOPCONF*. The low sensitivity setting (high sense resistor voltage, *vsense*=0) brings best and most robust current regulation, while high sensitivity (low sense resistor voltage, *vsense*=1) reduces power dissipation in the sense resistor. The high sensitivity setting reduces the power dissipation in the sense resistor by nearly half.

After choosing the *vsense* setting and selecting the sense resistor, the currents to both coils are scaled by the 5-bit current scale parameters (*IHOLD*, *IRUN*). The sense resistor value is chosen so that the maximum desired current (or slightly more) flows at the maximum current setting (*IRUN* = %11111).

Using the internal sine wave table, which has the amplitude of 248, the RMS motor current can be calculated by:

$$I_{RMS} = \frac{CS+1}{32} * \frac{V_{FS}}{R_{SENSE} + 20m\Omega} * \frac{1}{\sqrt{2}}$$

The momentary motor current is calculated by:

$$I_{MOT} = \frac{CUR_{A/B}}{248} * \frac{CS+1}{32} * \frac{V_{FS}}{R_{SENSE} + 20m\Omega}$$

CS is the current scale setting as set by the *IHOLD* and *IRUN* and coolStep.
V_{FS} is the full scale voltage as determined by *vsense* control bit (please refer to electrical characteristics, V_{SRTL} and V_{SRTH}).
$CUR_{A/B}$ is the actual value from the internal sine wave table.

When *I_scale_analog* is enabled for analog scaling of V_{FS}, the resulting voltage V_{FS}' is calculated by:

$$V'_{FS} = V_{FS} * \frac{V_{AIN}}{2.5V}$$

with V_{AIN} the voltage on pin AIN_IREF in the range 0V to $V_{5VOUT}/2$

CHOICE OF R_{SENSE} AND RESULTING MAX. MOTOR CURRENT		
R_{SENSE} [Ω]	***RMS current [A] (CS=31, vsense=0)***	***RMS current [A] (CS=31, vsense=1)***
1.00	0.23	0.12
0.82	0.27	0.15
0.75	0.30	0.17
0.68	0.33	0.18
0.50	0.44	0.24
0.47	0.47	0.26
0.33	0.66	0.36
0.27	0.79	0.44
0.22	0.96	0.53
0.15	1.35	0.75
0.12	1.64	0.91
0.10	1.92*)	1.06

*) Value exceeds upper current rating.

> *Hint*
> For best precision of current setting, it is advised to measure and fine tune the current in the application.

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
vsense	Allows control of the sense resistor *voltage range* for full scale current.	0	V_{FS} = 0.32 V
		1	V_{FS} = 0.18 V

10.1 Sense Resistors

Sense resistors should be carefully selected. The full motor current flows through the sense resistors. They also see the switching spikes from the MOSFET bridges. A low-inductance type such as film or composition resistors is required to prevent spikes causing ringing on the sense voltage inputs leading to unstable measurement results. A low-inductance, low-resistance PCB layout is essential. Any common GND path for the two sense resistors must be avoided, because this would lead to coupling between the two current sense signals. A massive ground plane is best. Please also refer to layout considerations in chapter 31.

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.

The peak sense resistor power dissipation is:

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

For high current applications, power dissipation is halved by using the low *vsense* setting and using an adapted resistance value. Please be aware, that in this case any voltage drop in PCB traces has a larger influence on the result. A compact layout with massive ground plane is best to avoid parasitic resistance effects.

11 RDSon Based Measurement Eliminates Sense Resistors

The TMC5130A provides the option to eliminate external sense resistors. In this mode the external sense resistors become omitted (shorted) and the internal on-resistance of the power MOSFETs is used for current measurement (see Figure 3.3). As MOSFETs are both, temperature dependent and subject to production stray, a tiny external resistor connected from +5VOUT to AIN/IREF is used to provide a precise absolute current reference. This resistor converts the 5V voltage into a reference current. Be sure to directly attach BRA and BRB pins to GND in this mode near the IC package. The mode is enabled by setting *internal_Rsense* in *GCONF*.

11.1 Limitations of RDSon Sensing

While the RDSon based measurements bring benefits concerning cost and size of the driver, it gives slightly less precise current setting when compared to external sense resistors. External sense resistors provide the possibility to adapt the driver to a wide range of motor operation currents without trading in more noise due to scaling down chopper comparator reference voltage. Therefore the use of RDSon based measurement should be considered for motor currents between 0.4A RMS and 1.2A RMS. For lower and higher current motors, the performance should be evaluated first. Low current motors (below 0.4A RMS) also perform well when using RDSon measurement in combination with the stealthChop mode, because digital scaling does not directly increase chopper noise and reduce effective microstep performance. RDSon sensing may suffer from increased chopper noise and reduced microstep precision in combination with spreadCycle, because the current measurement required for spreadCycle does not guarantee the same precision and symmetry as a sense resistor can deliver. Therefore, consider using classic constant off time chopper instead of spreadCycle in case audible high pitch chopper noise appears.

11.2 Dimensioning of Reference Resistor

For RDSon measurement, up to 1.5A (2A max.) peak current can be driven into the motor (with *vsense*=0). An external reference current into the AIN/IREF pin is used as a reference current. AIN/IREF input resistance is about 1kOhm. In order to realize a certain current a single resistor (R_{REF}) can be connected between 5VOUT and AIN/IREF (pls. refer the table for the choice of the resistor). The resulting current into AIN/IREF is amplified 3000 times. Thus, a current of 0.5mA yields a motor current of 1.5A peak. When using reference currents above 0.5mA resulting in higher theoretical current settings of up to 2A, the resulting current decreases linearly when chip temperature exceeds a certain maximum temperature. For a 2A setting it decreases from 2A at up to 100°C down to about 1.5A at 150°C. The resulting curve limits the maximum current setting in this mode. For calculation of the reference resistor, the internal resistance of AIN/RREF needs to be considered additionally.

vsense=1 allows a lower peak current setting of about 55% of the value yielded with *vsense*=0 (as specified by V_{SRTH} / V_{SRTL}). For fine tuning use the current scale *CS*.

CHOICE OF R_{REF} FOR OPERATION WITHOUT SENSE RESISTORS		
R_{REF} [Ω]	***Peak current [A] (CS=31, vsense=0)***	***Peak current [A] (CS=31, vsense=1)***
6k8	1.92	1.06
7k5	1.76	0.97
8k2	1.63	0.90
9k1	1.49	0.82
10k	1.36	0.75
12k	1.15	0.63
15k	0.94	0.52
18k	0.79	0.43
22k	0.65	0.36
27k	0.60	0.33
33k	0.54	0.29

In RDSon measurement mode, connect the BRA and BRB pins to GND using the shortest possible path (i.e. lowest possible PCB resistance). In a realistic setup, the effective current will be slightly lower than expected. RDSon based measurement gives best results when combined with classic constant off time chopper or with the voltage PWM stealthChop. When using spreadCycle with RDSon based current measurement, slightly asymmetric current measurement for positive currents (on phase) and negative currents (fast decay phase) can result in chopper noise. This especially occurs at increased die temperature and increased motor current.

Note

The absolute current levels achieved with RDSon based current sensing may depend on PCB layout exactly like with external sense resistors, because trace resistance on BR pins will add to the effective sense resistance. Therefore we recommend to measure and calibrate the current setting within the application.

Thumb rule

RDSon based current sensing works best for motors with up to 1.2A RMS current. The best results are yielded with stealthChop operation in combination with RDSon based current sensing. Consider using classic chopper rather than spreadCycle.

For most precise current control and best results with spreadCycle, it is recommended to use external 1% sense resistors rather than RDSon based current control.

12 Velocity Based Mode Control

The TMC5130A 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.

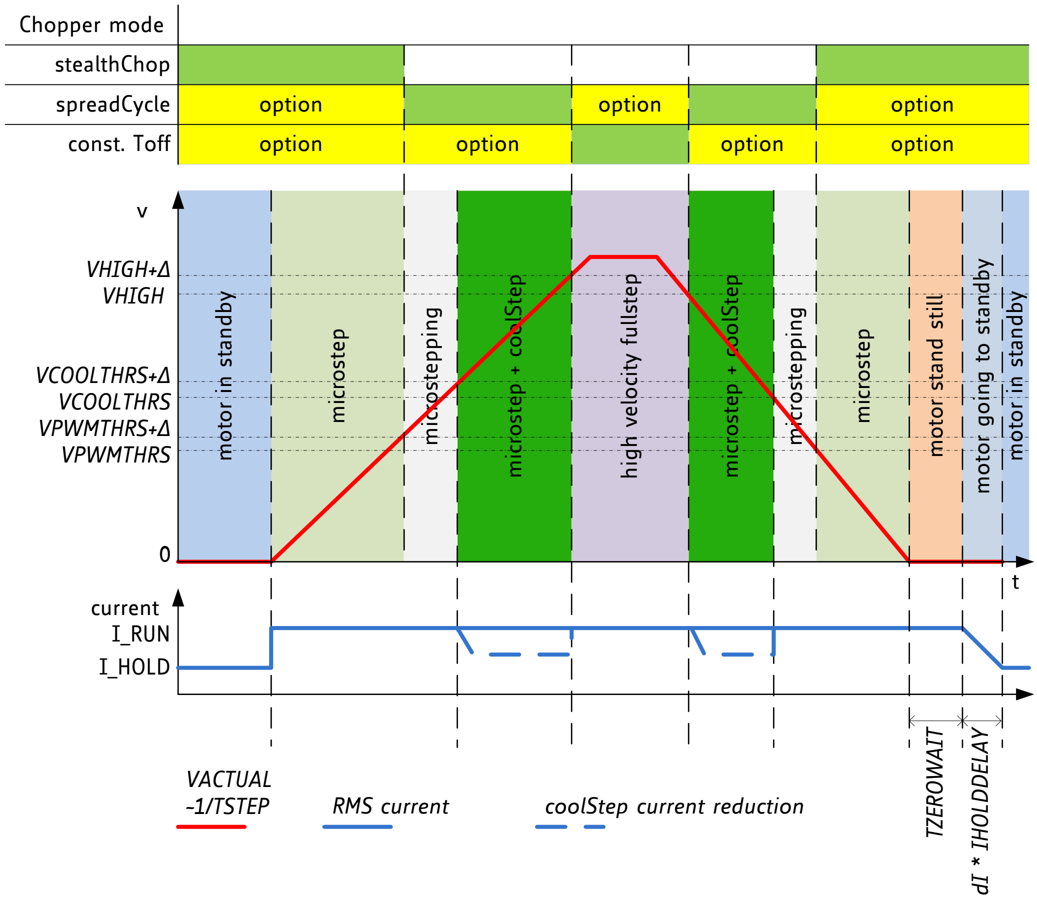


Figure 12.1 Choice of velocity dependent modes

Figure 12.1 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	Hysteresis is 1/16
		1	Hysteresis is 1/32
vhighfs	This bit enables switching to fullstep, when *VHIGH* is exceeded. Switching takes place only at 45° position. The fullstep target current uses the current value from the microstep table at the 45° position.	0	No switch to fullstep
		1	Fullstep at high velocities
vhighchm	This bit enables switching to *chm*=1 and *fd*=0, when *VHIGH* is exceeded. This way, a higher velocity can be achieved. Can be combined with *vhighfs*=1. If set, the *TOFF* setting automatically becomes doubled during high velocity operation in order to avoid doubling of the chopper frequency.	0	No change of chopper mode
		1	Classic const. Toff chopper at high velocities
en_pwm_ mode	stealthChop voltage PWM enable flag (depending on velocity thresholds). Switch from off to on state while in stand still, only.	0	No stealthChop
		1	StealthChop below VPWMTHRS

13 Driver Diagnostic Flags

The TMC5130A drivers supply a complete set of diagnostic and protection capabilities, like short to GND protection and undervoltage detection. A detection of an open load condition allows testing if a motor coil connection is interrupted. See the *DRV_STATUS* table for details.

13.1 Temperature Measurement

The driver integrates a two level temperature sensor (120°C pre-warning and 150°C thermal shutdown) for diagnostics and for protection of the IC against excess heat. Heat is mainly generated by the motor driver stages, and, at increased voltage, by the internal voltage regulator. Most critical situations, where the driver MOSFETs could be overheated, are avoided when enabling the short to GND protection. For many applications, 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.

13.2 Short to GND Protection

The TMC5130A power stages are protected against a short circuit condition by an additional measurement of the current flowing through the high-side MOSFETs. This is important, as most short circuit conditions result from a motor cable insulation defect, e.g. when touching the conducting parts connected to the system ground. The short detection is protected against spurious triggering, e.g. by ESD discharges, by retrying three times before switching off the motor.

Once a short condition is safely detected, the corresponding driver bridge becomes switched off, and the *s2ga* or *s2gb* flag becomes set. In order to restart the motor, the user must intervene by disabling and re-enabling the driver. It should be noted, that the short to GND 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.

13.3 Open Load Diagnostics

Interrupted cables are a common cause for systems failing, e.g. when connectors are not firmly plugged. The TMC5130A 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.

In order to safely detect an interrupted coil connection, read out the open load flags at low or nominal motor velocity operation, only. However, the *ola* and *olb* flags have just informative character and do not cause any action of the driver.

14 Ramp Generator

The TMC5130A 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 14.2.

14.1 Real World Unit Conversion

The TMC5130A 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 TMC5130A register content are written as register[5130A].

PARAMETER VS. UNITS

Parameter / Symbol	**Unit**	**calculation / description / comment**
f_{CLK}[Hz]	[Hz]	clock frequency of the TMC5130A in [Hz]
s	[s]	second
US	µstep	
FS	fullstep	
µstep velocity v[Hz]	µsteps / s	v[Hz] = v[5130A] * (f_{CLK}[Hz]/2 / 2^23)
µstep acceleration a[Hz/s]	µsteps / s^2	a[Hz/s] = a[5130A] * 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[5130A])^2 / a[5130A] / 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 24.

14.2 Ramp Generator Functionality

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

14.2.1 Ramp Mode

The ramp generator delivers two phase acceleration and two phase deceleration ramps with additional programmable start and stop velocities (see Figure 14.1).

> *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.*

14.2.2 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.

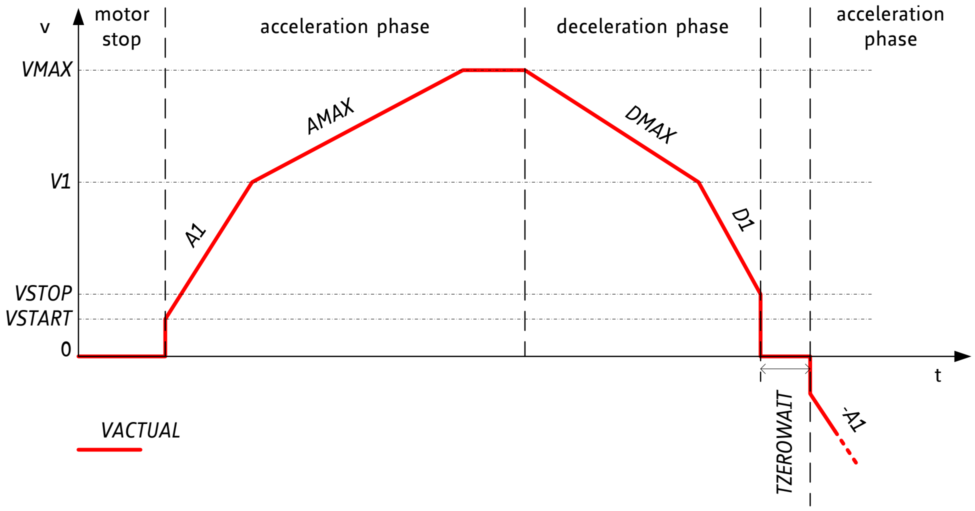


Figure 14.1 Ramp generator velocity trace showing consequent move in negative direction

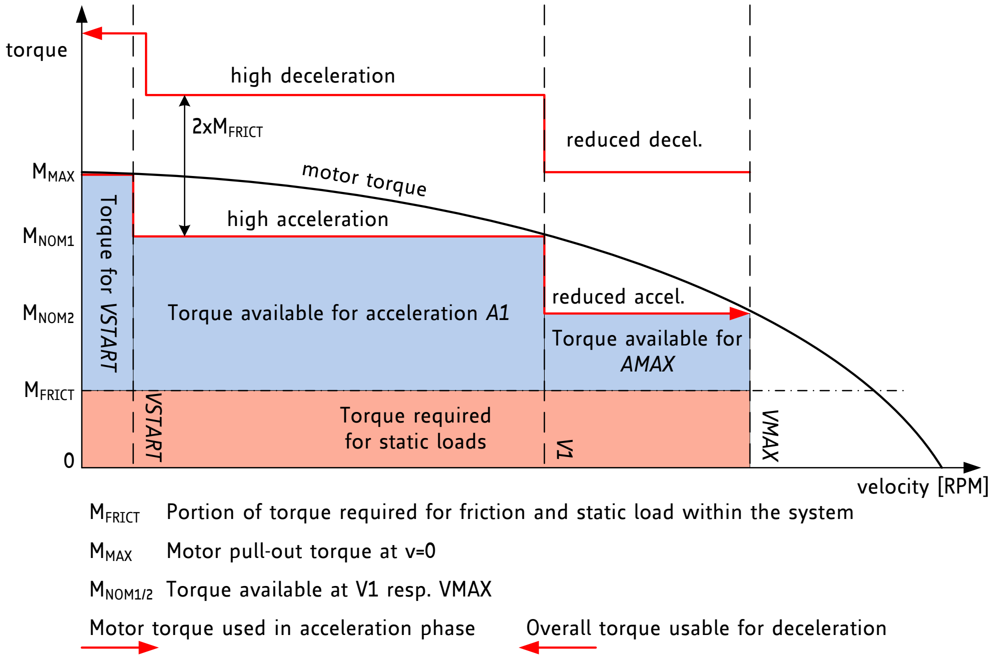


Figure 14.2 Illustration of optimized motor torque usage with TMC5130A ramp generator

14.2.3 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.

14.3 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 14.1. 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.

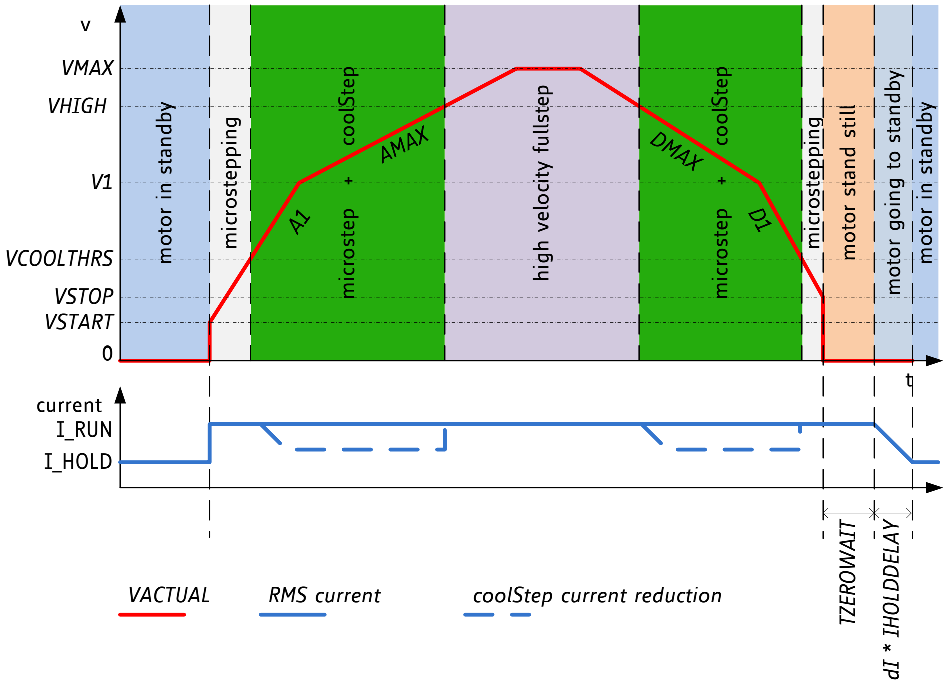


Figure 14.3 Ramp generator velocity dependent motor control

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

14.4 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.

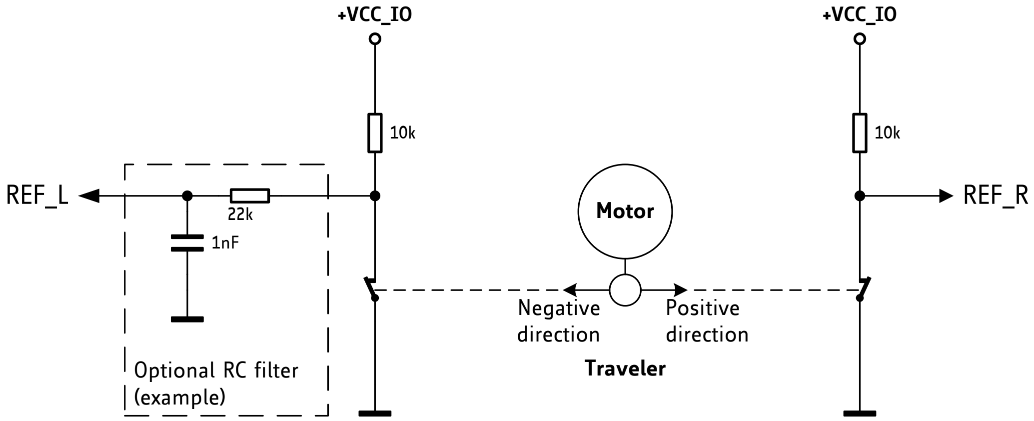


Figure 14.4 Using reference switches (example)

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 TMC5130A 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. Please be aware that reading *RAMP_STAT* may clear flags (e.g. *sg_stop*) and thus the motor may restart after expiration of *TZEROWAIT*. In case the stop condition might be reset by the read and clear (R+C) function, be sure to execute step 5 within the time range set by *TZEROWAIT*.
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, a read access to *RAMP_STAT* clears the stallGuard stop event *event_stop_sg* and releases the motor from the stop condition.

14.5 External STEP/DIR Driver

The TMC5130A allows using the internal ramp generator to control an external STEP/DIR driver like the TRINAMIC TMC262, TMC2660 or TMC389 for powerful stepper applications. In this configuration, the internal driver will normally not be used, but it may be used in addition to the external driver, e.g. when two motors shall move synchronously. The SWN_DIAG0 and SWP_DIAG1 outputs are enabled for STEP and DIR output by setting *GCONF* flags *diag0_step* and *diag1_dir*. Additional internal driver features like dcStep and automatic motor current control are not available in this mode, because there is no feedback from the external driver to the TMC5130A. In order to provide a robust and simple interface, the STEP output uses the edge triggered mode, i.e. it toggles with each (micro)step taken. Enable the *dedge* function on the external driver.

The feature also can be used to provide a step-synchronous signal to external logic.

15 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 15.1. 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.

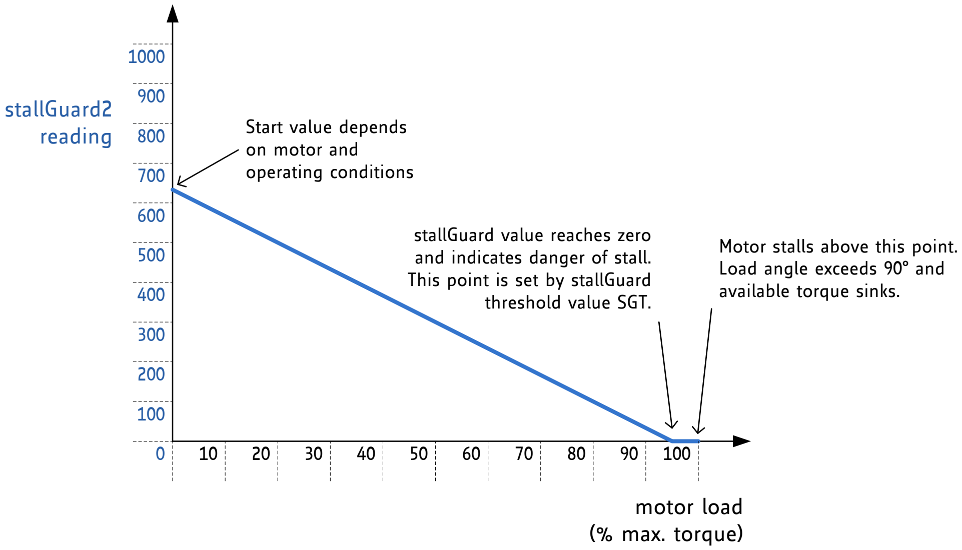


Figure 15.1 Function principle of stallGuard2

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 +1… +63 -1… -64	indifferent value less sensitivity 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 1	standard mode filtered mode
Status word	**Description**	**Range**	**Comment**
SG	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* 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

Attention
In order to use stallGuard2 and coolStep, the stallGuard2 sensitivity should first be tuned using the SGT setting!

15.1 Tuning stallGuard2 Threshold SGT

The stallGuard2 value *SG* 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*.
2. Apply slowly increasing mechanical load to the motor. If the motor stalls before *SG* reaches zero, decrease *SGT*. If *SG* 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. Now enable *sg_stop* and 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 the *RAMP_STAT* register (read and clear function).
4. The optimum setting is reached when *SG* is between 0 and roughly 100 at increasing load shortly before the motor stalls, and *SG* 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* 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* starts rising. With a high *SGT*, *SG* will rise up to the maximum value. Reduce again to the highest value, where *SG* stays at 0. Now the *SGT* value is set as sensibly as possible. When you see *SG* 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 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 *TCOOLSTEP* to match the lower velocity threshold where stallGuard delivers a good result in order to use *sg_stop*.

The system clock frequency affects *SG*. An external crystal-stabilized clock should be used for applications that demand the highest performance. The power supply voltage also affects *SG*, so tighter regulation results in more accurate values. *SG* 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 24.
For detail procedure see Application Note AN002 - *Parameterization of stallGuard2 & coolStep*

15.1.1 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.

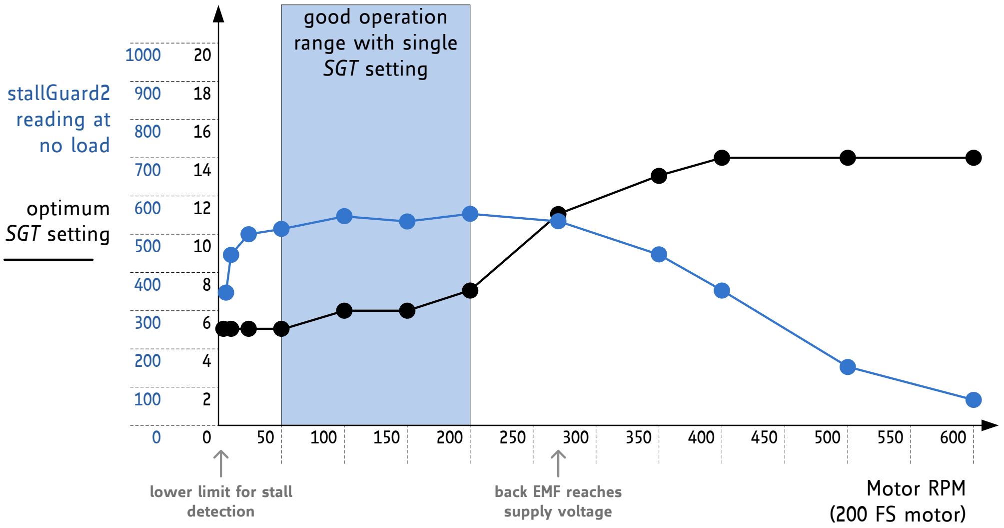


Figure 15.2 Example: optimum SGT setting and stallGuard2 reading with an example motor

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.

15.1.2 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.

15.1.3 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 *SG* at increasing temperature, as motor efficiency is reduced.

15.1.4 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:

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

15.2 stallGuard2 Update Rate and Filter

The stallGuard2 measurement value *SG* 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.

15.3 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* 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* reading.

15.4 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 *RAMP_STAT*. The read and clear function of the *event_stop_sg* flag in *RAMP_STAT* would restart the motor after expiration of *TZEROWAIT* in case the motion parameters have not been modified.

15.5 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* 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.

16 coolStep Operation

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

16.1 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.

16.2 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 16.1 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* falls below *SEMIN*, and coolStep increases the current. When the load decreases, *SG* rises above (*SEMIN* + *SEMAX* + 1) * 32, and the current is reduced.

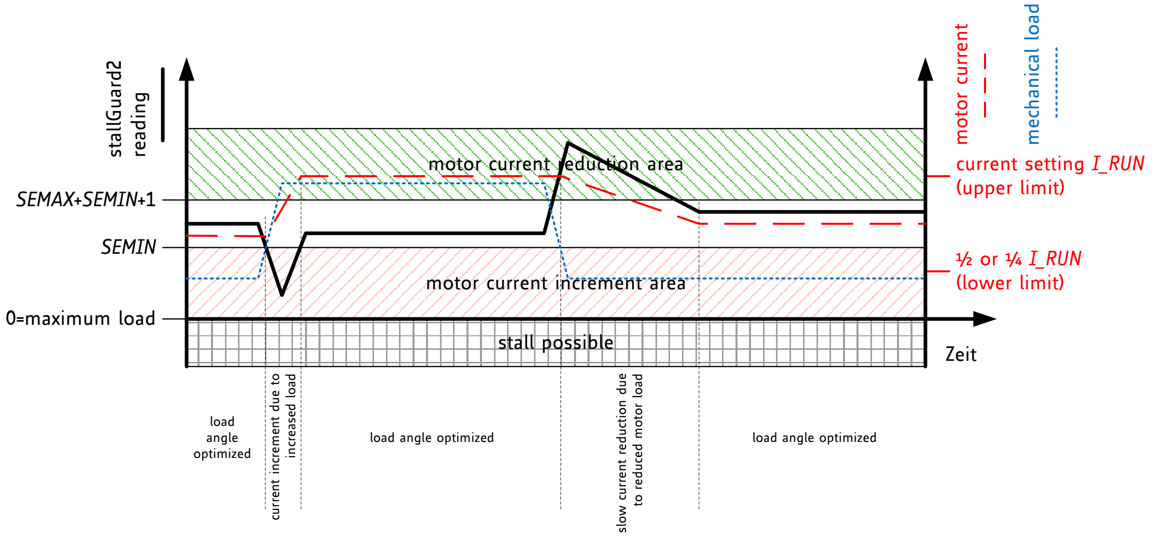


Figure 16.1 coolStep adapts motor current to the load

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
TCOOL THRS	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

16.3 Tuning coolStep

Before tuning coolStep, first tune the stallGuard2 threshold level *SGT*, which affects the range of the load measurement value *SG*. coolStep uses *SG* 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.

16.3.1 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.

16.3.2 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.

17 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 for one or both motors at any time. The only motion controller registers remaining active in this case are the current settings in register *IHOLD_IRUN*.

17.1 Timing

Figure 17.1 shows the timing parameters for the STEP and DIR signals, and the table below gives their specifications. When the DEDGE mode bit in the DRVCTRL 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.

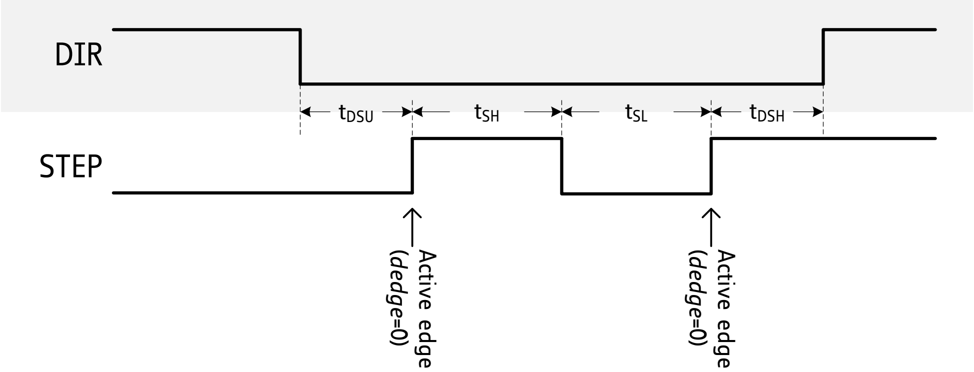


Figure 17.1 STEP and DIR timing

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)			ns
STEP input high time	t_{SH}		max(t_{FILTSD}, t_{CLK}+20)			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	36	60	85	ns
STEP and DIR sampling relative to rising CLK input	$t_{SDCLKHI}$	before rising edge of CLK input		t_{FILTSD}		ns

17.2 Changing Resolution

Sometimes operation of a motor in reduced microstep resolution is desired, in order to stay compatible to an older, less performing driver, or, when using motion controllers with limited frequency capabilities for the STEP/DIR interface. The internal microstep table uses 1024 sine wave entries to generate the wave. The step width taken within the table depends on the microstep resolution setting. Depending on the DIR input, the microstep counter is increased (DIR=0) or decreased (DIR=1) with each STEP pulse by the step width. In principle, the microstep resolution can be changed at any time. The microstep resolution determines the increment respectively the decrement, the sequencer uses for advancing in the microstep table. At maximum resolution, it advances one step for each step pulse. At half resolution, it advances two steps and so on. This way, a change of resolution is possible transparently at each time.

The sequencer has special provision to allow seamless switching between different microstep rates. When the microstep resolution becomes switched to a lower resolution, it calculates the nearest step within the target resolution and reads the current vector at that position. This behavior is especially 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.

Generally, different microstep resolutions are realized by stepping through the internal 256 entry microstep table in more coarse steps. In 256 microstep resolution, 1024 steps are done for a full electrical revolution using an increment of one. The increment is higher for lower resolutions, up to 256 for fullstep. When a lower resolution, each calculated table pointer becomes modified as follows, in order to point to the nearest valid microstep table address in the target resolution:

Fullstep:	The first valid table position is 128 (45° electrical position, i.e. both coils on identical current). This value is the RMS-Value of 0.7 * sine wave amplitude. Step size is 256 (90° electrical)
Half step:	The first valid table position is 64 (22.5° electrical), Step size is 128 (45° steps)
Quarter step:	The first valid table position is 32 (i.e. 90°/8=11.25° electrical), Step size is 64 (22.5° steps)

etc.

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. Especially for full stepping the condition one coil at maximum current and one coil off should be avoided, because in this condition only one coil contributes to the motion at each point of time.

Step position	***MSCNT* value**	**current coil A**	**current coil B**
Half step 0	0	0%	100%
Full step 0	128	70.7%	70.7%
Half step 1	256	100%	0%
Full step 1	384	70.7%	-70.7%
Half step 2	512	0%	-100%
Full step 2	640	-70.7%	-70.7%
Half step 3	768	-100%	0%
Full step 3	896	-70.7%	70.7%

17.3 microPlyer Step Interpolator and Stand Still Detection

For each active edge on STEP, microPlyer produces microsteps at 256x resolution, as shown in Figure 17.2. 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. Operation is only recommended in STEP/DIR mode.

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 16 MHz system clock frequency, this results in a minimum step input frequency of 16 Hz for microPlyer operation. 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*.

Attention
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.

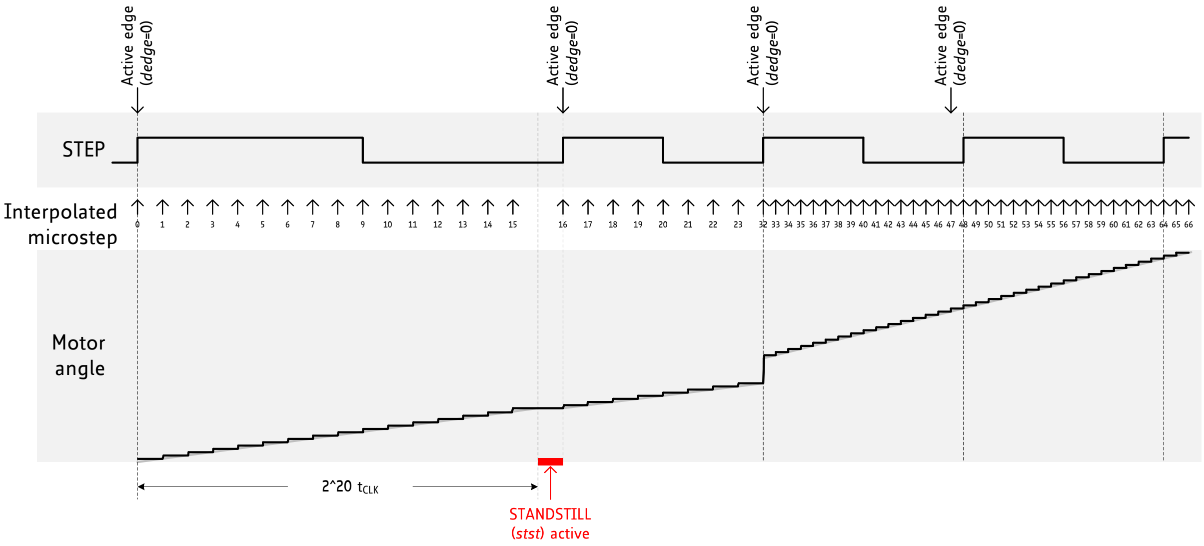


Figure 17.2 microPlyer microstep interpolation with rising STEP frequency (Example: 16 to 256)

In Figure 17.2, 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.

18 DIAG Outputs

18.1 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 18.1 shows the available signals and control bits.

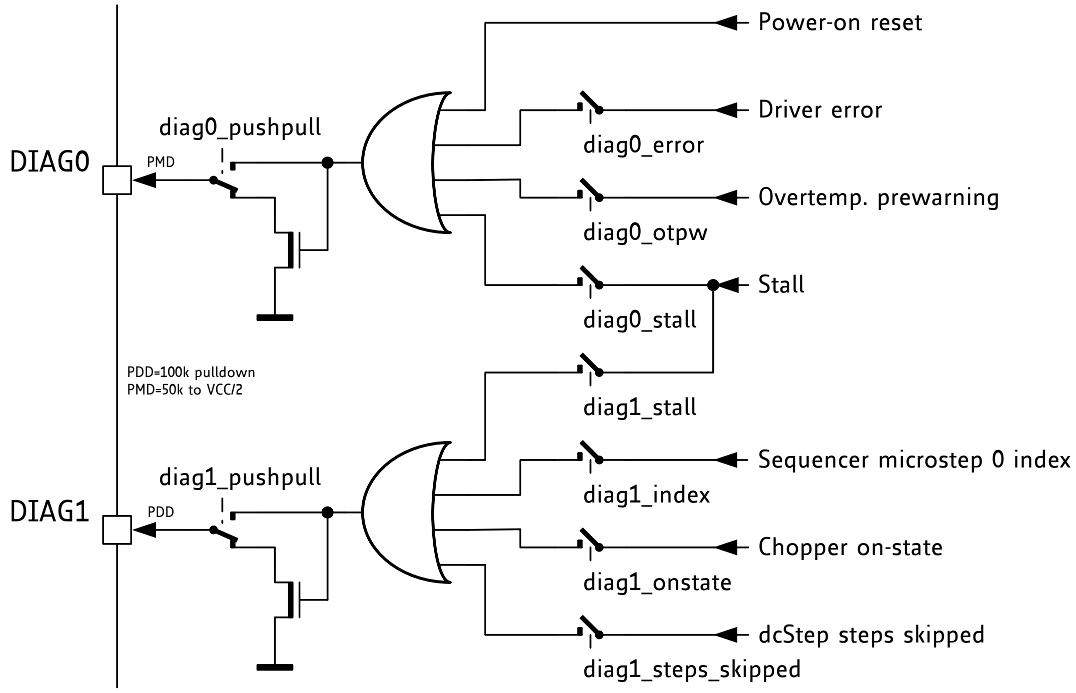


Figure 18.1 DIAG outputs in STEP/DIR mode

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.

18.2 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.

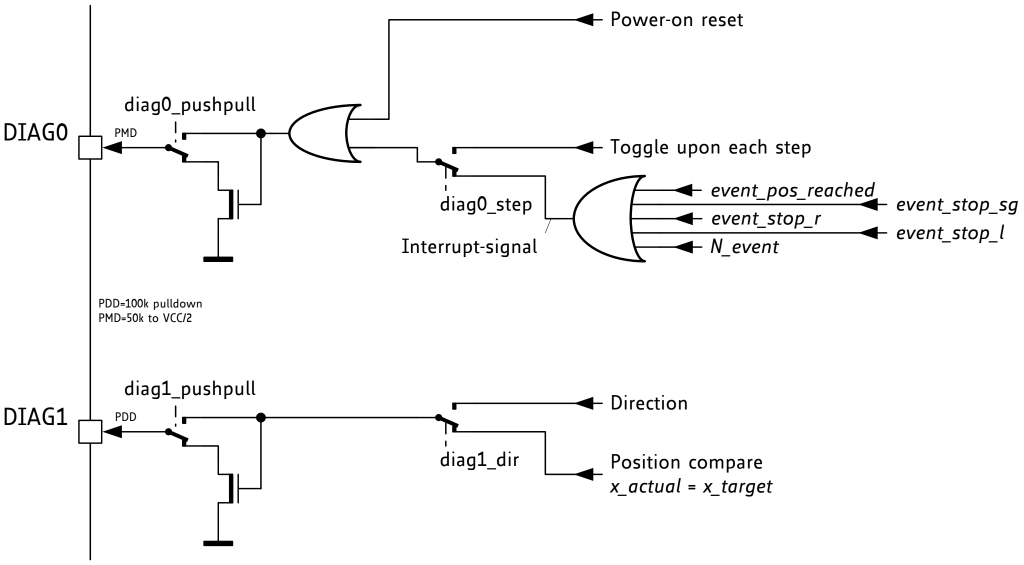


Figure 18.2 DIAG outputs with SD_MODE=0

19 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.

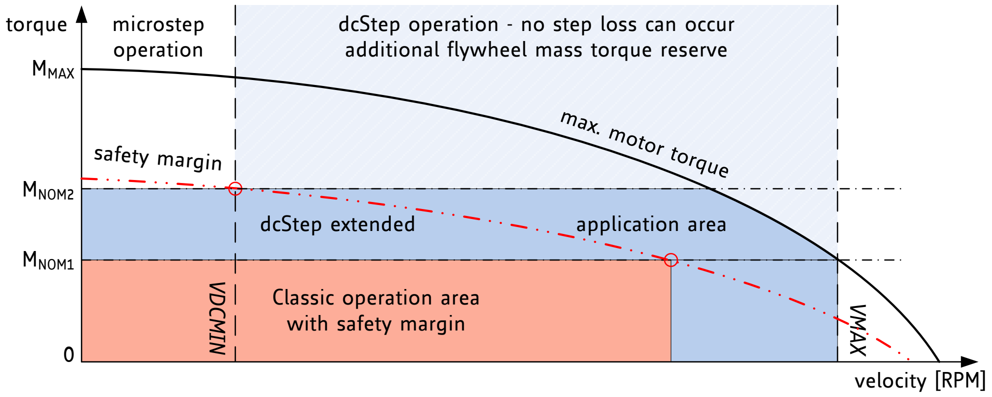
19.1 User Benefits

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!

19.2 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.



M_{NOM}: Nominal torque required by application

M_{MAX}: Motor pull-out torque at v=0

Safety margin: Classical application operation area is limited by a certain percentage of motor pull-out torque

Figure 19.1 dcStep extended application operation area

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 24.
For detail configuration procedure see Application Note AN003 - *dcStep*

19.3 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.

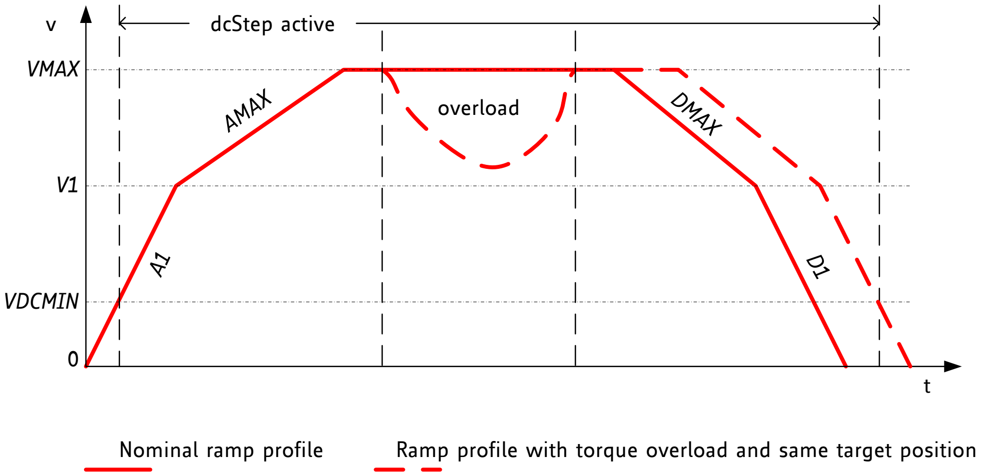


Figure 19.2 Velocity profile with impact by overload situation

> *Attention*
> 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 20.2 for an initialization with the default table.

19.4 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… 255	Lower limit is t_{BLANK} (as defined by *TBL*) in clock cycles + 1
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

19.5 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.

19.6 dcStep with STEP/DIR Interface

The TMC5130A 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 TMC5130A 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.

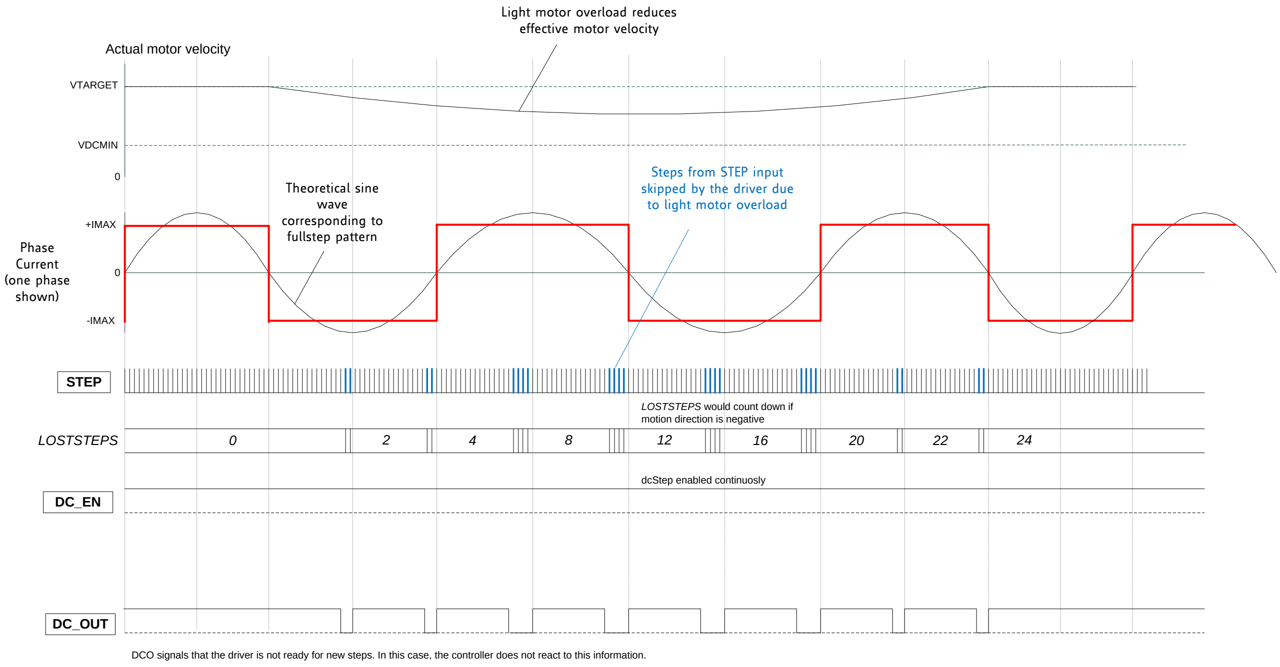
dcStep operation is controlled by three pins in STEP and DIR mode:

- 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.

19.6.1 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 TMC5130A 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 19.3 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.



DCO signals that the driver is not ready for new steps. In this case, the controller does not react to this information.

Figure 19.3 Motor moving slower than STEP input due to light overload. LOSTSTEPS incremented

19.6.2 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 19.5 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.

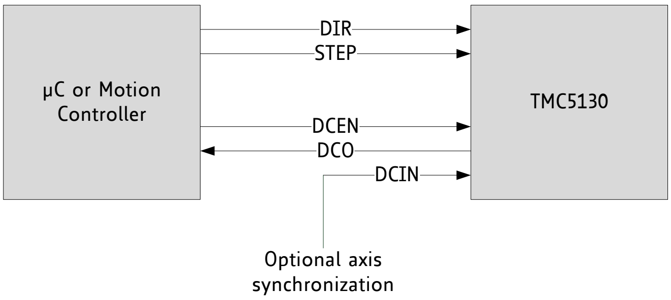


Figure 19.4 Full signal interconnection for dcStep

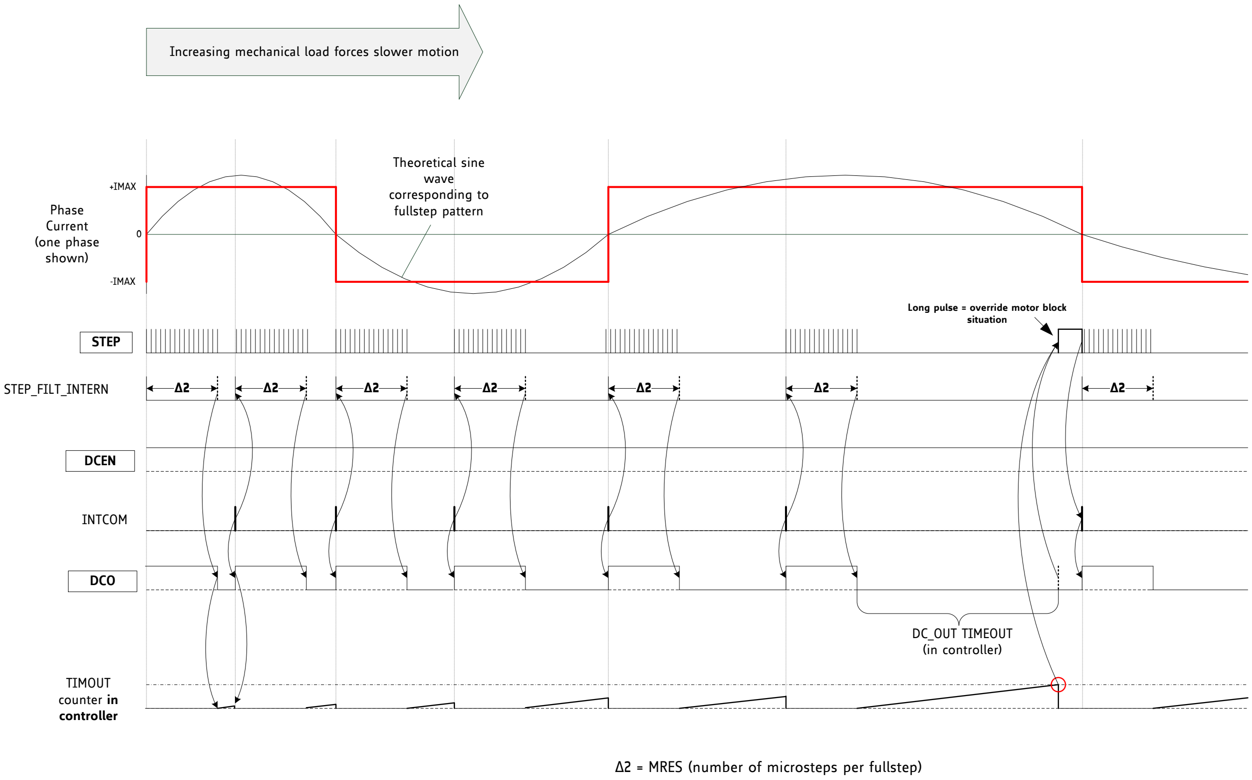


Figure 19.5 DCO Interface to motion controller – step generator stops when DCO is asserted

20 Sine-Wave Look-up Table

The TMC5130A 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.

20.1 User Benefits

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

20.2 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 a 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.

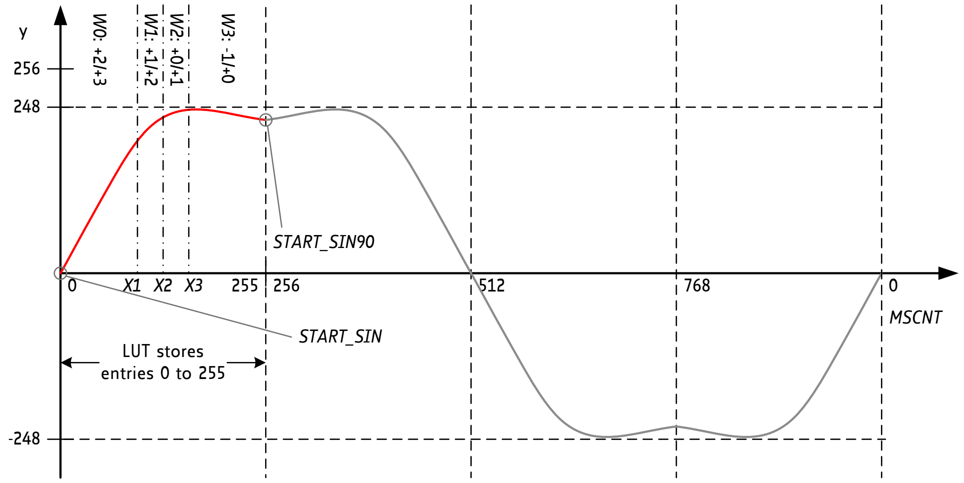


Figure 20.1 LUT programming example

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 TMC5130A-EVAL comes with a calculation tool for own waves.

Initialization example for the default microstep table:

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

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

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

21 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 low, the motor will stop abruptly and go to the power down state, as configured via *IHOLD*, *IHOLD_DELAY* and stealthChop standstill options. Please be aware, that disabling the driver via ENN will require three clock cycles to safely switch off the driver. In case the external CLK fails, it is not safe to disable ENN. In this case, the driver should be reset, i.e. by switching off VCC_IO.

22 ABN Incremental Encoder Interface

The TMC5130A 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 *clear_on_n* = 1 and *clr_once* = 1 or *clr_cont* = 1.

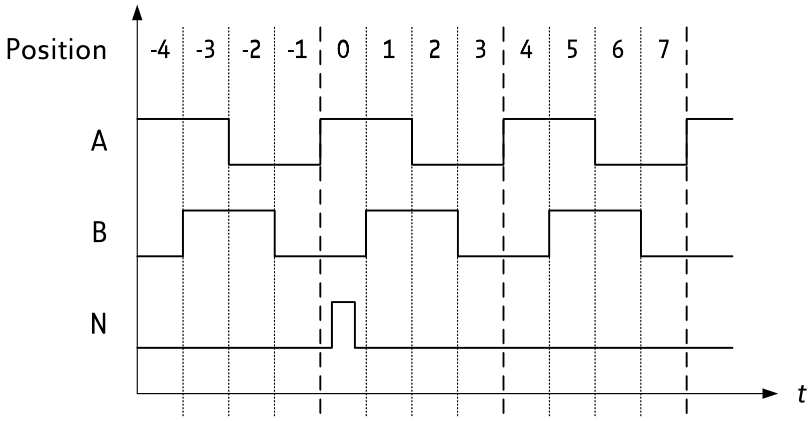


Figure 22.1 Outline of ABN signals of an incremental encoder

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. The sign allows inversion of the counting direction to match motor and encoder direction.

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.

22.1 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}		

22.2 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

$ENC\_CONST = 51 * 2^{16} + 0.2 * 10000$

22.3 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. 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 motion controller TMC4361.

23 DC Motor or Solenoid Operation

The TMC5130A 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_AMPL* = 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 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.

TORQUE LIMITED OPERATION

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% of the current 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.

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).

23.1 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.

24 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

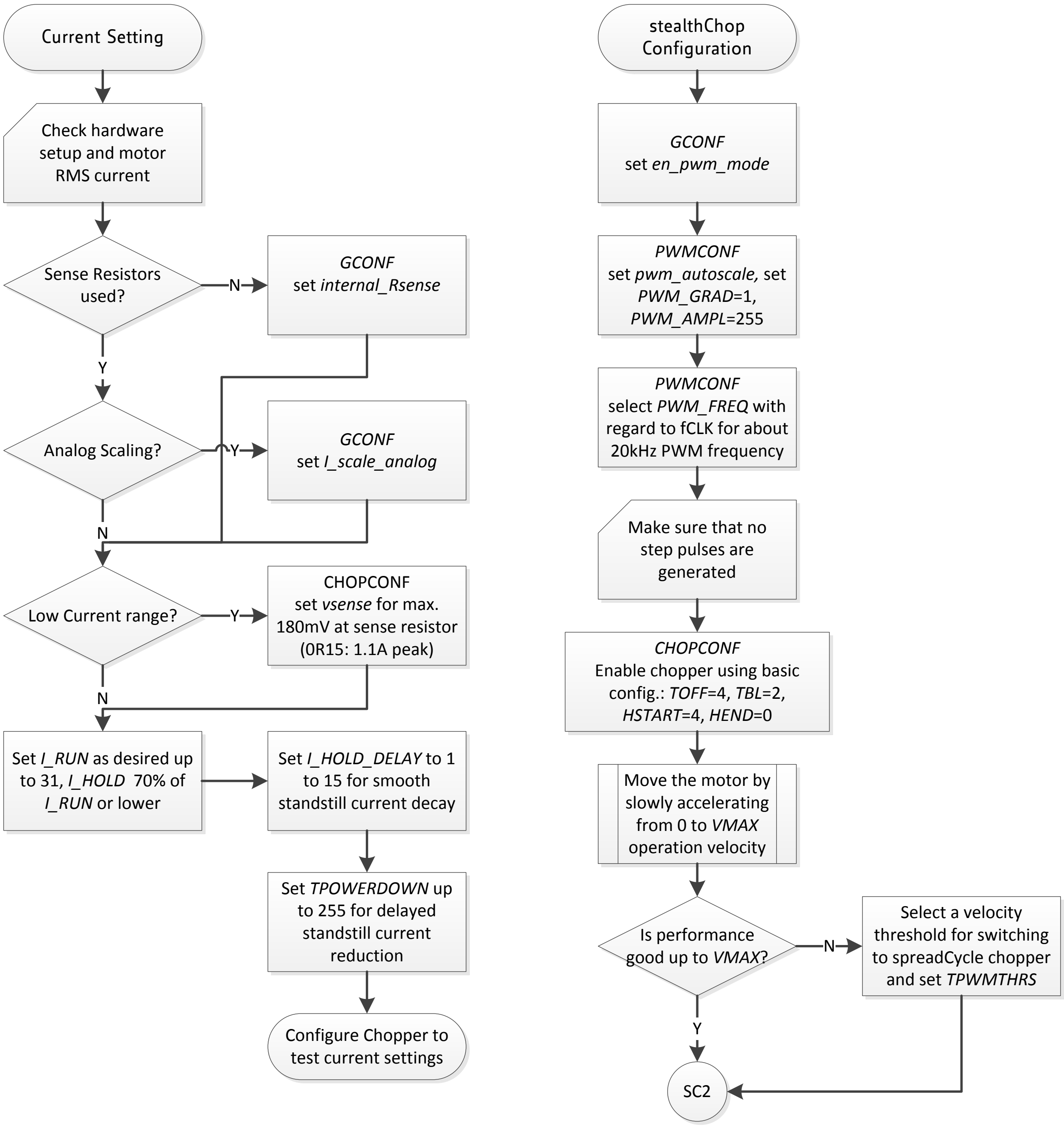


Figure 24.1 Current setting and first steps with stealthChop

TUNING STEALTHCHOP AND SPREADCYCLE

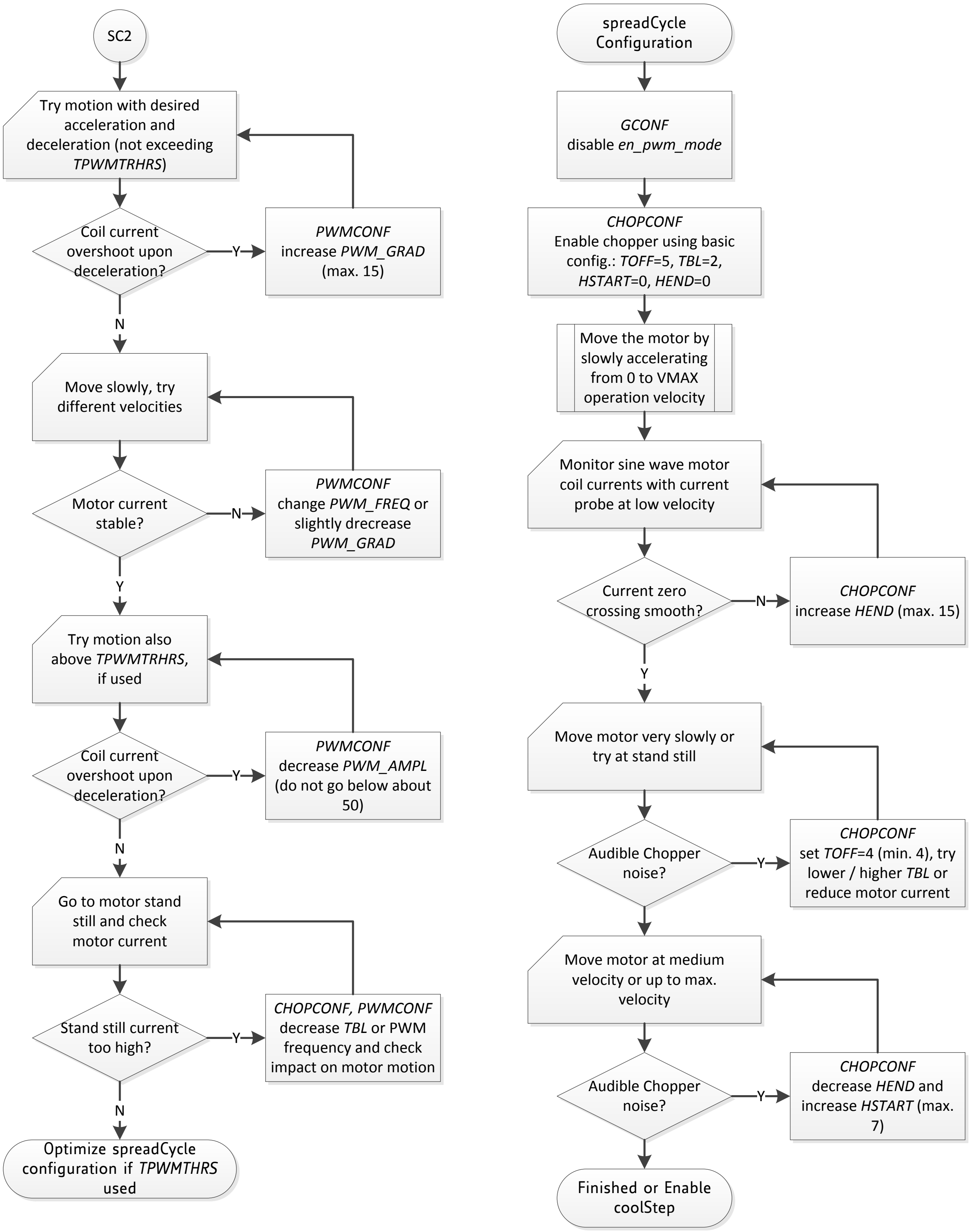


Figure 24.2 Tuning stealthChop and spreadCycle

MOVING THE MOTOR USING THE MOTION CONTROLLER

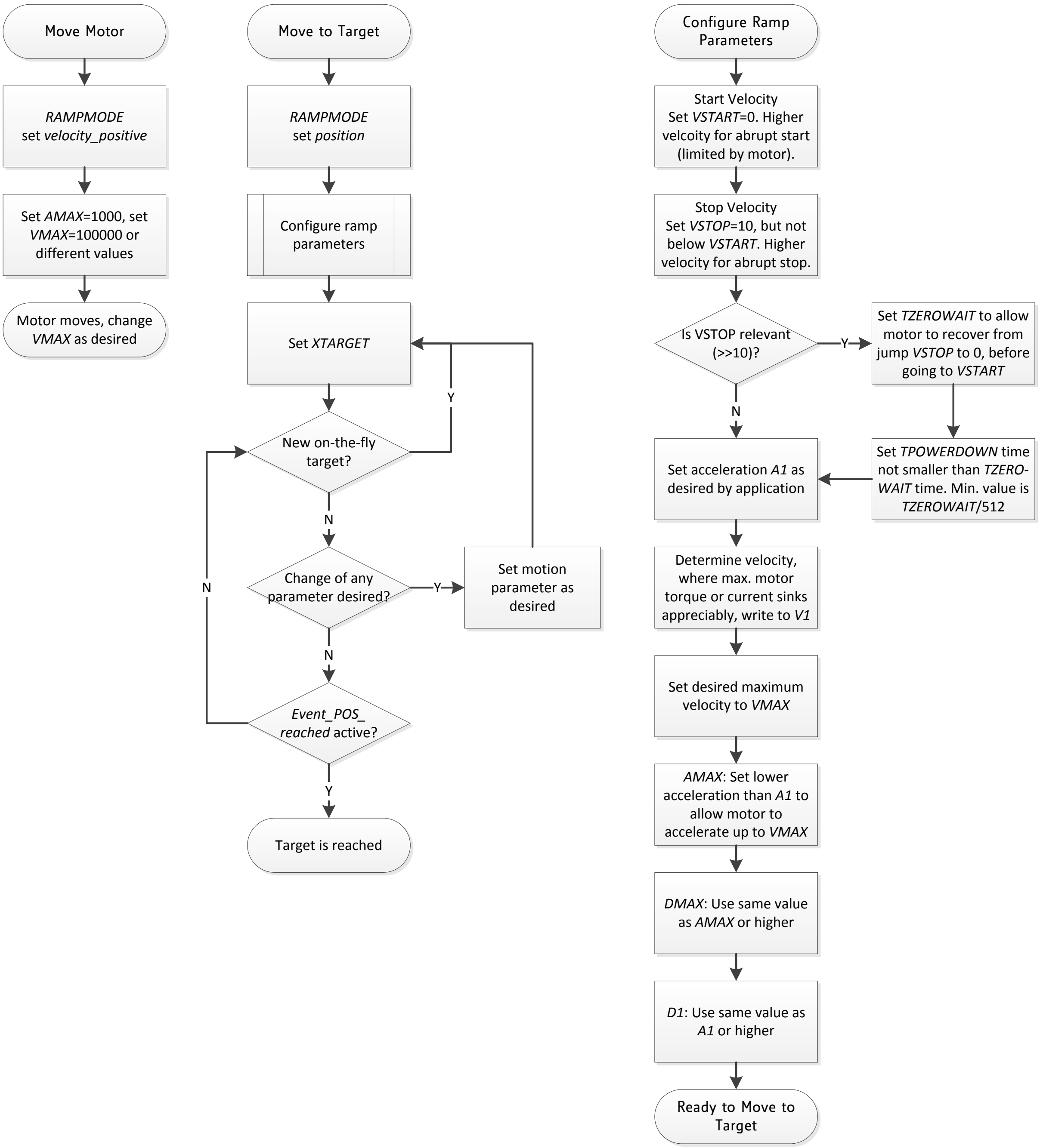


Figure 24.3 Moving the motor using the motion controller

ENABLING COOLSTEP (ONLY IN COMBINATION WITH SPREADCYCLE)

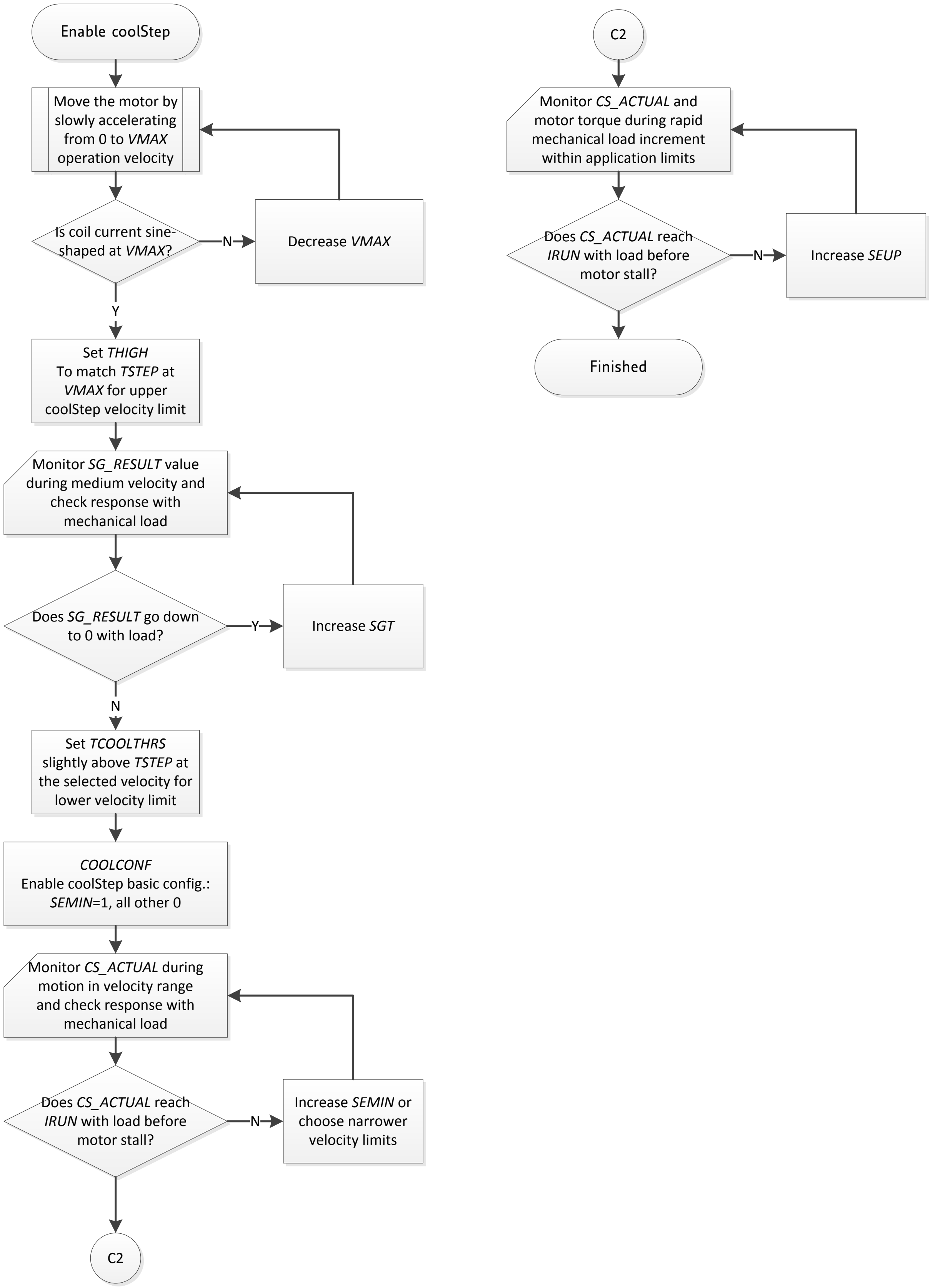


Figure 24.4 Enabling coolStep (only in combination with spreadCycle)

SETTING UP DCSTEP

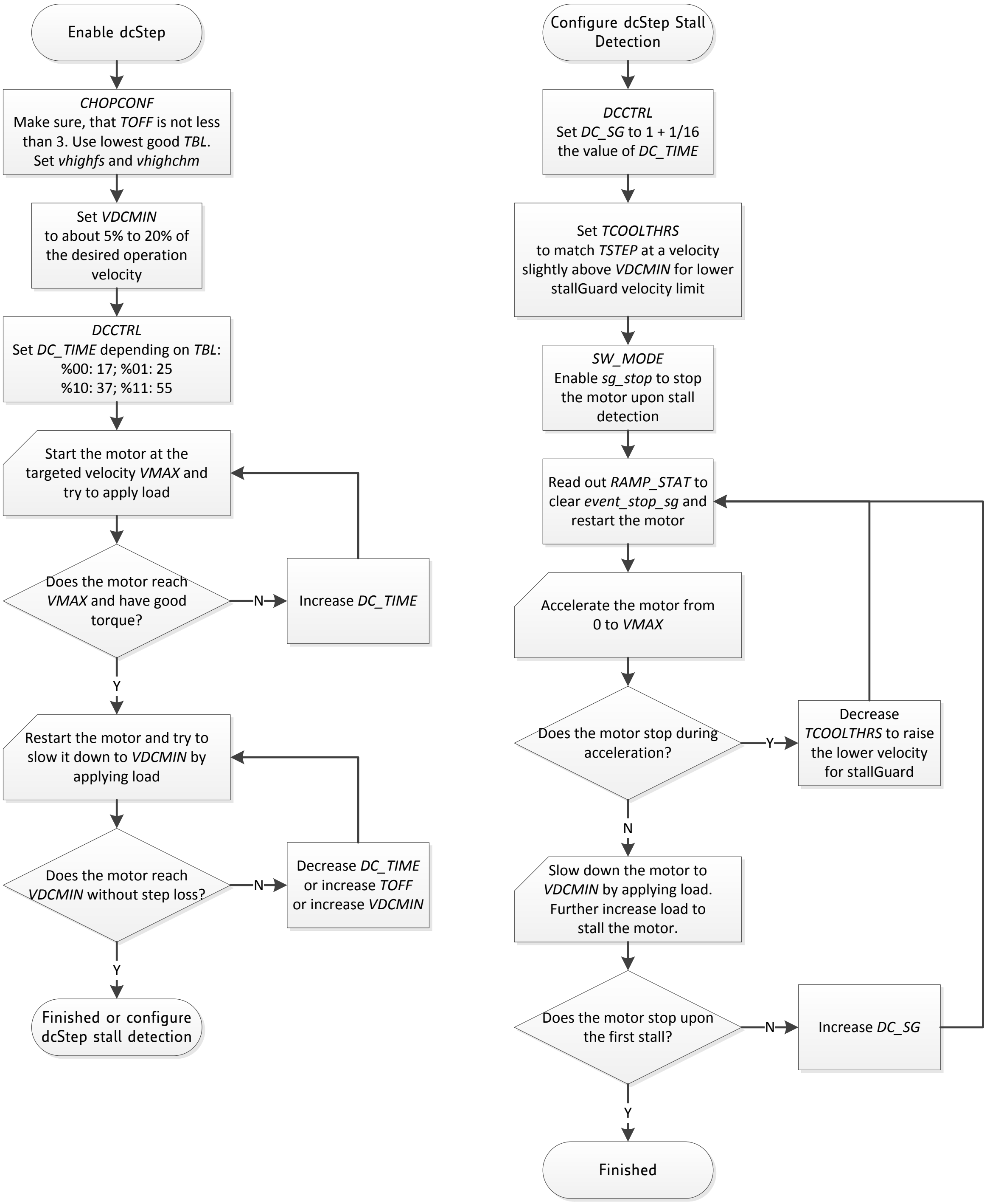


Figure 24.5 Setting up dcStep

25 Getting Started

Please refer to the TMC5130A 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 24 will guide you through the process of correctly setting up all registers.

25.1 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: 0xEC000100C5;  // CHOPCONF: TOFF=5, 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 send: 0xF0000401C8;  // PWM_CONF: AUTO=1, 1/1024 Fclk, Switch amplitude limit=200, Grad=1
```

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.

26 Standalone Operation

For standalone operation, no SPI interface is required to configure the TMC5130A. All pins with suffix CFG0 to CFG6 have a special meaning in this mode. They are evaluated using tristate detection, in order to differentiate between

- CFG pin tied to GND
- CFG pin open (no connection)
- CFG pin tied to VCC_IO

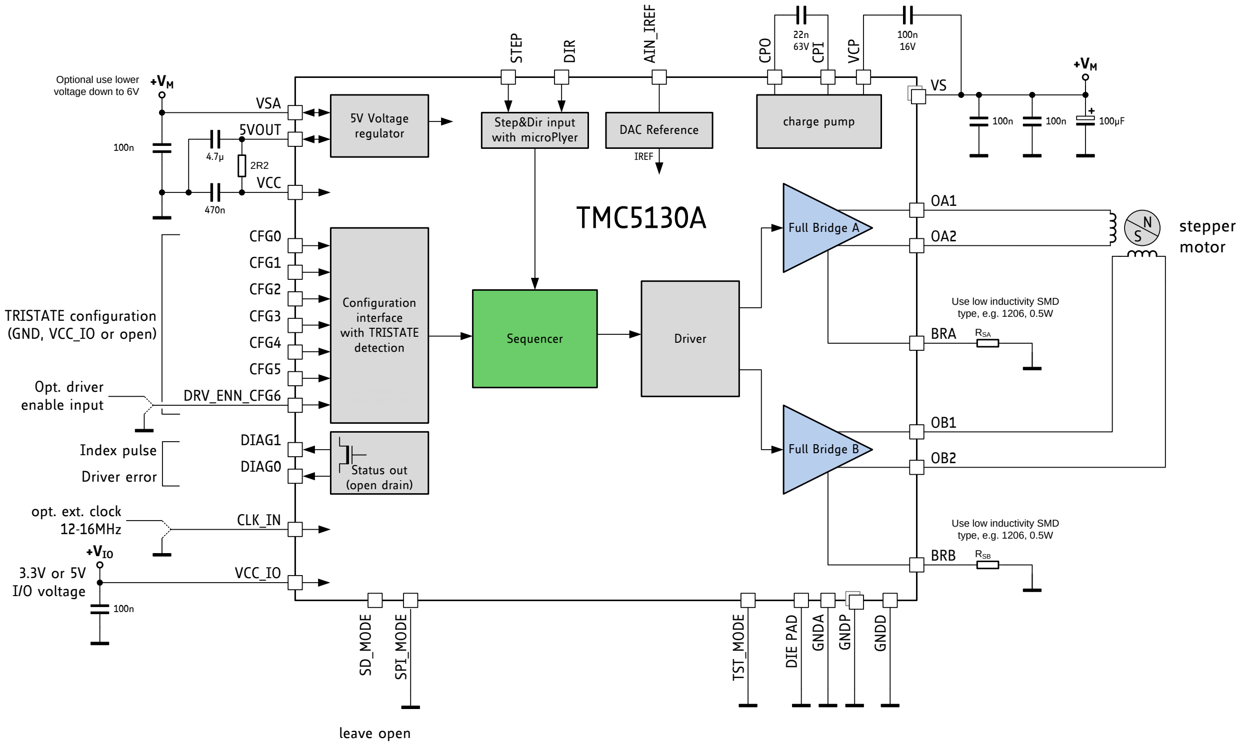


Figure 26.1 Standalone operation with TMC5130A (pins shown with their standalone mode names)

To activate standalone mode, tie pin SPI_MODE to GND. Pin SD_MODE can be left open (high) in this constellation. 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: SETS CHOPPER OFF TIME (DURATION OF SLOW DECAY PHASE)

CFG0	*TOFF* Setting	Registers
GND	140 T_{CLK} (recommended, most universal choice)	*TOFF*=4
VCC_IO	236 T_{CLK}	*TOFF*=7
open	332 T_{CLK}	*TOFF*=10

CFG1 AND CFG2: SETS MICROSTEP RESOLUTION FOR STEP INPUT

CFG2, CFG1	**Microsteps**	**Interpolation**	**Chopper Mode**	**Registers**
GND, GND	1 (Fullstep)	N	spreadCycle	*MRES*=8, *intpol*=0
GND, VCC_IO	2 (Halfstep)	N		*MRES*=7, *intpol*=0
GND, open	2 (Halfstep)	Y, to 256 µsteps		*MRES*=7, *intpol*=1
VCC_IO, GND	4 (Quarterstep)	N		*MRES*=6, *intpol*=0
VCC_IO, VCC_IO	16 µsteps	N		*MRES*=4, *intpol*=0
VCC_IO, open	4 (Quarterstep)	Y, to 256 µsteps		*MRES*=6, *intpol*=1
open, GND	16 µsteps	Y, to 256 µsteps		*MRES*=4, *intpol*=1
open, VCC_IO	4 (Quarterstep)	Y, to 256 µsteps	stealthChop	*MRES*=6, *intpol*=1, *en_PWM_mode*=1
open, open	16 µsteps	Y, to 256 µsteps		*MRES*=4, *intpol*=1, *en_PWM_mode*=1

CFG3: SETS MODE OF CURRENT SETTING

CFG3	**Current Setting**	**Registers**
GND	Internal reference voltage. Current scale set by sense resistors, only.	
VCC_IO	Internal sense resistors. Use analog input current on AIN as reference current for internal sense resistor. This setting gives best results when combined with stealthChop voltage PWM chopper.	*internal_Rsense=1*
open	External reference voltage on pin AIN. Current scale set by sense resistors and scaled by AIN.	*I_scale_analog*=1

CFG4: SETS CHOPPER HYSTERESIS (TUNING OF ZERO CROSSING PRECISION)

CFG4	***HEND* Setting**	**Registers**
GND	5 (recommended, most universal choice)	*HEND*=7
VCC_IO	9	*HEND*=11
open	13	*HEND*=15

CFG5: SETS CHOPPER BLANK TIME (DURATION OF BLANKING OF SWITCHING SPIKE)

CFG5	**Blank time (in number of clock cycles)**	**Registers**
GND	16	*TBL*=%00
VCC_IO	24 (recommended, most universal choice)	*TBL*=%01
open	36	*TBL*=%10

CFG6_ENN: ENABLE PIN AND CONFIGURATION OF STANDSTILL POWER DOWN

CFG6	Motor driver enable	Standstill power down	Registers
GND	Enable	N	*IRUN*=31, *IHOLD*=31
VCC_IO	Disable	- (Driver disable)	
open	Enable	Y, ramp down from 100% to 34% motor current in 44M clock cycles (3 to 4 seconds) if no step pulse for more than 1M clock cycles (standstill). In combination with stealthChop, be sure not to work with too low overall current setting, as regulation will not be able to measure the motor current after stand still current reduction. This will result in very low motor current after the stand-still period.	*IRUN*=31, *IHOLD*=11, *IHOLDDELAY*=8

While the parameters for spreadCycle can be configured for good microstep performance, stealthChop mode is configured with its power on default values (*PWMCONF*=0x00050480):

f_{PWM}=1/683 f_{CLK} (i.e. roughly 19kHz with internal clock)
pwm_autoscale=1
PWM_GRAD=4
PWM_AMPL=128

CFG0 and CFG4 settings do not influence the stealthChop configuration. This way, it is even possible to switch between spreadCycle and stealthChop mode by simply switching CFG1 and CFG2.

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.

Example:
It is desired to do small motions in smooth and noiseless stealthChop mode. For quick motions, spreadCycle is to be used. The controller can deliver 1/16 microstep step signals. Tie together CFG1 and CFG2 and drive them with a three state driver. Switch both to VCC_IO to operate in spreadCycle, switch them to hi-Z (open) state for a motion in stealthChop.

27 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.

28 Clock Oscillator and Clock 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 on-chip clock oscillator provides timing in case no external clock is easily available.

USING THE INTERNAL CLOCK

Directly tie the CLK input to GND near to the IC if the internal clock oscillator is to be used. The internal clock can be calibrated by driving the ramp generator at a certain velocity setting. Reading out position values via the interface and comparing the resulting velocity to the remote masters' clock gives a time reference. A similar procedure also is described in 19.5. For a Step/Dir application, read out *TSTEP* at a defined external step frequency. Scale acceleration and velocity settings, *TOFF* and *PWM_FREQ* as a result. Temperature dependency and ageing of the internal clock is comparatively low.

In case well defined velocity settings and precise motor chopper operation are desired, it is supposed to work with an external clock source.

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 first positive polarity seen on the CLK input.

Attention
Switching off the external clock frequency prevents the driver from operating normally. Therefore be careful to switch off the motor drivers before switching off the clock (e.g. using the enable input), because otherwise the chopper would stop and the motor current level could rise uncontrolled. The short to GND detection stays active even without clock, if enabled.

28.1 Considerations on the Frequency

A higher frequency allows faster step rates, faster SPI operation and higher chopper frequencies. On the other hand, it may cause more electromagnetic emission of the system and causes more power dissipation in the TMC5130A digital core and voltage regulator. Generally a frequency of 10 MHz to 16 MHz should be sufficient for most applications. For reduced requirements concerning the motor dynamics, a clock frequency of down to 8 MHz (or even lower) can be considered.

29 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 ($V_{VS} \geq V_{VSA}$)	V_{VS}, V_{VSA}	-0.5	49	V
Supply and bridge voltage max. *)	V_{VMAX}		50	V
VSA when different from to VS	V_{VSA}	-0.5	V_{VS}+0.5	V
I/O supply voltage	V_{VIO}	-0.5	5.5	V
digital VCC supply voltage (if not supplied by internal regulator)	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	I_{IO}		+/-10	mA
5V regulator output current (internal plus external load)	I_{5VOUT}		50	mA
5V regulator continuous power dissipation (V_{VM}-5V) * I_{5VOUT}	P_{5VOUT}		1	W
Power bridge repetitive output current	I_{Ox}		3.0	A
Junction temperature	T_J	-50	150	°C
Storage temperature	T_{STG}	-55	150	°C
ESD-Protection for interface pins (Human body model, HBM)	V_{ESDAP}		4	kV
ESD-Protection for handling (Human body model, HBM)	V_{ESD}		1	kV

*) Stray inductivity of GND and VS connections 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.

30 Electrical Characteristics

30.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	T_J	-40	125	°C
Supply voltage (using internal +5V regulator)	V_{VS}, V_{VSA}	5.5	46	V
Supply voltage (internal +5V regulator bridged: V_{VCC}=V_{VSA}=V_{VS})	V_{VS}	4.7	5.4	V
I/O supply voltage	V_{VIO}	3.00	5.25	V
VCC voltage when using optional external source (supplies digital logic and charge pump)	V_{VCC}	4.6	5.25	V
RMS motor coil current per coil (value for design guideline)	I_{RMS}		1.4	A
Peak output current per motor coil output (sine wave peak) using external or internal current sensing	I_{Ox}		2.0	A
Peak output current per motor coil output (sine wave peak) for short term operation. Limit $T_J \leq 105°C$, e.g. for 100ms short time acceleration phase below 50% duty cycle.	I_{Ox}		2.5	A

30.2 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_{VCC}	I_S	f_{CLK}=16MHz		15	22	mA
Total supply current, operating, I_{VS} + I_{VSA} + I_{VCC}	I_S	f_{CLK}=16MHz, 23.4kHz chopper, no load		19		mA
Idle supply current from VS, charge pump operating	I_{VS0}	f_{CLK}=0Hz, driver disabled		0.25	0.5	mA
Static supply current from VSA	I_{VSA0}	f_{CLK}=0Hz	1.4	2	3	mA
Supply current, driver disabled, dependency on CLK frequency	I_{VSA}	f_{CLK} variable, additional to I_{VSA0}		0.8		mA/MHz
Internal current consumption from 5V supply on VCC pin	I_{VCC}	f_{CLK}=16MHz, 23.4kHz chopper		16		mA
IO supply current (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					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
RDS_{ON} lowside MOSFET	R_{ONL}	measure at 100mA, 25°C, static state		0.4	0.5	Ω
RDS_{ON} highside MOSFET	R_{ONH}	measure at 100mA, 25°C, static state		0.5	0.6	Ω
slope, MOSFET turning on	t_{SLPON}	measured at 700mA load current (resistive load)	50	120	220	ns
slope, MOSFET turning off	t_{SLPOFF}	measured at 700mA load current (resistive load)	50	120	220	ns
Current sourcing, driver off	I_{OIDLE}	O_{XX} pulled to GND	120	180	250	µA

Charge pump	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Charge pump output voltage	V_{VCP}-V_{VS}	operating, typical f_{chop}<40kHz	4.0	V_{VCC} - 0.3	V_{VCC}	V
Charge pump voltage threshold for undervoltage detection	V_{VCP}-V_{VS}	using internal 5V regulator voltage	3.3	3.6	3.8	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}	I_{5VOUT} = 0mA T_J = 25°C	4.80	5.0	5.25	V
Output resistance	R_{5VOUT}	Static load		3		Ω
Deviation of output voltage over the full temperature range	$V_{5VOUT(DEV)}$	I_{5VOUT} = 16mA T_J = full range		+/-30	+/-100	mV
Deviation of output voltage over the full supply voltage range	$V_{5VOUT(DEV)}$	I_{5VOUT} = 0mA V_{VSA} = variable		+/-15	+/-30	mV / 10V
Deviation of output voltage over the full supply voltage range	$V_{5VOUT(DEV)}$	I_{5VOUT} = 16mA V_{VSA} = variable		-38	+/-75	mV / 10V

Clock oscillator and input	**Timing-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Clock oscillator frequency	f_{CLKOSC}	t_J=-50°C	9	12.4		MHz
Clock oscillator frequency	f_{CLKOSC}	t_J=50°C	10.1	13.2	17.2	MHz
Clock oscillator frequency	f_{CLKOSC}	t_J=150°C		13.4	18	MHz
External clock frequency (operating)	f_{CLK}		4	10-16	18	MHz
External clock high / low level time	t_{CLK}	CLK driven to 0.1 V_{VIO} / 0.9 V_{VIO}	10			Ns
External clock first cycle triggering switching to external clock source	t_{CLK1}	CLK driven high	30	25		ns

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)	2.1	2.55	3.0	V
$V_{VCC\ IO}$ undervoltage detector hysteresis	$V_{UV\_VIOHYST}$			0.3		V
Short to GND detector threshold (V_{VS} - V_{Ox})	V_{OS2G}		2	2.5	3	V
Short to GND detector delay (high side switch on to short detected)	t_{S2G}	High side output clamped to V_{SP}-3V	0.8	1.3	2	µs
Overtemperature prewarning	t_{OTPW}	Temperature rising	100	120	140	°C
Overtemperature shutdown	t_{OT}	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_{SRTL}	*vsense*=0 *csactual*=31 *sin_x*=248 Hyst.=0; I_{BRxy}=0		325		mV
Sense input peak threshold voltage (high sensitivity)	V_{SRTH}	*vsense*=1 *csactual*=31 *sin_x*=248 Hyst.=0; I_{BRxy}=0		180		mV
Sense input tolerance / motor current full scale tolerance -using internal reference	I_{COIL}	*I_scale_analog*=0, *vsense*=0	-5		+5	%
Sense input tolerance / motor current full scale tolerance -using external reference voltage	I_{COIL}	*I_scale_analog*=1, V_{AIN}=2V, *vsense*=0	-2		+2	%
Internal resistance from pin BRxy to internal sense comparator (additional to sense resistor)	R_{BRxy}			20		mΩ

Digital logic levels	**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

AIN/IREF input	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
AIN_IREF input resistance to 2.5V (=5VOUT/2)	R_{AIN}	Measured to GND (*internalRsense*=0)	260	330	400	kΩ
AIN_IREF input voltage range for linear current scaling	V_{AIN}	Measured to GND (*IscaleAnalog*=1)	0	0.5-2.4	V_{5VOUT}/2	V
AIN_IREF open input voltage level	V_{AINO}	Open circuit voltage (*internalRsense*=0)		V_{5VOUT}/2		V
AIN_IREF input resistance to GND for reference current input	R_{IREF}	Measured to GND (*internalRsense*=1)	0.8	1	1.2	kΩ
AIN_IREF current amplification for reference current to coil current at maximum setting	$I_{REFAMPL}$	I_{IREF} = 0.25mA		3000		Times
Motor current full scale tolerance -using RDSon measurement	I_{COIL}	*Internal_Rsense*=1, *vsense*=0, I_{IREF} = 0.25mA	-10		+10	%

30.3 Thermal Characteristics

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

A thermal resistance of 21K/W for a typical board means, that the package is capable of continuously dissipating 4.7W at an ambient temperature of 25°C with the die temperature staying below 125°C.

Parameter	Symbol	Conditions	Typ	Unit
Typical power dissipation	P_D	stealthChop or spreadCycle, 1A RMS in two phase motor, sinewave, 20kHz chopper, 24V, internal supply, 85°C peak surface of package (motor QSH4218-035-10-027)	3.0	W
Thermal resistance junction to ambient on a multilayer board	R_{TMJA}	Dual signal and two internal power plane board (2s2p) as defined in JEDEC EIA JESD51-5 and JESD51-7 (FR4, 35µm CU, 70mm x 133mm, d=1.5mm)	21	K/W
Thermal resistance junction to board	R_{TJB}	PCB temperature measured within 1mm distance to the package leads	8	K/W
Thermal resistance junction to case	R_{TJC}	Junction temperature to heat slug of package	3	K/W

Table 30.1 Thermal characteristics TQFP48-EP

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

Note
A spread-sheet for calculating TMC5130 power dissipation is available on www.trinamic.com.

31 Layout Considerations

31.1 Exposed Die Pad

The TMC5130A uses its die attach pad to dissipate heat from the drivers and the linear regulator to the board. For best electrical and thermal performance, use a reasonable amount of solid, thermally conducting vias between the die attach pad and the ground plane. The printed circuit board should have a solid ground plane spreading heat into the board and providing for a stable GND reference.

31.2 Wiring GND

All signals of the TMC5130A 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
Especially the sense resistors are susceptible to GND differences and GND ripple voltage, as the microstep current steps make up for voltages down to 0.5 mV. No current other than the sense resistor current should flow on their connections to GND and to the TMC5130A. Optimally place them close to the IC, with one or more vias to the GND plane for each sense resistor. The two sense resistors for one coil should not share a common ground connection trace or vias, as also PCB traces have a certain resistance.

31.3 Supply Filtering

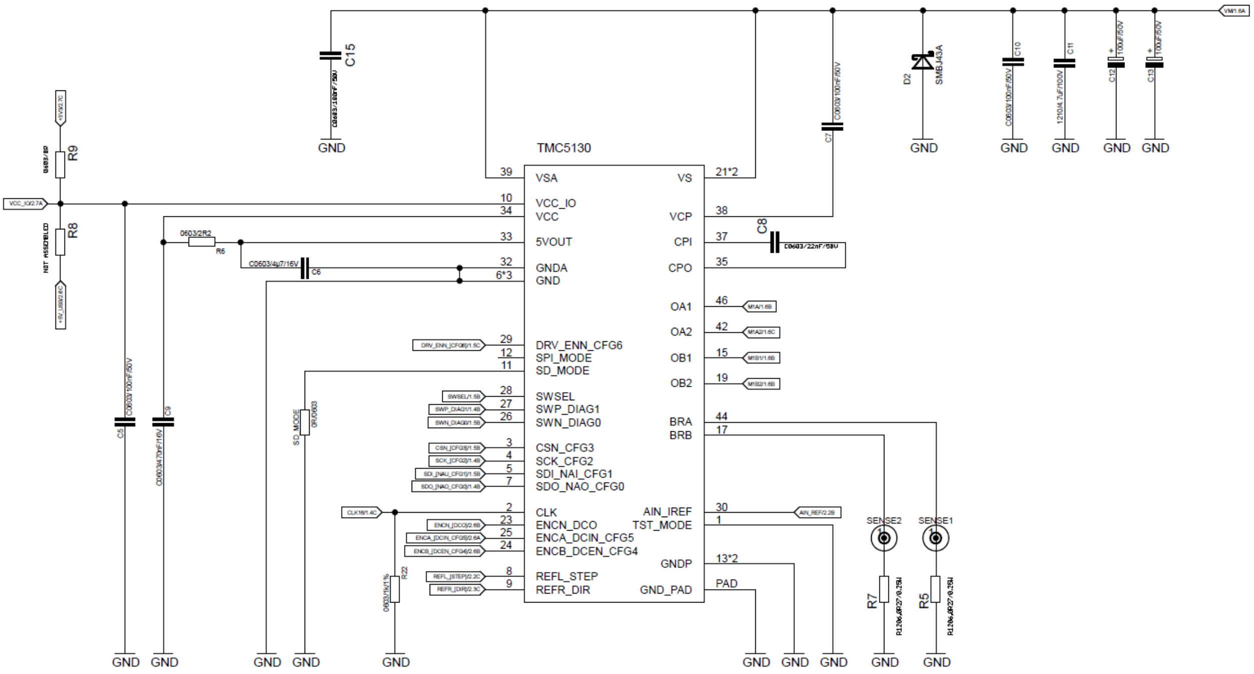
The 5VOUT output voltage ceramic filtering capacitor (4.7 µF recommended) should be placed as close as possible to the 5VOUT pin, with its GND return going directly to the GNDA pin. This ground connection shall not be shared with other loads or additional vias to the GND plan. Use as short and as thick 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 charge pump and 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. The motor supply pins VS should be decoupled with an electrolytic capacitor (47 µF or larger is recommended) and a ceramic capacitor, placed close to the device.

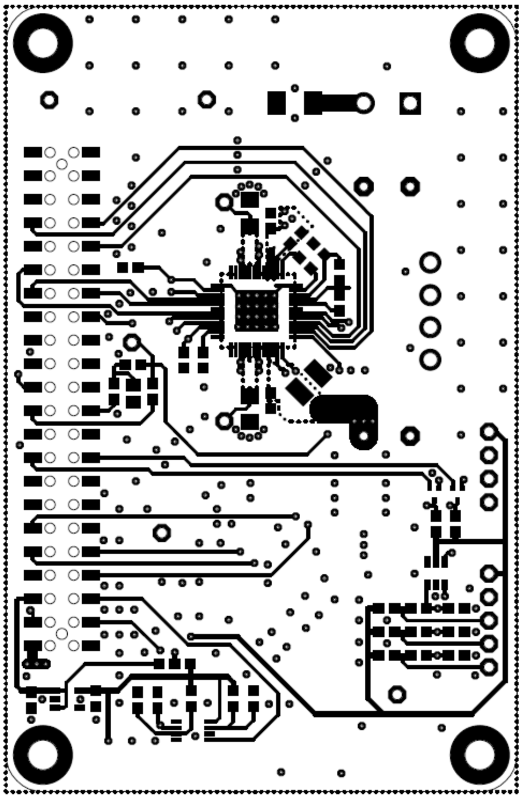
Take into account that the switching motor coil outputs have a high dV/dt. Thus capacitive stray into high resistive signals can occur, if the motor traces are near other traces over longer distances.

31.4 Layout Example

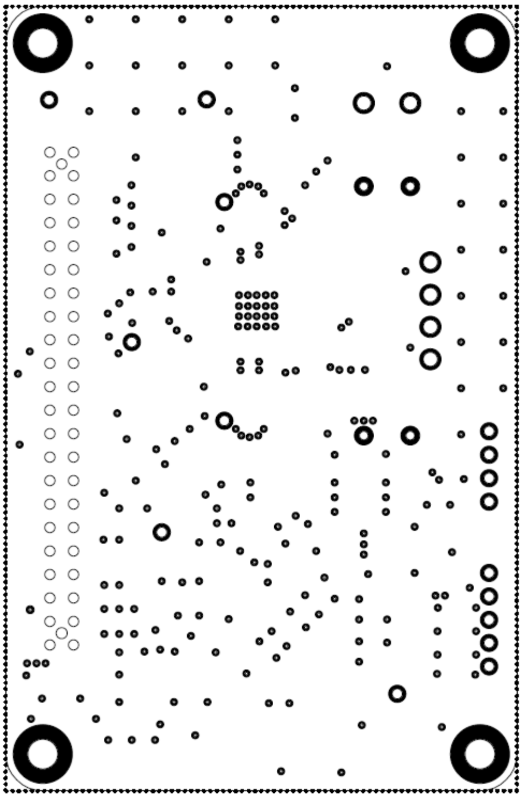
Schematic



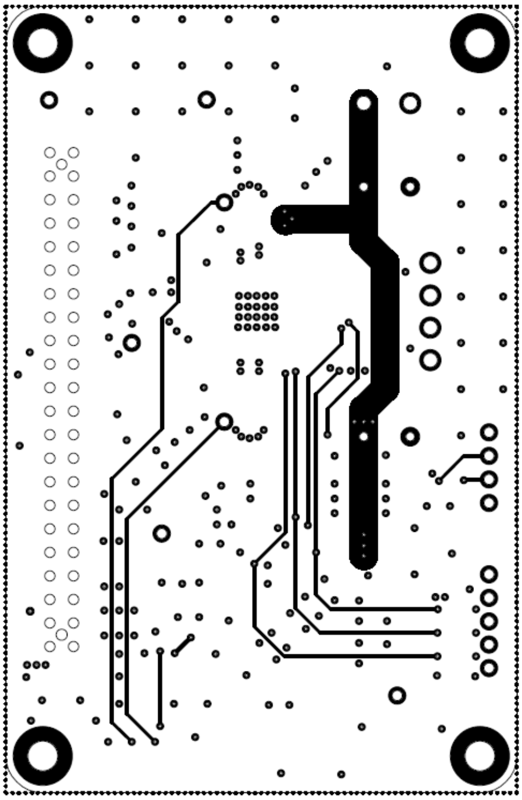
1- Top Layer (assembly side)



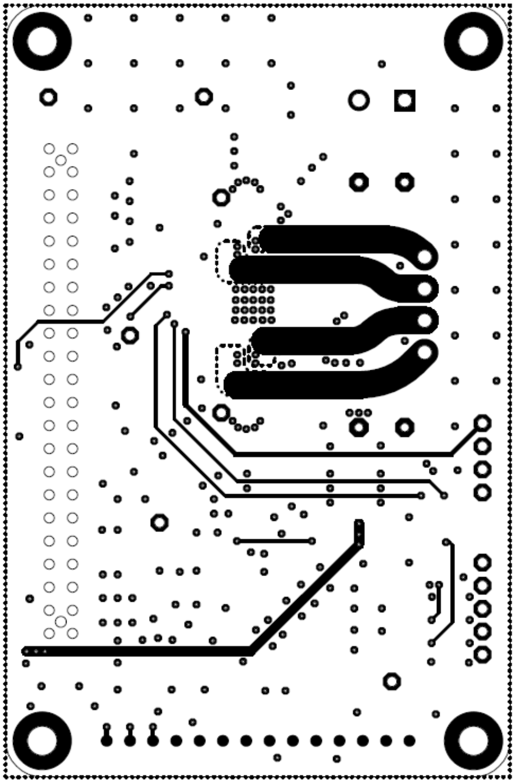
2- Inner Layer (GND)



3- Inner Layer (supply VS)



4- Bottom Layer



Components

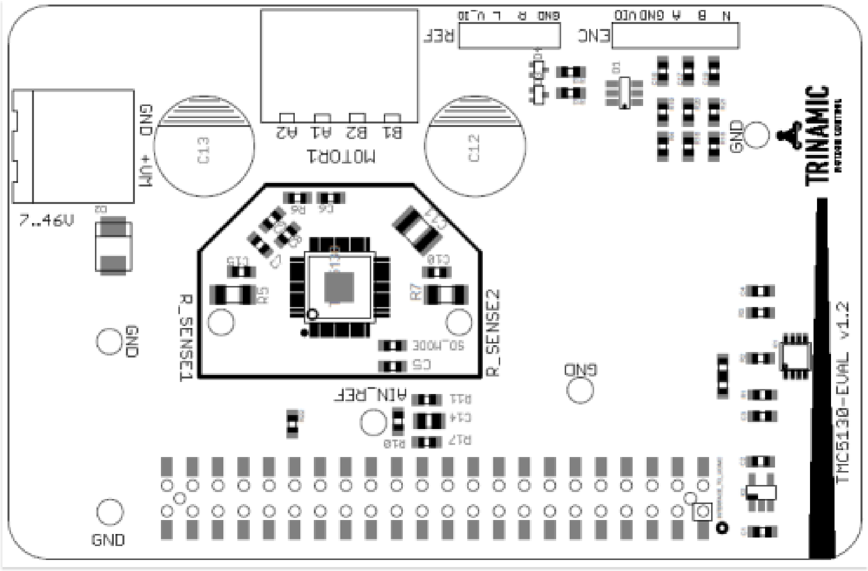


Figure 31.1 Layout example

32 Package Mechanical Data

32.1 Dimensional Drawings TQFP48-EP

Attention: Drawings not to scale.

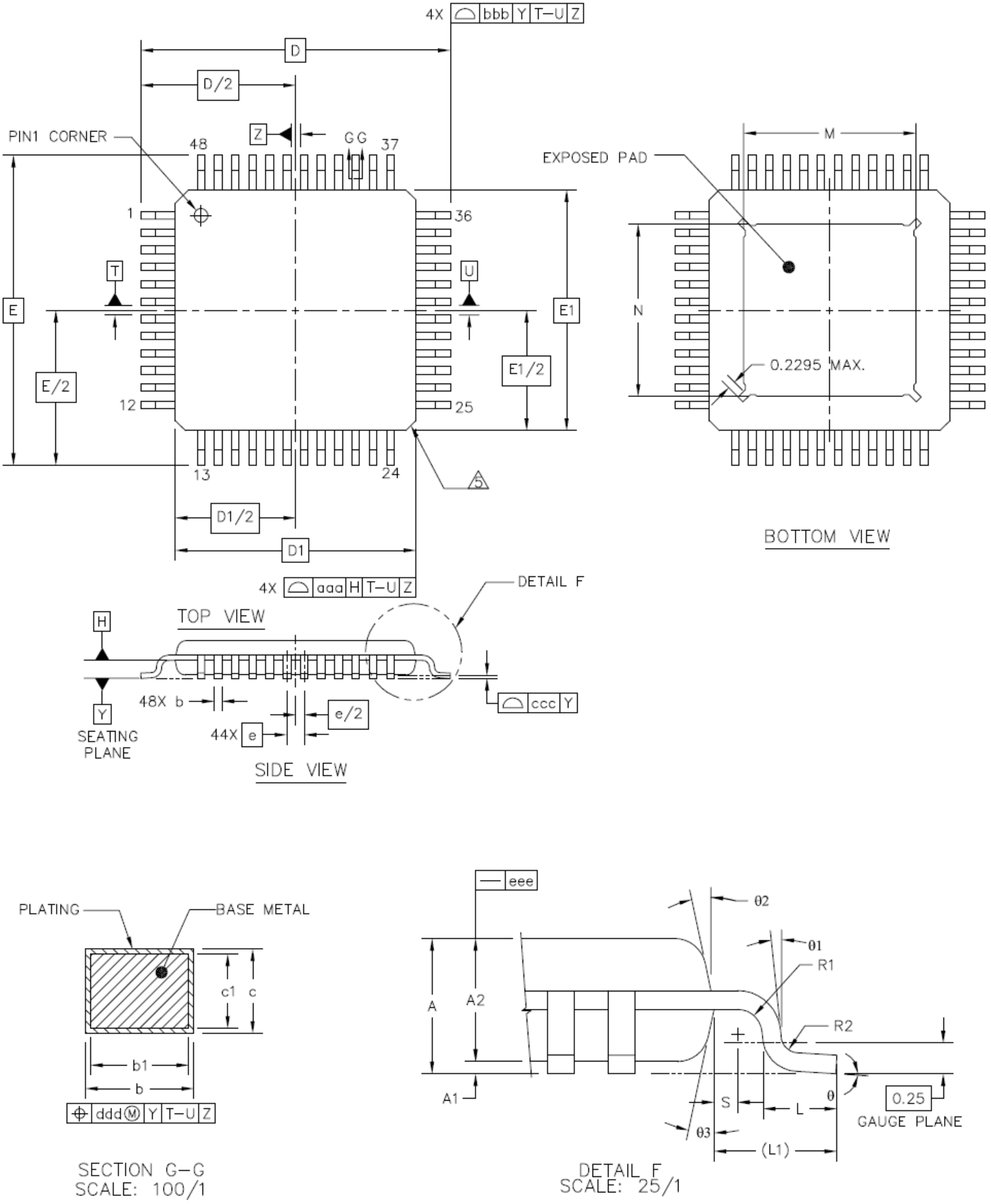


Figure 32.1 Dimensional drawings TQFP48-EP

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

32.2 Package Codes

Type	Package	Temperature range	Code & marking
TMC5130A-TA	TQFP-EP48 (RoHS)	-40°C ... +125°C	TMC5130A-TA

33 Design Philosophy

We feel that this is one of the coolest chips which we did within the last years. The TMC50XX and TMC5130 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 mm^2 of layout using one of the most modern smart power processes. The IC comprises 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

34 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.

35 ESD Sensitive Device

The TMC5130A 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.

36 Table of Figures

37 Revision History

Version	**Date**	**Author** **BD= Bernhard Dwersteg** **SD= Sonja Dwersteg**	**Description**
V014	2013-JUN-18	BD	Added preliminary electrical data, Corrected TEST output pin Added description of microstep sequence in lower resolutions.
V025	2014-JAN-31	BD	Adaptation to pre-series silicon
V038	2014-MAR-25	BD	Value corrections in conjunction with test program Clock frequency range, Conditions for limiting values Limits for deviation of motor current Added hint for highest motor velocities
V0.42	2014-JUN-26	SD	Front page and page 2 new. Changes related to the design. Few chapters reorganized.
V0.44	2014-JUL-15	SD	Principles of operation updated. SPI datagram structure updated.
V0.46	2014-SEP-11	SD	Product name changed.
V1.00	2014-OCT-13	SD	Full version for release, corrected typos, etc.
V1.01	2014-NOV-03	BD	Corrected sense resistor table current values
V1.02	2014-DEC-01	BD	Wording thermal shutdown, encoder IF, hints for mode switching in chapter 1. Added text in 14.4 and 14.5.
V1.03	2014-DEC-05	BD	stallGuard Stop details: Improved homing algorithm in 14.4, Added 15.4, Text for event_stop_sg, improved 19.4
V1.04	2014-DEC-11	BD	Pin table formatting, some comments, CLK info in emergency stop chapter, numbering for homing procedure, comment in 15.4 for event_stop_sg
V1.05	2015-JAN-19	BD	Added design Philosophy, added References, Minor wording corrections, Example with stealthChop
V1.06	2015-FEB-12	BD	Added chapter Closing the Loop. Added UART interface errata.
V1.08	2015-FEB-24	BD	Improved AN links, dcStep description & flowchart, blue blocks
V1.09	2015-MAR-10	BD	Added fSTEP in 14.1, renamed register *TZEROCROSS* to *TZEROWAIT* and register *TZEROWAIT* to *TPOWERDOWN* for consistency.
V1.10	2015-APR-21	BD	More details on DC motor operation, shifted chapter 7.3.1 to 7.2.2
V1.11	2015-OCT-08	BD	Some Typos (*RAMP_STAT*, *position_reached*, *sfilt*); added TCLK spec for first clock event, 20.2 swapped X1 and X3, corr. example in 4.1.1, SPI mode 3 hint, *TOFF* calculation 8.1, fCLK measurement for S/D, GSTAT explanations added

Table 37.1 Document Revisions

38 References

[TMC5130-EVAL] TMC5130-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 TMC5130_TMC2130_TMC2100_Calculations.xlsx