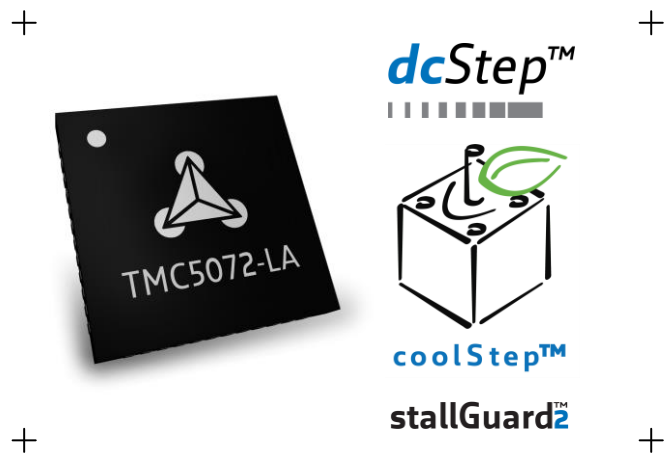


TMC5072 DATASHEET

Dual controller/driver for up to two 2-phase bipolar stepper motors. No-noise stepper operation. Integrated motion controller and encoder counter. SPI, UART (single wire) and Step/Dir.



- APPLICATIONS**
- CCTV, Security
 - Office Automation
 - Antenna Positioning
 - Heliostat Controller
 - Battery powered applications
 - ATM, Cash recycler, POS
 - Lab Automation
 - Liquid Handling
 - Medical
 - Printer and Scanner
 - Pumps and Valves

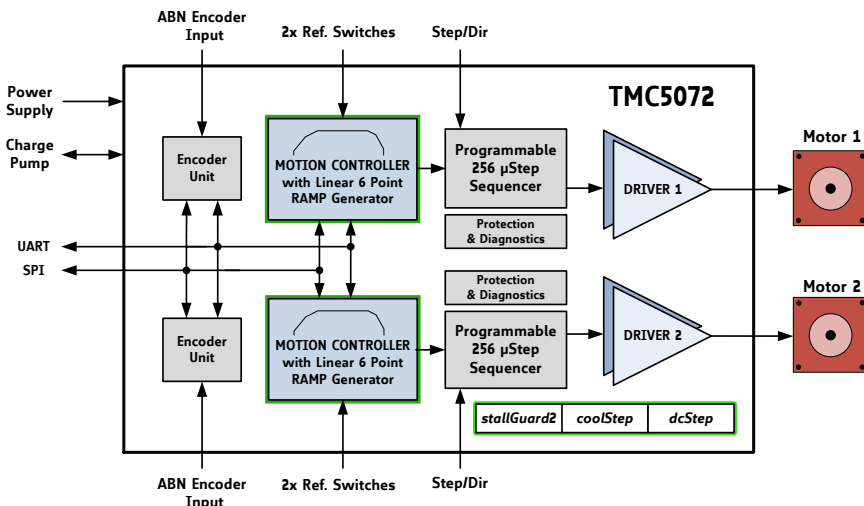
FEATURES AND BENEFITS

- Two 2-phase** stepper motors
- Drive Capability** up to 2x 1.1A coil current (2x 1.5A peak)
- Parallel Option** for one motor at 2.2A (3A peak)
- Motion Controller** with sixPoint™ ramp
- Voltage Range** 4.75... 26V DC
- SPI & Single Wire UART**
- Dual Encoder Interface** and **2x Ref.-Switch input** per axis
- Highest Resolution** up to 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%
- Passive Breaking** and freewheeling mode
- Full Protection & Diagnostics**
- Compact Size** 7x7mm² QFN48 package

DESCRIPTION

The TMC5072 is a dual high performance stepper motor controller and driver IC with serial communication interfaces. It combines flexible ramp generators for automatic target positioning with industries' most advanced stepper motor drivers. 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.

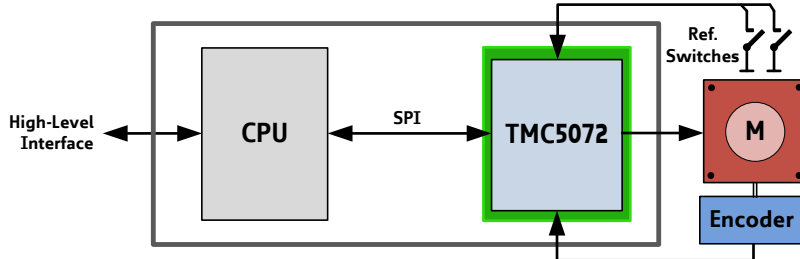
BLOCK DIAGRAM



APPLICATION EXAMPLES: HIGH FLEXIBILITY – MULTIPURPOSE USE

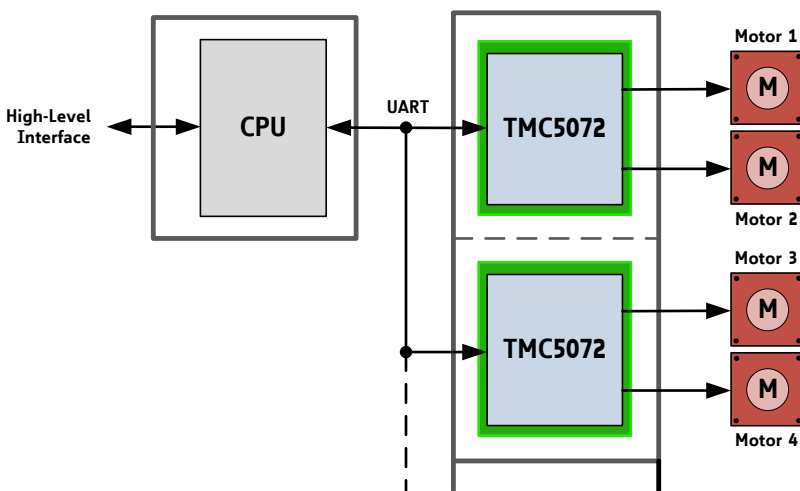
The TMC5072 scores with power density, complete motion controlling features and integrated power stages. It offers a versatility that covers a wide spectrum of applications from battery systems up to embedded applications with 1.5A motor current per coil. The small form factor keeps costs down and allows 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



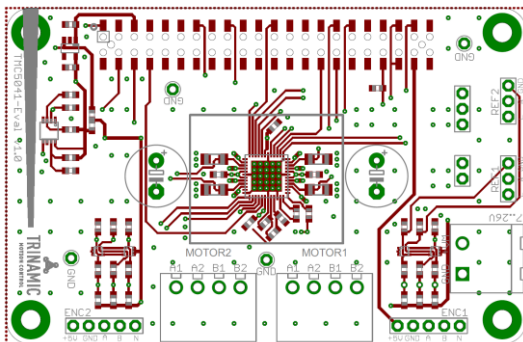
The stepper motor driver outputs are switched in parallel. A dual ABN encoder interface and two reference switch inputs are used.

COMPACT DESIGN FOR UP TO 510 STEPPER MOTORS



Up to 255 TMC5072 can be addressed.

An application for up to 510 stepper motors is shown. The UART single wire differential interface allows for a decentralized distributed system with a minimized number of components. Additionally, an ABN encoder and up to two reference switches can be used for each motor. A single CPU can control the whole system. The CPU-board and controller / driver boards are highly economical and space saving.



TMC5072-EVAL EVALUATION BOARD EVALUATION & DEVELOPMENT PLATFORM

The TMC5072-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 TMC5072 evaluation board system consists of three parts: STARTRAMPE (base board), ESELSBRÜCKE (connector board including several test points), and TMC5072-EVAL.

ORDER CODES

Order code	Description	Size [mm ²]
TMC5072-LA	Dual axis stealthChop controller/driver, QFN-48	7 x 7
TMC5072-EVAL	Evaluation board for TMC5072	85 x 55
STARTRAMPE	Baseboard for TMC5072-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	5	9	SPREADCYCLE AND CLASSIC CHOPPER	57
1.1	KEY CONCEPTS	5	9.1	SPREADCYCLE CHOPPER	58
1.2	CONTROL INTERFACES	6	9.2	CLASSIC CONSTANT OFF TIME CHOPPER	61
1.3	SOFTWARE	6	9.3	RANDOM OFF TIME	62
1.4	MOVING AND CONTROLLING THE MOTOR	7	10	DRIVER DIAGNOSTIC FLAGS	63
1.5	STEALTHCHOP DRIVER WITH PROGRAMMABLE MICROSTEPPING WAVE	7	10.1	TEMPERATURE MEASUREMENT	63
1.6	STALLGUARD2 – MECHANICAL LOAD SENSING	7	10.2	SHORT TO GND PROTECTION	63
1.7	COOLSTEP – LOAD ADAPTIVE CURRENT CONTROL	8	10.3	OPEN LOAD DIAGNOSTICS	63
1.8	DCSTEP – LOAD DEPENDENT SPEED CONTROL	8	11	RAMP GENERATOR	64
1.9	ENCODER INTERFACES	8	11.1	REAL WORLD UNIT CONVERSION	64
2	PIN ASSIGNMENTS	9	11.2	MOTION PROFILES	65
2.1	PACKAGE OUTLINE	9	11.3	INTERRUPT HANDLING	67
2.2	SIGNAL DESCRIPTIONS	9	11.4	VELOCITY THRESHOLDS	67
3	SAMPLE CIRCUITS	12	11.5	REFERENCE SWITCHES	68
3.1	STANDARD APPLICATION CIRCUIT	12	12	STALLGUARD2 LOAD MEASUREMENT	70
3.2	5 V ONLY SUPPLY	13	12.1	TUNING STALLGUARD2 THRESHOLD SGT	71
3.3	ONE MOTOR WITH HIGH CURRENT	14	12.2	STALLGUARD2 UPDATE RATE AND FILTER	73
3.4	EXTERNAL 5V POWER SUPPLY	14	12.3	DETECTING A MOTOR STALL	73
3.5	OPTIMIZING ANALOG PRECISION	16	12.4	HOMING WITH STALLGUARD	73
3.6	DRIVER PROTECTION AND EME CIRCUITRY	16	12.5	LIMITS OF STALLGUARD2 OPERATION	73
4	SPI INTERFACE	18	13	COOLSTEP OPERATION	74
4.1	SPI DATAGRAM STRUCTURE	18	13.1	USER BENEFITS	74
4.2	SPI SIGNALS	19	13.2	SETTING UP FOR COOLSTEP	74
4.3	TIMING	20	13.3	TUNING COOLSTEP	76
5	UART SINGLE WIRE INTERFACE	21	14	DCSTEP	77
5.1	DATAGRAM STRUCTURE	21	14.1	USER BENEFITS	77
5.2	CRC CALCULATION	23	14.2	DESIGNING-IN DCSTEP	77
5.3	UART SIGNALS	23	14.3	ENABLING DCSTEP	78
5.4	ADDRESSING MULTIPLE SLAVES	24	14.4	STALL DETECTION IN DCSTEP MODE	78
5.5	RING MODE	26	14.5	MEASURING ACTUAL MOTOR VELOCITY IN DCSTEP OPERATION	79
6	REGISTER MAPPING	27	15	SINE-WAVE LOOK-UP TABLE	80
6.1	GENERAL CONFIGURATION REGISTERS	28	15.1	USER BENEFITS	80
6.2	RAMP GENERATOR REGISTERS	31	15.2	MICROSTEP TABLE	80
6.3	ENCODER REGISTERS	37	16	STEP/DIR INTERFACE	82
6.4	MICROSTEP TABLE REGISTERS	39	16.1	TIMING	82
6.5	MOTOR DRIVER REGISTERS	41	16.2	CHANGING RESOLUTION	83
6.6	VOLTAGE PWM MODE STEALTHCHOP	46	16.3	MICROPLYER STEP INTERPOLATOR AND STAND STILL DETECTION	83
7	CURRENT SETTING	47	17	ABN INCREMENTAL ENCODER INTERFACE	85
7.1	SENSE RESISTORS	48	17.1	ENCODER TIMING	86
8	STEALTHCHOP™	49	17.2	SETTING THE ENCODER TO MATCH MOTOR RESOLUTION	86
8.1	TWO MODES FOR CURRENT REGULATION	49	17.3	CLOSING THE LOOP	86
8.2	AUTOMATIC SCALING	50	18	QUICK CONFIGURATION GUIDE	88
8.3	FIXED SCALING	52	19	GETTING STARTED	93
8.4	COMBINING STEALTHCHOP WITH OTHER CHOPPER MODES	54			
8.5	FLAGS IN STEALTHCHOP	55			
8.6	FREEWHEELING AND PASSIVE MOTOR BRAKING	56			

19.1	INITIALIZATION EXAMPLES	93	24.3	SUPPLY FILTERING	102
20	EXTERNAL RESET	95	24.4	SINGLE DRIVER CONNECTION	102
21	CLOCK OSCILLATOR AND CLOCK INPUT	95	24.5	LAYOUT EXAMPLE	103
21.1	USING THE INTERNAL CLOCK	95	25	PACKAGE MECHANICAL DATA	104
21.2	USING AN EXTERNAL CLOCK	95	25.1	DIMENSIONAL DRAWINGS	104
21.3	CONSIDERATIONS ON THE FREQUENCY	96	25.2	PACKAGE CODES	104
22	ABSOLUTE MAXIMUM RATINGS	97	26	DESIGN PHILOSOPHY	105
23	ELECTRICAL CHARACTERISTICS	97	27	DISCLAIMER	105
23.1	OPERATIONAL RANGE	97	28	ESD SENSITIVE DEVICE	105
23.2	DC CHARACTERISTICS AND TIMING CHARACTERISTICS	98	29	TABLE OF FIGURES	106
23.3	THERMAL CHARACTERISTICS	101	30	REVISION HISTORY	107
24	LAYOUT CONSIDERATIONS	102	31	REFERENCES	107
24.1	EXPOSED DIE PAD	102			
24.2	WIRING GND	102			

1 Principles of Operation

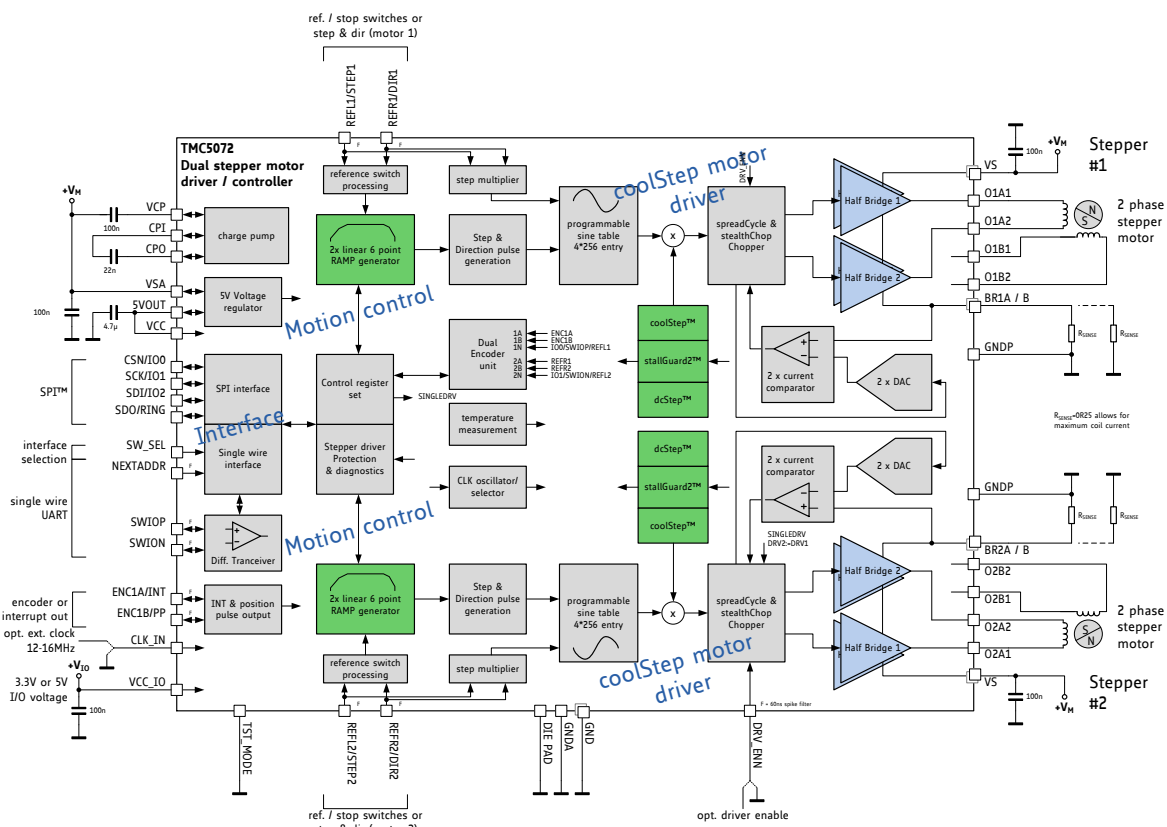


Figure 1.1 Basic application and block diagram

The TMC5072 motion controller and driver chip is an intelligent power component interfacing between the CPU and one or two stepper motors. All stepper motor logic is completely within the TMC5072. No software is required to control the motor – just provide target positions. The TMC5072 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 TMC5072 uses stealthChop, dcStep, coolStep, and stallGuard2 automatically to optimize every motor movement. The clear concept and the comprehensive solution save design time.

1.1 Key Concepts

The TMC5072 implements several 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.
- 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%.
- spreadCycle™** High-precision chopper algorithm available as an alternative to the traditional constant off-time algorithm.
- sixPoint™** Fast and precise positioning using a hardware ramp generator with a set of four acceleration / deceleration settings. Quickest response due to dedicated hardware.

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 TMC5072 supports both, an SPI 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 TMC5072 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. An optional ring mode allows chaining of slaves to optimize interfacing for applications with regularly distributed drives.

1.3 Software

From a software point of view the TMC5072 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

One or both motors can optionally be controlled by a step and direction input. In this case, the respective 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 1.2 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.

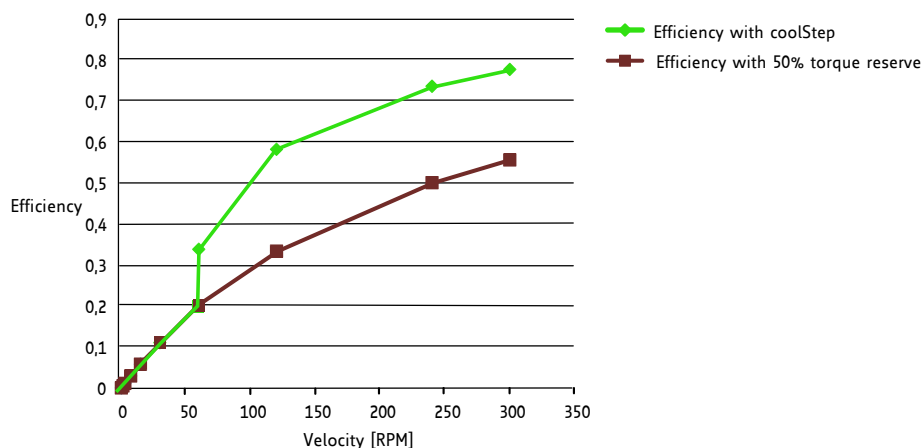


Figure 1.2 Energy efficiency with coolStep (example)

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 lose 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 Interfaces

The TMC5072 provides two encoder interfaces for external incremental encoders. The encoders can be used for homing of the motion controllers (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. 32 bit encoder counters are provided.

2 Pin Assignments

2.1 Package Outline

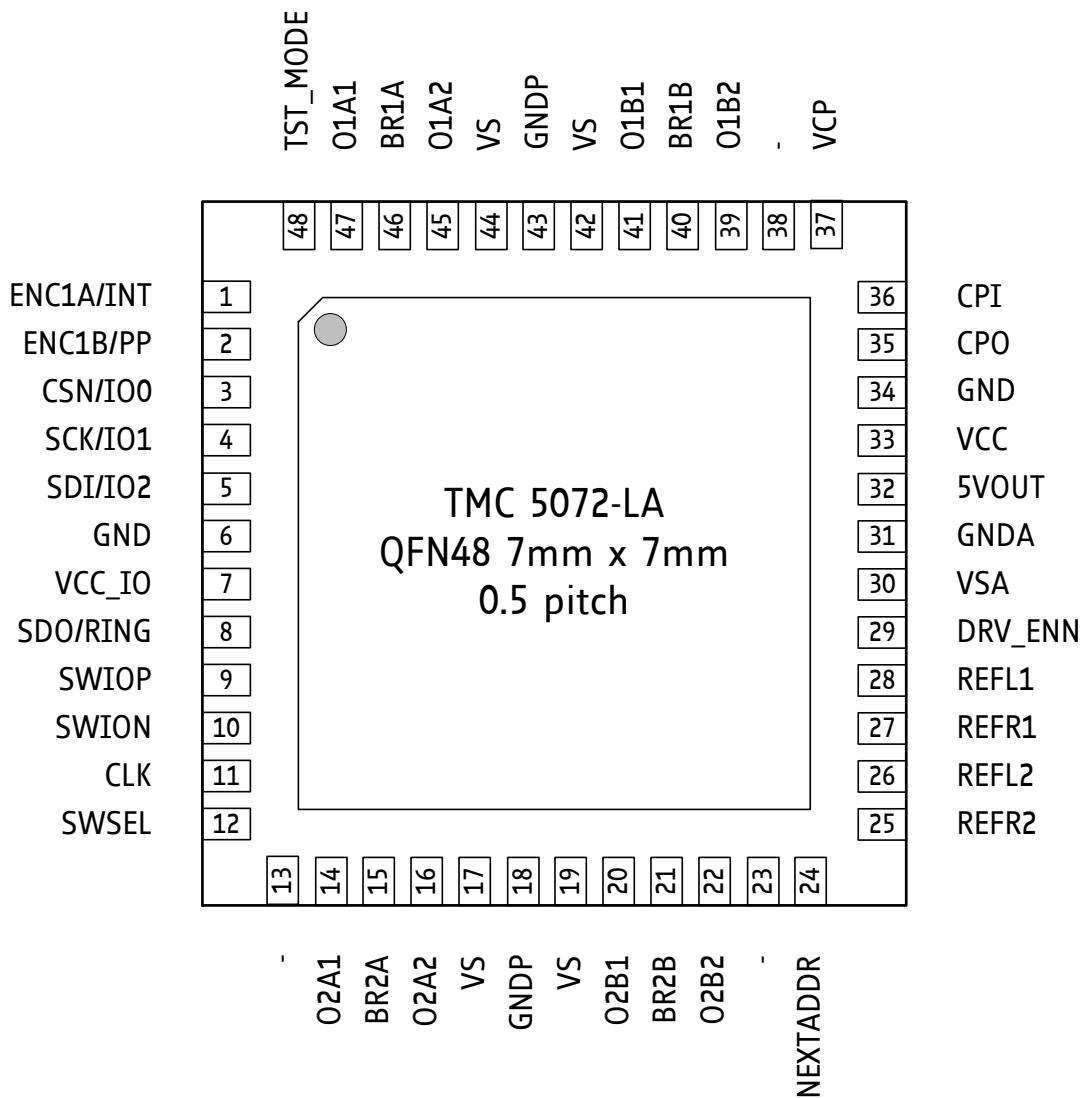


Figure 2.1 TMC5072 pin assignments.

2.2 Signal Descriptions

Pin	Number	Type	Function
GND	6, 34	GND	Digital ground pin for IO pins and digital circuitry.
VCC_IO	7		3.3V or 5V I/O supply voltage pin for all digital pins.
VSA	30		Analog supply voltage for 5V regulator – typically supplied with driver supply voltage. An additional 100nF capacitor to GND (GND plane) is recommended for best performance.
GNDA	31	GND	Analog GND. Tie to GND plane.
5VOUT	32		Output of internal 5V regulator. Attach 2.2 μ F or larger ceramic capacitor to GNDA near to pin for best performance. May be used to supply VCC of chip.

Pin	Number	Type	Function
VCC	33		5V supply input for digital circuitry within chip and charge pump. Attach 470nF capacitor to GND (GND plane). May be supplied by 5VOUT. A 2.2Ω 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!
DIE_PAD	-	GND	Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane.

Table 2.1 Low voltage digital and analog power supply pins

Pin	Number	Type	Function
CPO	35	O(VCC)	Charge pump driver output. Outputs 5V (GND to VCC) square wave with 1/16 of internal oscillator frequency.
CPI	36	I(VCP)	Charge pump capacitor input: Provide external 22nF or 33nF / 50V capacitor to CPO.
VCP	37		Output of charge pump. Provide external 100nF capacitor to VS.

Table 2.2 Charge pump pins

Pin	Number	Type	Function
ENC1A/INT	1	I/O	Input A for incremental encoder 1. Can be programmed to provide interrupt output based on ramp generator flags <i>RAMP_STAT</i> bits 4, 5, 6 & 7 and encoder null event status <i>ENC_STATUS</i> bit 0 (<i>poscmp_enable=1</i>).
ENC1B/PP	2	I/O	Input B for incremental encoder 1. Can be programmed to provide position compare output for motor 1 (<i>poscmp_enable=1</i>).
CSN/IO0	3	I/O	Chip select input of SPI interface, programmable IO in UART mode
SCK/IO1	4	I/O	Serial clock input of SPI interface, programmable IO in UART mode
SDI/IO2	5	I/O	Data input of SPI interface, programmable IO in UART mode
SDO/RING	8	I/O	Data output of SPI interface (Tristate, enabled with CSN=0), mode configuration input in UART mode (0 = Normal mode, 1 = Single wire ring mode – SWIO_P is input, SWIO_N is output)
SWIOP	9	I/O	Single wire I/O (positive). Serial input in ring mode. Multi-purpose input in SPI mode or encoder 1 N input.
SWION	10	I/O	Single wire I/O (negative) for differential mode. Leave open in non-differential mode when operating at 5V IO voltage or tie to desired threshold voltage. Serial output in ring mode. Multi-purpose input in SPI mode or encoder 2 N input.
CLK	11	I	Clock input. Tie to GND using short wire for internal clock or supply external clock. The first high signal disables the internal oscillator until power down.
SWSEL	12	I	Interface selection input. Tie to GND for SPI mode, tie to VCC_IO for single wire (UART) interface mode.
NEXTADDR	24	I	Address increment (if tied high) for single wire (UART) mode. General purpose input in SPI mode
REFR2/DIR2	25	I	Right reference switch input for motor 2, optional DIR input for STEP/DIR operation of motor 2 or encoder 2 B input
REFL2/STEP2	26	I	Left reference switch input for motor 2, optional STEP input for STEP/DIR operation of motor 2
REFR1/DIR1	27	I	Right reference switch input for motor 1, optional DIR input for STEP/DIR operation of motor 1 or encoder 2 A input
REFL1/STEP1	28	I	Left reference switch input for motor 1, optional STEP input for STEP/DIR operation of motor 1
DRV_ENN	29	I	Enable input for motor drivers. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level. Tie to GND for normal operation.
TST_MODE	48	I	Test mode input. Tie to GND using short wire.

Pin	Number	Type	Function
-	13, 23, 38	N.C.	Unused pins – no internal electrical connection. Leave open or tie to GND for compatibility with future devices.

Table 2.3 Digital I/O pins (all related to VCC_IO supply)

Pin	Number	Type	Function
O2A1	14	O (VS)	Motor 2 coil A output 1
BR2A	15		Sense resistor connection for motor 2 coil A. Place sense resistor to GND near pin.
O2A2	16	O (VS)	Motor 2 coil A output 2
VS	17, 19		Motor supply voltage. Provide filtering capacity near pin with shortest loop to nearest GNDP pin (respectively via GND plane).
GNDP	18	GND	Power GND. Connect to GND plane near pin.
O2B1	20	O (VS)	Motor 2 coil B output 1
BR2B	21		Sense resistor connection for motor 2 coil B. Place sense resistor to GND near pin.
O2B2	22	O (VS)	Motor 2 coil B output 2
O1B2	39	O (VS)	Motor 1 coil B output 2
BR1B	40		Sense resistor connection for motor 1 coil B. Place sense resistor to GND near pin.
O1B1	41	O (VS)	Motor 1 coil B output 1
VS	42, 44		Motor supply voltage. Provide filtering capacity near pin with shortest loop to nearest GNDP pin (respectively via GND plane).
GNDP	43	GND	Power GND. Connect to GND plane near pin.
O1A2	45	O (VS)	Motor 1 coil A output 2
BR1A	46		Sense resistor connection for motor 1 coil A. Place sense resistor to GND near pin.
O1A1	47	O (VS)	Motor 1 coil A output 1

Table 2.4 Power driver pins

3 Sample Circuits

The sample circuits show the connection of the 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

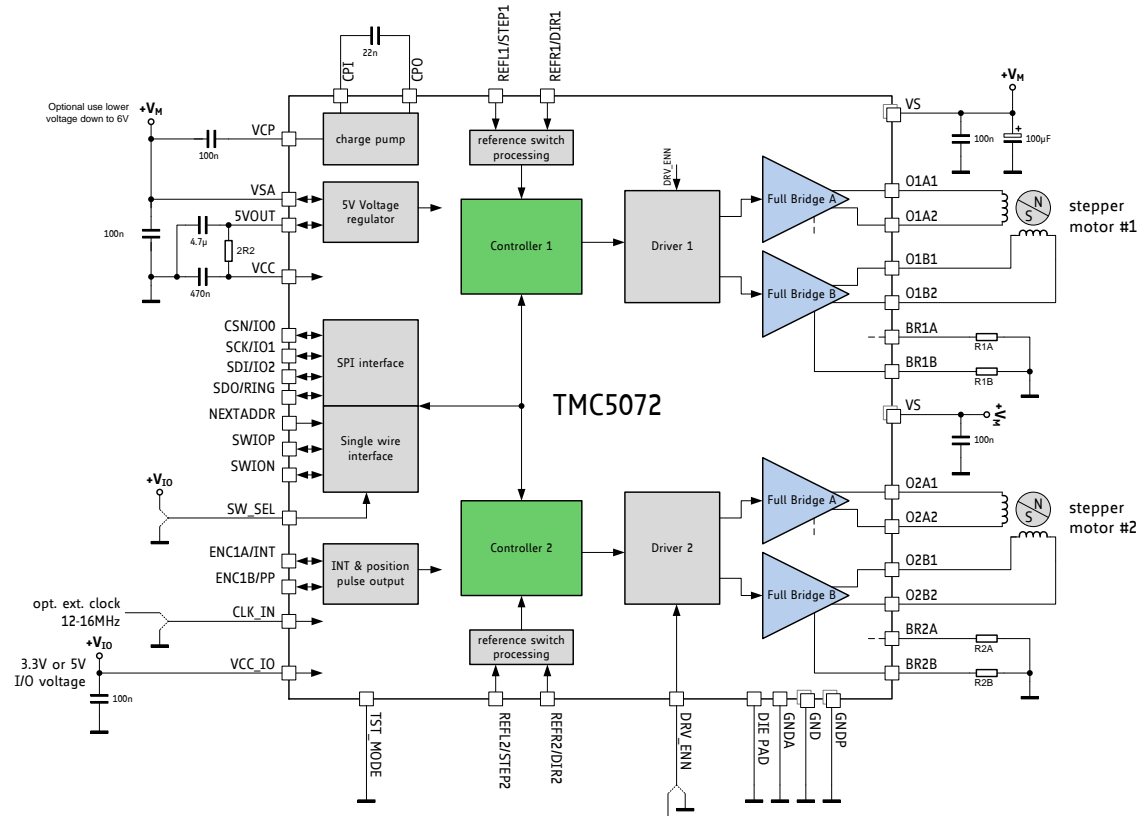


Figure 3.1 Standard application circuit

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. Using the 12V supply for VSA will reduce the power dissipation of the internal 5V regulator to about 37% of the dissipation caused by supply with the full motor voltage. For best motor chopper performance, an optional R/C-filter de-couples 5VOUT from digital noise cause by power drawn from VCC.

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 GND_A 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 5 V Only Supply

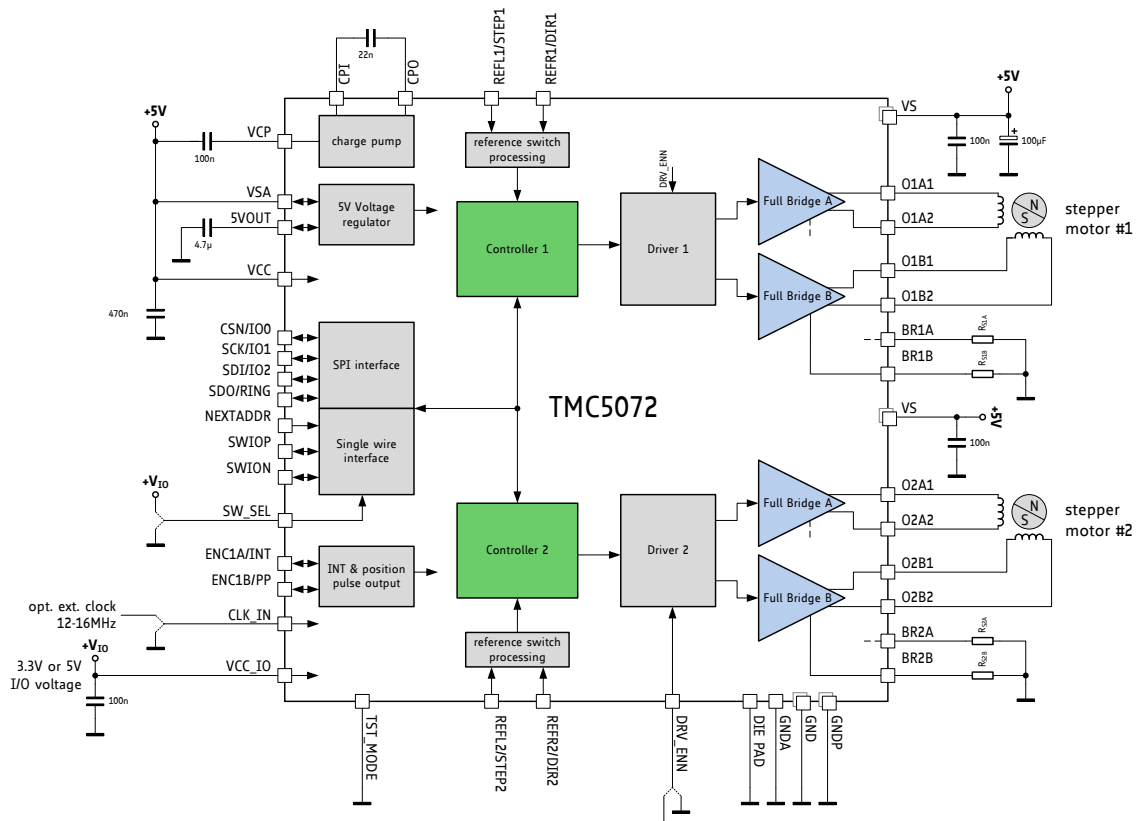


Figure 3.2 5V only operation

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 +/-5% 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.

3.3 One Motor with High Current

The TMC5072 supports double motor current for a single driver by paralleling both power stages. In order to operate in this mode, activate the flag *single_driver* in the global configuration register *GCONF*. This register can be locked for subsequent write access.

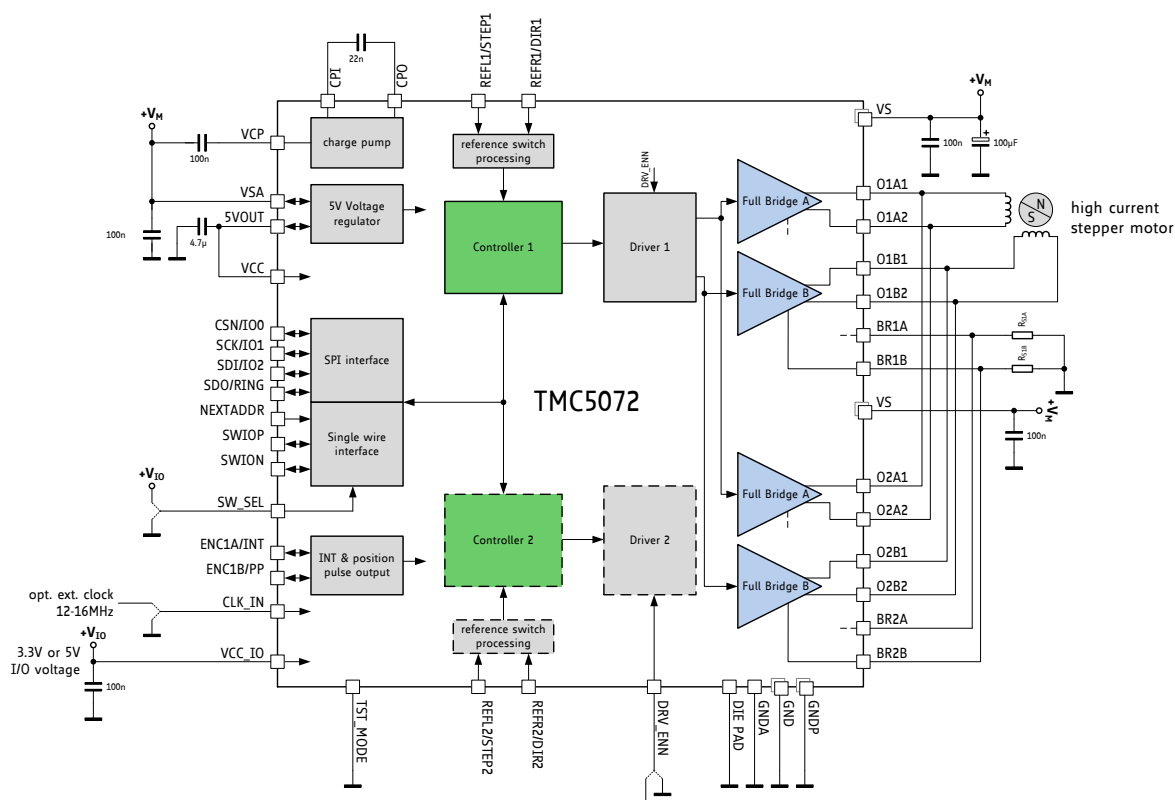


Figure 3.3 Driving a single motor with high current

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.

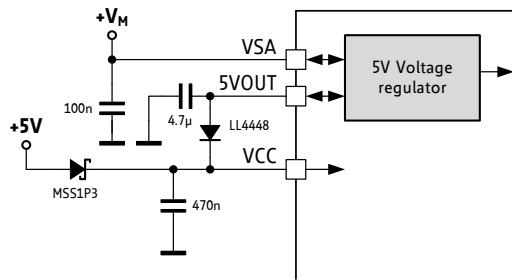
3.4.1 Support for the VCC Supply

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.

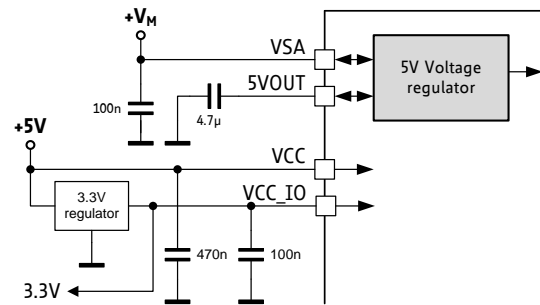
When cutting pin VCC from 5VOUT, make sure that the VCC supply comes up before or synchronously with the 5VOUT supply to ensure a correct power up reset of the internal logic. A simple schematic uses two diodes forming an OR of the internal and the external power supplies for VCC. In order to prevent 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.

In certain setups, switching of VCC voltage can be eliminated. A third variant uses the VCC_IO supply to ensure power-on reset. This is possible, if VCC_IO comes up synchronously with or delayed to VCC. Use a linear regulator to generate a 3.3V VCC_IO from the external 5V VCC source. This 3.3V regulator

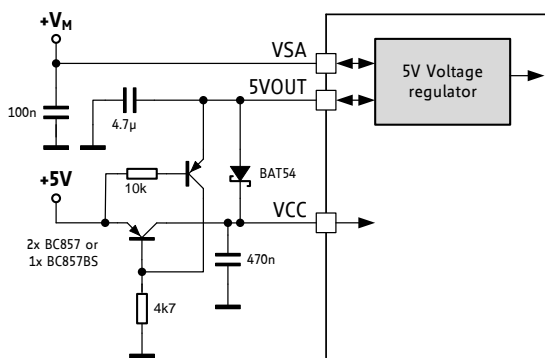
will cause a certain voltage drop. A voltage drop in the regulator of 0.9V or more (e.g. LD1117-3.3) ensures that the 5V supply already has exceeded the lower limit of about 3.0V once the reset conditions ends. The reset condition ends earliest, when VCC_IO exceeds the undervoltage limit of minimum 2.1V. 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.

Well-regulated, stable supply, better than +/-5%

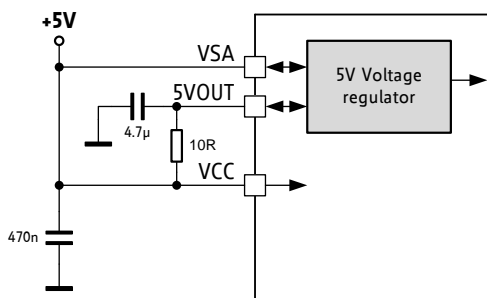


Figure 3.5 Using an external 5V supply to bypass internal regulator

3.5 Optimizing Analog Precision

The 5VOUT pin is used as an analog reference for operation of the TMC5072. Performance will degrade when there is voltage ripple on this pin. Most of the high frequency ripple in a TMC5072 design results from the operation of the internal digital logic. The digital logic switches with each edge of the clock signal. Further, ripple results from operation of the charge pump, which operates with roughly 1MHz and draws current from the VCC pin. In order to keep this ripple as low as possible, an additional filtering capacitor can be put directly next to the VCC pin with vias to the GND plane giving a short connection to the digital GND pins (pin 6 and pin 34). Analog performance is best, when this ripple is kept away from the analog supply pin 5VOUT, using an additional series resistor of 2.2Ω. The voltage drop on this resistor will be roughly 100 mV ($I_{VCC} * R$).

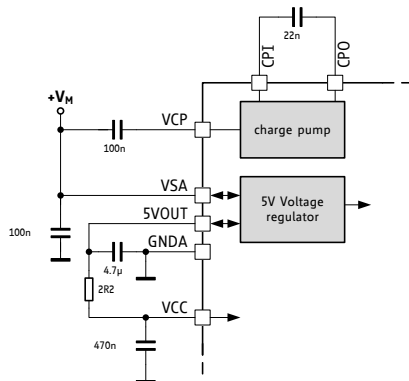


Figure 3.6 RC-Filter on VCC for reduced ripple

3.6 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 in between of the coil terminals eliminate coil overvoltage caused by live plugging. Optionally protect all outputs by a varistor against ESD voltage.

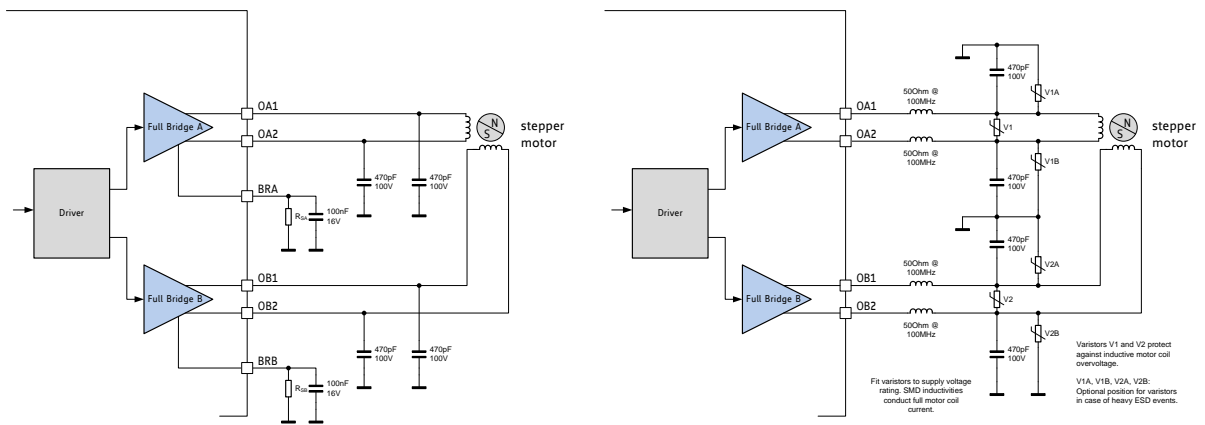


Figure 3.7 Simple ESD enhancement and more elaborate motor output protection

4 SPI Interface

4.1 SPI Datagram Structure

The TMC5072 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 TMC5072 must be handled in a way, that it stays active (low) for the complete duration of the datagram transmission.

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

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

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

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

SPI DATAGRAM STRUCTURE																																																	
MSB (transmitted first)										40 bit										LSB (transmitted last)																													
39 0																													
→ 8 bit address										← → 32 bit data										← 8 bit SPI status																													
39 ... 32										31 ... 0																																							
→ to TMC5072: RW + 7 bit address										8 bit data								8 bit data								8 bit data								8 bit data															
← from TMC5072: 8 bit SPI status																																																	
39 / 38 ... 32										31 ... 24								23 ... 16								15 ... 8								7 ... 0															
w	38...32								31...28				27...24				23...20				19...16				15...12				11...8				7...4				3...0												
3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	9	8	7	6	5	4	3	2	1	0			
9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

4.1.1 Selection of Write / Read (WRITE_notREAD)

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

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

Whenever data is read from or written to the TMC5072, 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 TMC5072	data received from TMC5072
read <i>XACTUAL</i>	→ 0x2100000000	← 0xSS & unused data
read <i>XACTUAL</i>	→ 0x2100000000	← 0xSS & <i>XACTUAL</i>
write <i>VMAX</i> = 0x00ABCDEF	→ 0xA700ABCDEF	← 0xSS & <i>XACTUAL</i>
write <i>VMAX</i> = 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.

<i>SPI_STATUS</i> – status flags transmitted with each SPI access in bits 39 to 32		
Bit	Name	Comment
7	-	reserved (0)
6	<i>status_stop_l(2)</i>	<i>RAMP_STAT2</i> [0] – 1: Signals motor 2 stop left switch status
5	<i>status_stop_l(1)</i>	<i>RAMP_STAT1</i> [0] – 1: Signals motor 1 stop left switch status
4	<i>velocity_reached(2)</i>	<i>RAMP_STAT2</i> [8] – 1: Signals motor 2 has reached its target velocity
3	<i>velocity_reached(1)</i>	<i>RAMP_STAT1</i> [8] – 1: Signals motor 1 has reached its target velocity
2	<i>driver_error(2)</i>	<i>GSTAT</i> [2] – 1: Signals driver 2 driver error (clear by reading <i>GSTAT</i>)
1	<i>driver_error(1)</i>	<i>GSTAT</i> [1] – 1: Signals driver 1 driver error (clear by reading <i>GSTAT</i>)
0	<i>reset_flag</i>	<i>GSTAT</i> [0] – 1: Signals, that a reset has occurred (clear by reading <i>GSTAT</i>)

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 TMC5072 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 TMC5072.

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.

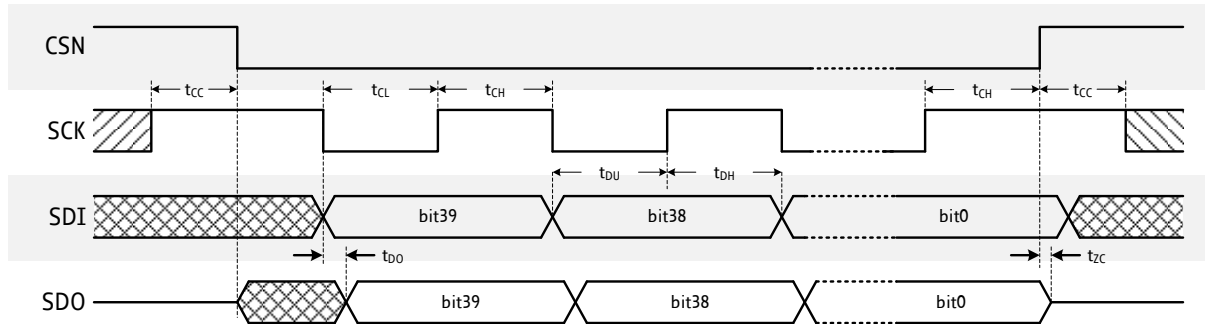


Figure 4.1 SPI timing

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 TMC5072 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 TMC5072 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 TMC5072 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 TMC5072 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. Therefore it is advised to use the latching function instead. 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 TMC5072 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 TMC5072-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 TMC5072 has following signals:

TMC5072 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.
NEXTADDR	Address increment pin for sequential addressing scheme
SDO/RING	A low level on this input selects standard mode, a high level switches to ring mode

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.

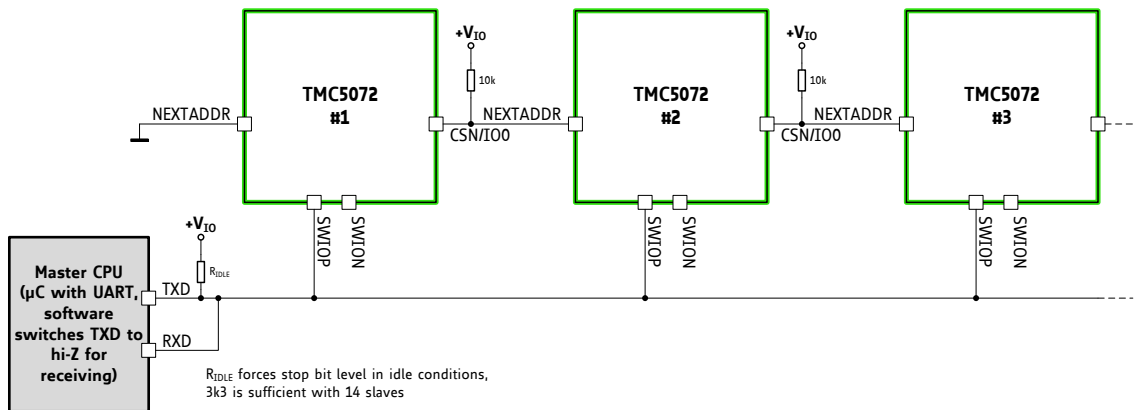
5.4 Addressing Multiple Slaves

ADDRESSING ONE OR TWO SLAVES

If only one or two TMC5072 are addressed by a master using a single UART interface, a hardware address selection can be done by *setting the NEXTADDR pins to different levels*.

ADDRESSING UP TO 255 SLAVES

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



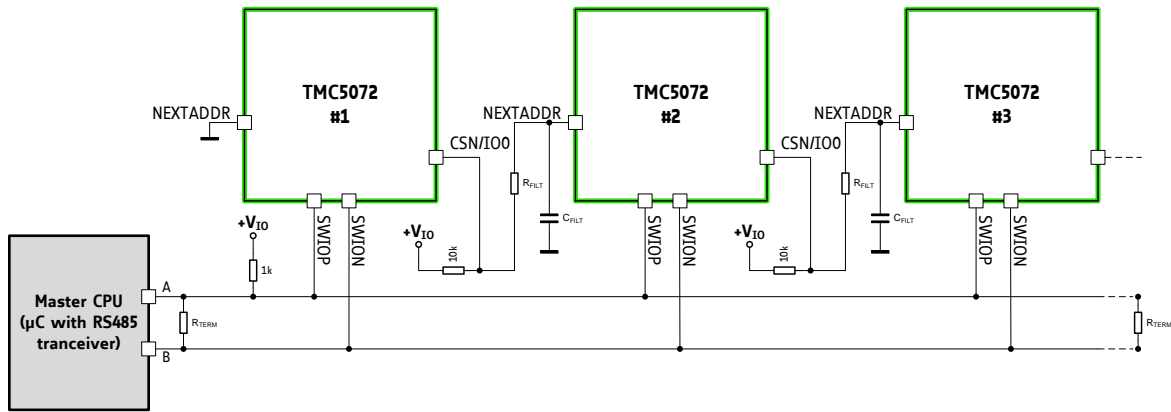
EXAMPLE FOR ADDRESSING UP TO 255 TMC5072

Addressing phase 1:	address 0, I00 is high-Z	address 1	address 1
Addressing phase 2:	program to address 254 & set I00 low	address 0, I00 is high-Z	address 1
Addressing phase 3:	address 254	program to address 253 & set I00 low	address 0
Addressing phase 4:	address 254	address 253	program to address 252 & set I00 low
Addressing phase X:	continue procedure		

Figure 5.1 Addressing multiple TMC5072 via single wire interface using chaining

Proceed as follows:

- Tie the NEXTADDR pin of your first TMC5072 to GND.
- Interconnect one of the general purpose IO-pins of the first TMC5072 to the next drivers NEXTADDR pin using an additional pull-up resistor. 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 general purpose output, which is tied to the next drivers NEXTADDR has to be programmed as output and set to 0.
- 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 TMC5072

Addressing phase 1:	address 0, I00 high	address 1	address 1
Addressing phase 2:	program to address 254 & set I00 low	address 0, I00 high	address 1
Addressing phase 3:	address 254	program to address 253 & set NAO low	address 0, I00 high
Addressing phase 4:	address 254	address 253	program to address 252 & set I00 low
Addressing phase X:	continue procedure		

Figure 5.2 Addressing multiple TMC5072 via differential interface, additional filtering for NEXTADDR

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 TMC5072 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.

5.5 Ring Mode

In a second mode of operation, all slaves are cascaded in a ring. The intention is to allow a simple scheme of addressing without the need for additional NEXTADDR wires. At the same time, the distance between each two chips can be kept short due to the chain structure, and the load on each line is only a single input. Therefore the logical ring is optimum when wiring cost is critical. In case the physical structure more resembles a line rather than a ring, the devices can be cascaded in a way that each second IC is chained in direction left to right and back. This way, the distance from the last slave's data output to the master is no longer than the distance between three slaves.

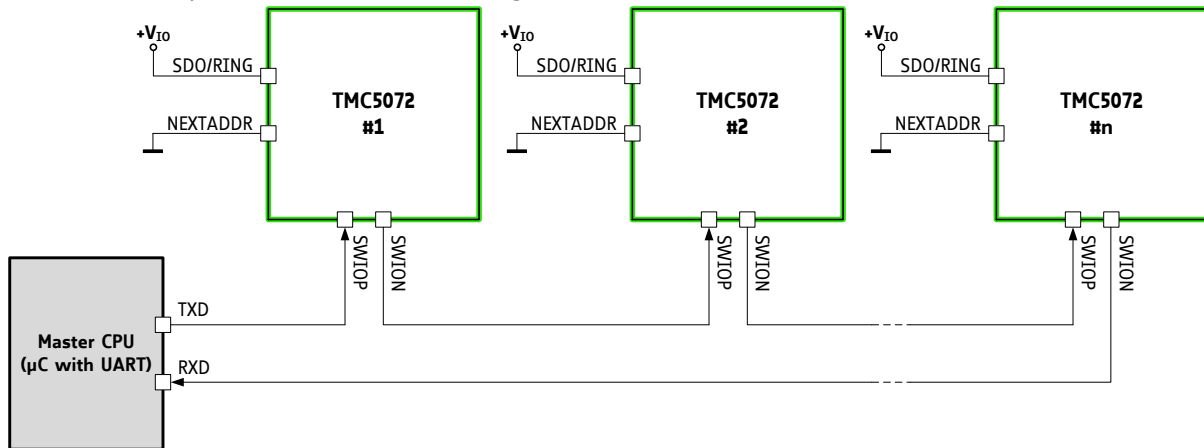


Figure 5.3 Ring Mode Example

The ring mode is enabled by tying SDO/RING high and enabling UART mode. In this mode, SWIO_P is the data input (CMOS level), while SWIO_N is the data output. SWIO_N stays high (idle state) and does forward data coming in via SWIO_P, until SLAVEADDR has been programmed. This way, the addressing phase can be accomplished for a number of cascaded slaves. All data is forwarded from SWIO_P to SWIO_N after the address setting has been programmed to a value different from 0. Having addressed all slaves in the chain allows master requests to go to all slaves, while slave response travels via all devices to the master. The NEXTADDR input is not required for addressing in this mode, unless two rings with a common start point are intended. Please be aware, that data can only pass from any slave through the ring structure back to the master after all slaves have been assigned an address different from 0. Do not assign two identical addresses.

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 <ul style="list-style-type: none"> - global configuration - global status flags - slave address configuration - and I/O configuration
Ramp Generator Motion Control Register Set	This register set offers registers for <ul style="list-style-type: none"> - choosing a ramp mode - choosing velocities - homing - acceleration and deceleration - target positioning
Ramp Generator Driver Feature Control Register Set	This register set offers registers for <ul style="list-style-type: none"> - driver current control - setting thresholds for coolStep operation - setting thresholds for different chopper modes - setting thresholds for dcStep operation - reference switch and stallGuard2 event configuration - a ramp and reference switch status register
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 <ul style="list-style-type: none"> - setting / reading out microstep table and counter - chopper and driver configuration - coolStep and stallGuard2 configuration - dcStep configuration, and - 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	11	GCONF	Bit GCONF – Global configuration flags	
				0	<i>single_driver</i> 0: Two motors can be operated. 1: Single motor, double current operation - driver 2 outputs are identical to driver 1, all driver 2 related controls are unused in this mode. <i>Attention: Set correctly before driver enable!</i>
				1	<i>stepdir1_enable</i> 0: Motor 1 is driven by internal ramp generator 1. 1: External control of motor 1 using STEP1 and DIR1 – ramp generator 1 is not used.
				2	<i>stepdir2_enable</i> 0: Motor 2 is driven by internal ramp generator 2. 1: External control of motor 2 using STEP2 and DIR2 – ramp generator 2 is not used.
				3	<i>poscmp_enable</i> 0: Encoder 1 A and B inputs are mapped. 1: Position compare pulse (PP) and interrupt output (INT) are available, Encoder 1 is unused.
				4	<i>enc1_refsel</i> 0: N channel 1 mapped depending on interface to SWIOP (if SW_SEL=0) or IO0 (if SW_SEL=1). 1: N channel 1 mapped to REFL1.
				5	<i>enc2_enable</i> 0: Right reference switches are available. 1: Encoder 2 A and B signals are mapped to REFR1 and REFR2 inputs.
				6	<i>enc2_refsel</i> 0: N channel 2 mapped depending on interface to SWION (if SW_SEL=0) or IO1 (if SW_SEL=1). 1: N channel 2 mapped to REFL2.
				7	<i>test_mode</i> 0: Normal operation 1: Enable analog test output on pin REFR2 SLAVEADDR selects the function of REFR2: 0...4: T120, DAC1, VDDH1, DAC2, VDDH2 <i>Attention: Not for user, set to 0 for normal operation!</i>
				8	<i>shaft1</i> 1: Inverse motor 1 direction
				9	<i>shaft2</i> 1: Inverse motor 2 direction
				10	<i>lock_gconf</i> 1: GCONF is locked against further write access.
11	<i>dc_sync</i> 1: Synchronizes both motors, when both are operated in dcStep mode. The slower motor will slow down the other motor, too.				

GENERAL CONFIGURATION REGISTERS (0x00...0x0F)																												
R/W	Addr	n	Register	Description / bit names																								
R+C	0x01	4	GSTAT	<table border="1"> <thead> <tr> <th>Bit</th> <th>GSTAT – Global status flags</th> </tr> </thead> <tbody> <tr> <td>0</td> <td> <i>reset</i> 1: Indicates that the IC has been reset since the last read access to GSTAT. All registers have been cleared to reset values. </td> </tr> <tr> <td>1</td> <td> <i>drv_err1</i> 1: Indicates, that driver 1 has been shut down due to overtemperature or short circuit detection since the last read access. Read <i>DRV_STATUS1</i> for details. The flag can only be reset when all error conditions are cleared. </td> </tr> <tr> <td>2</td> <td> <i>drv_err2</i> 1: Indicates, that driver 2 has been shut down due to overtemperature or short circuit detection since the last read access. Read <i>DRV_STATUS2</i> for details. The flag can only be reset when all error conditions are cleared. </td> </tr> <tr> <td>3</td> <td> <i>uv_cp</i> 1: Indicates an undervoltage on the charge pump. The driver is disabled in this case. </td> </tr> </tbody> </table>	Bit	GSTAT – Global status flags	0	<i>reset</i> 1: Indicates that the IC has been reset since the last read access to GSTAT. All registers have been cleared to reset values.	1	<i>drv_err1</i> 1: Indicates, that driver 1 has been shut down due to overtemperature or short circuit detection since the last read access. Read <i>DRV_STATUS1</i> for details. The flag can only be reset when all error conditions are cleared.	2	<i>drv_err2</i> 1: Indicates, that driver 2 has been shut down due to overtemperature or short circuit detection since the last read access. Read <i>DRV_STATUS2</i> for details. The flag can only be reset when all error conditions are cleared.	3	<i>uv_cp</i> 1: Indicates an undervoltage on the charge pump. The driver is disabled in this case.														
				Bit	GSTAT – Global status flags																							
				0	<i>reset</i> 1: Indicates that the IC has been reset since the last read access to GSTAT. All registers have been cleared to reset values.																							
				1	<i>drv_err1</i> 1: Indicates, that driver 1 has been shut down due to overtemperature or short circuit detection since the last read access. Read <i>DRV_STATUS1</i> for details. The flag can only be reset when all error conditions are cleared.																							
2	<i>drv_err2</i> 1: Indicates, that driver 2 has been shut down due to overtemperature or short circuit detection since the last read access. Read <i>DRV_STATUS2</i> for details. The flag can only be reset when all error conditions are cleared.																											
3	<i>uv_cp</i> 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.																								
W	0x03	8 + 4	SLAVECONF	<table border="1"> <thead> <tr> <th>Bit</th> <th>SLAVECONF</th> </tr> </thead> <tbody> <tr> <td>7..0</td> <td> <i>SLAVEADDR</i>: Sets 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), <i>default=0</i> In ring mode, 0 disables forwarding. </td> </tr> <tr> <td>11..8</td> <td> <i>SENDDelay</i>: 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 </td> </tr> </tbody> </table>	Bit	SLAVECONF	7..0	<i>SLAVEADDR</i> : Sets 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), <i>default=0</i> In ring mode, 0 disables forwarding.	11..8	<i>SENDDelay</i> : 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																		
				Bit	SLAVECONF																							
7..0	<i>SLAVEADDR</i> : Sets 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), <i>default=0</i> In ring mode, 0 disables forwarding.																											
11..8	<i>SENDDelay</i> : 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	INPUT	<table border="1"> <thead> <tr> <th>Bit</th> <th>INPUT</th> </tr> </thead> <tbody> <tr> <td></td> <td>Reads the digital state of all input pins available plus the state of IO pins set to output.</td> </tr> <tr> <td>0</td> <td><i>io0_in</i>: IO0 polarity</td> </tr> <tr> <td>1</td> <td><i>io1_in</i>: IO1 polarity</td> </tr> <tr> <td>2</td> <td><i>io2_in</i>: IO2 polarity</td> </tr> <tr> <td>3</td> <td><i>io3_in</i>: IO3 polarity</td> </tr> <tr> <td>4</td> <td><i>iop_in</i>: IOP pin polarity (always input in SPI mode)</td> </tr> <tr> <td>5</td> <td><i>ion_in</i>: ION pin polarity (always input in SPI mode)</td> </tr> <tr> <td>6</td> <td><i>nextaddr_in</i>: NEXTADDR pin polarity</td> </tr> <tr> <td>7</td> <td><i>drv_enn_in</i>: DRV_ENN pin polarity</td> </tr> <tr> <td>8</td> <td><i>sw_comp_in</i>: UART input comparator (1: IOP voltage is above ION voltage). The accuracy is about 20mV.</td> </tr> <tr> <td>31..</td> <td><i>VERSION</i>: 0x10=version of the IC</td> </tr> </tbody> </table>	Bit	INPUT		Reads the digital state of all input pins available plus the state of IO pins set to output.	0	<i>io0_in</i> : IO0 polarity	1	<i>io1_in</i> : IO1 polarity	2	<i>io2_in</i> : IO2 polarity	3	<i>io3_in</i> : IO3 polarity	4	<i>iop_in</i> : IOP pin polarity (always input in SPI mode)	5	<i>ion_in</i> : ION pin polarity (always input in SPI mode)	6	<i>nextaddr_in</i> : NEXTADDR pin polarity	7	<i>drv_enn_in</i> : DRV_ENN pin polarity	8	<i>sw_comp_in</i> : UART input comparator (1: IOP voltage is above ION voltage). The accuracy is about 20mV.	31..	<i>VERSION</i> : 0x10=version of the IC
Bit	INPUT																											
	Reads the digital state of all input pins available plus the state of IO pins set to output.																											
0	<i>io0_in</i> : IO0 polarity																											
1	<i>io1_in</i> : IO1 polarity																											
2	<i>io2_in</i> : IO2 polarity																											
3	<i>io3_in</i> : IO3 polarity																											
4	<i>iop_in</i> : IOP pin polarity (always input in SPI mode)																											
5	<i>ion_in</i> : ION pin polarity (always input in SPI mode)																											
6	<i>nextaddr_in</i> : NEXTADDR pin polarity																											
7	<i>drv_enn_in</i> : DRV_ENN pin polarity																											
8	<i>sw_comp_in</i> : UART input comparator (1: IOP voltage is above ION voltage). The accuracy is about 20mV.																											
31..	<i>VERSION</i> : 0x10=version of the IC																											

GENERAL CONFIGURATION REGISTERS (0x00...0x0F)				
R/W	Addr	n	Register	Description / bit names
				24 Identical numbers mean full digital compatibility.
W		4 + 4	OUTPUT	Bit OUTPUT
				Sets the IO output pin polarity and data direction.
				0 <i>io0_out</i> : IO0 output polarity
				1 <i>io1_out</i> : IO1 output polarity
				2 <i>io2_out</i> : IO2 output polarity
				3 -
				8 <i>iaddr0</i> (IO0: 0=input, 1=output)
				9 <i>iaddr1</i> (IO1: 0=input, 1=output)
				10 <i>iaddr2</i> (IO2: 0=input, 1=output)
				11 - (IO3 is always input)
W	0x05	32	X_COMPARE	<p>Position comparison register for motor 1 position strobe. Activate <i>poscmp_enable</i> to get position pulse on output PP.</p> <p><i>XACTUAL</i> = <i>X_COMPARE</i>:</p> <ul style="list-style-type: none"> - Output PP becomes high. It returns to a low state, if the positions mismatch.

6.2 Ramp Generator Registers

Addresses **Addr** are specified for motor 1 (upper value) and motor 2 (second address).

6.2.1 Ramp Generator Motion Control Register Set

RAMP GENERATOR MOTION CONTROL REGISTER SET (MOTOR 1: 0x20...0x2D, MOTOR 2: 0x40...0x4D)						
R/W	Addr	n	Register	Description / bit names	Range [Unit]	
RW	0x20 0x40	2	RAMPMODE	<p>RAMPMODE:</p> <p>0: Positioning mode (using all A, D and V parameters)</p> <p>1: Velocity mode to positive VMAX (using AMAX acceleration)</p> <p>2: Velocity mode to negative VMAX (using AMAX acceleration)</p> <p>3: Hold mode (velocity remains unchanged, unless stop event occurs)</p>	0...3	
RW	0x21 0x41	32	XACTUAL	<p>Actual motor position (signed)</p> <p><i>Hint:</i> This value normally should only be modified, when homing the drive. In positioning mode, modifying the register content will start a motion.</p>	-2 ³¹ ... +(2 ³¹)-1	
R	0x22 0x42	24	VACTUAL	<p>Actual motor velocity from ramp generator (signed)</p> <p>The sign matches the motion direction. A negative sign means motion to lower XACTUAL.</p>	+(2 ²³)-1 [μsteps / t]	
W	0x23 0x43	18	VSTART	<p>Motor start velocity (unsigned)</p> <p>Set $VSTOP \geq VSTART!$</p>	0...(2 ¹⁸)-1 [μsteps / t]	
W	0x24 0x44	16	A1	First acceleration between VSTART and V1 (unsigned)	0...(2 ¹⁶)-1 [μsteps / ta ²]	
W	0x25 0x45	20	V1	<p>First acceleration / deceleration phase threshold velocity (unsigned)</p> <p>0: Disables A1 and D1 phase, use AMAX, DMAX only</p>	0...(2 ²⁰)-1 [μsteps / t]	
W	0x26 0x46	16	AMAX	<p>Second acceleration between V1 and VMAX (unsigned)</p> <p>This is the acceleration and deceleration value for velocity mode.</p>	0...(2 ¹⁶)-1 [μsteps / ta ²]	
W	0x27 0x47	23	VMAX	<p>Motion ramp target velocity (for positioning ensure VMAX ≥ VSTART) (unsigned)</p> <p>This is the target velocity in velocity mode. It can be changed any time during a motion.</p>	0...(2 ²³)-512 [μsteps / t]	
W	0x28 0x48	16	DMAX	Deceleration between VMAX and V1 (unsigned)	0...(2 ¹⁶)-1 [μsteps / ta ²]	
W	0x2A 0x4A	16	D1	<p>Deceleration between V1 and VSTOP (unsigned)</p> <p>Attention: Do not set 0 in positioning mode, even if V1=0!</p>	1...(2 ¹⁶)-1 [μsteps / ta ²]	

RAMP GENERATOR MOTION CONTROL REGISTER SET (MOTOR 1: 0x20...0x2D, MOTOR 2: 0x40...0x4D)					
R/W	Addr	n	Register	Description / bit names	Range [Unit]
W	0x2B 0x4B	18	VSTOP	<p>Motor stop velocity (unsigned)</p> <p><i>Attention: Set VSTOP ≥ VSTART!</i></p> <p><i>Attention: Do not set 0 in positioning mode, minimum 10 recommended!</i></p>	1...(2 ¹⁸)-1 [μsteps / t]
W	0x2C 0x4C	16	TZEROWAIT	<p>Waiting time after ramping down to zero velocity before next movement or direction inversion can start and before motor power down starts. Time range is about 0 to 2 seconds.</p> <p>This setting avoids excess acceleration e.g. from VSTOP to -VSTART.</p>	0...(2 ¹⁶)-1 * 512 t _{CLK}
RW	0x2D 0x4D	32	XTARGET	<p>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.</p> <p><i>Hint:</i> The position is allowed to wrap around, thus, XTARGET value optionally can be treated as an unsigned number.</p> <p><i>Hint:</i> The maximum possible displacement is +/-((2³¹)-1).</p> <p><i>Hint:</i> When increasing V1, D1 or DMAX during a motion, rewrite XTARGET afterwards in order to trigger a second acceleration phase, if desired.</p>	-2 ³¹ ... +(2 ³¹)-1

6.2.2 Ramp Generator Driver Feature Control Register Set

RAMP GENERATOR DRIVER FEATURE CONTROL REGISTER SET (MOTOR 1: 0x30...0x36, MOTOR 2: 0x50...0x56)					
R/W	Addr	n	Register	Description / bit names	
W	0x30 0x50	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 <i>IHOLD</i> =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) <i>Hint:</i> Choose sense resistors in a way, that normal <i>IRUN</i> 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 <i>TZEROWAIT</i> 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	0x31 0x51	23	VCOOLTHRS	<p>This is the lower threshold velocity for switching on smart energy coolStep and stallGuard feature. Further it is the upper operation velocity for stealthChop. (unsigned)</p> <p>Set this parameter to disable coolStep at low speeds, where it cannot work reliably. The stop on stall function (enable with <i>sg_stop</i> when using internal motion controller) becomes enabled when exceeding this velocity. In non-dcStep mode, it becomes disabled again once the velocity falls below this threshold. This allows for homing procedures with stallGuard by blanking out the stallGuard signal at low velocities (will not work in combination with stealthChop).</p> <p>$VHIGH \geq VACT \geq VCOOLTHRS$:</p> <ul style="list-style-type: none"> - coolStep and stop on stall are enabled, if configured - Voltage PWM mode stealthChop is switched off, if configured <p>(Only bits 22..8 are used for value and for comparison)</p>	
W	0x32 0x52	23	VHIGH	<p>This velocity setting allows velocity dependent switching into a different chopper mode and fullstepping to maximize torque. (unsigned)</p> <p>$VACT \geq VHIGH$:</p> <ul style="list-style-type: none"> - coolStep is disabled (motor runs with normal current scale) - If <i>vhighchm</i> is set, the chopper switches to <i>chm</i>=1 with <i>TFD</i>=0 (constant off time with slow decay, only). - If <i>vhighfs</i> is set, the motor operates in fullstep mode. - Voltage PWM mode stealthChop is switched off, if configured <p>(Only bits 22..8 are used for value and for comparison)</p>	

RAMP GENERATOR DRIVER FEATURE CONTROL REGISTER SET (MOTOR 1: 0x30...0x36, MOTOR 2: 0x50...0x56)				
R/W	Addr	n	Register	Description / bit names
W	0x33 0x53	23	VDCMIN	<p>Automatic commutation dcStep becomes enabled above velocity <i>VDCMIN</i> (unsigned)</p> <p>In this mode, the actual position is determined by the sensorless motor commutation and becomes fed back to <i>XACTUAL</i>. In case the motor becomes heavily loaded, <i>VDCMIN</i> also is used as the minimum step velocity.</p> <p>0: Disable, dcStep off</p> <p>$VACT \geq VDCMIN \geq 256$:</p> <ul style="list-style-type: none"> - Triggers the same actions as exceeding <i>VHIGH</i>. - Switches on automatic commutation dcStep <p><i>Hint:</i> Also set bits <i>vhighfs</i> and <i>vhighchm</i> and set <i>DCCTRL</i> parameters in order to operate dcStep.</p> <p>(Only bits 22... 8 are used for value and for comparison)</p>
RW	0x34 0x54	12	SW_MODE	<p>Switch mode configuration</p> <p><i>See separate table!</i></p>
R+C	0x35 0x55	14	RAMP_STAT	<p>Ramp status and switch event status</p> <p><i>See separate table!</i></p>
R	0x36 0x56	32	XLATCH	<p>Ramp generator latch position, latches <i>XACTUAL</i> upon a programmable switch event (see <i>SW_MODE</i>).</p> <p><i>Hint:</i> The encoder position can be latched to <i>ENC_LATCH</i> together with <i>XLATCH</i> to allow consistency checks.</p>

Time reference t for velocities: $t = 2^{24} / f_{CLK}$

Time reference ta² for accelerations: $ta^2 = 2^{41} / (f_{CLK})^2$

6.2.2.1 SW_MODE – Reference Switch & stallGuard2 Event Configuration Register

0x34, 0x54: SW_MODE – REFERENCE SWITCH AND STALLGUARD2 EVENT CONFIGURATION REGISTER		
Bit	Name	Comment
11	en_softstop	<p>0: Hard stop 1: Soft stop</p> <p>The soft stop mode always uses the deceleration ramp settings <i>DMAX</i>, <i>V1</i>, <i>D1</i>, <i>VSTOP</i> and <i>TZEROWAIT</i> 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.</p> <p>A hard stop also uses <i>TZEROWAIT</i> before the motor becomes released.</p> <p><i>Attention: Do not use soft stop in combination with stallGuard2.</i></p>
10	sg_stop	<p>1: Enable stop by stallGuard2. Disable to release motor after stop event.</p> <p><i>Attention: Do not enable during motor spin-up, wait until the motor velocity exceeds a certain value, where stallGuard2 delivers a stable result, or set VCOOLTHRS to a suitable value.</i></p>
9	en_latch_encoder	1: Latch encoder position to <i>ENC_LATCH</i> upon reference switch event.
8	latch_r_inactive	1: Activates latching of the position to <i>XLATCH</i> upon an inactive going edge on the right reference switch input REFR. The active level is defined by <i>pol_stop_r</i> .
7	latch_r_active	<p>1: Activates latching of the position to <i>XLATCH</i> upon an active going edge on the right reference switch input REFR.</p> <p><i>Hint: Activate latch_r_active to detect any spurious stop event by reading status_latch_r.</i></p>
6	latch_l_inactive	1: Activates latching of the position to <i>XLATCH</i> upon an inactive going edge on the left reference switch input REFL. The active level is defined by <i>pol_stop_l</i> .
5	latch_l_active	<p>1: Activates latching of the position to <i>XLATCH</i> upon an active going edge on the left reference switch input REFL.</p> <p><i>Hint: Activate latch_l_active to detect any spurious stop event by reading status_latch_l.</i></p>
4	swap_lr	1: Swap the left and the right reference switch input REFL and REFR
3	pol_stop_r	<p>Sets the active polarity of the right reference switch input</p> <p>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</p>
2	pol_stop_l	<p>Sets the active polarity of the left reference switch input</p> <p>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</p>
1	stop_r_enable	<p>1: Enables automatic motor stop during active right reference switch input</p> <p><i>Hint: The motor restarts in case the stop switch becomes released.</i></p>
0	stop_l_enable	<p>1: Enables automatic motor stop during active left reference switch input</p> <p><i>Hint: The motor restarts in case the stop switch becomes released.</i></p>

6.2.2.2 RAMP_STAT – Ramp and Reference Switch Status Register

0x35, 0x55: RAMP_STAT – RAMP AND REFERENCE SWITCH STATUS REGISTER			
R/W	Bit	Name	Comment
R	13	<i>status_sg</i>	1: Signals an active stallGuard2 input from the coolStep driver or from the dcStep unit, if enabled. <i>Hint:</i> When polling this flag, stall events may be missed – activate <i>sg_stop</i> to be sure not to miss the stall event.
R+C	12	<i>second_move</i>	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	<i>t_zerowait_active</i>	1: Signals, that <i>TZEROWAIT</i> is active after a motor stop. During this time, the motor is in standstill.
R	10	<i>vzero</i>	1: Signals, that the actual velocity is 0.
R	9	<i>position_reached</i>	1: Signals, that the target position is reached. This flag becomes set while <i>XACTUAL</i> and <i>XTARGET</i> match.
R	8	<i>velocity_reached</i>	1: Signals, that the target velocity is reached. This flag becomes set while <i>VACTUAL</i> and <i>VMAX</i> match.
R+C	7	<i>event_pos_reached</i>	1: Signals, that the target position has been reached (<i>position_reached</i> becoming active). (Flag and interrupt condition are cleared upon reading) This bit is ORed to the <i>interrupt output</i> signal.
R+C	6	<i>event_stop_sg</i>	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 <i>interrupt output</i> signal.
R	5	<i>event_stop_r</i>	1: Signals an active stop right condition due to stop switch. The stop condition and the interrupt condition can be removed by setting <i>RAMP_MODE</i> to hold mode or by commanding a move to the opposite direction. In <i>soft_stop</i> 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 <i>interrupt output</i> signal.
	4	<i>event_stop_l</i>	1: Signals an active stop left condition due to stop switch. The stop condition and the interrupt condition can be removed by setting <i>RAMP_MODE</i> to hold mode or by commanding a move to the opposite direction. In <i>soft_stop</i> 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 <i>interrupt output</i> signal.
R+C	3	<i>status_latch_r</i>	1: Latch right ready (enable position latching using <i>SWITCH_MODE</i> settings <i>latch_r_active</i> or <i>latch_r_inactive</i>) (Flag is cleared upon reading)
	2	<i>status_latch_l</i>	1: Latch left ready (enable position latching using <i>SWITCH_MODE</i> settings <i>latch_l_active</i> or <i>latch_l_inactive</i>) (Flag is cleared upon reading)
R	1	<i>status_stop_r</i>	Reference switch right status (1=active)
	0	<i>status_stop_l</i>	Reference switch left status (1=active)

6.3 Encoder Registers

ENCODER REGISTER SET (MOTOR 1: 0x38...0x3C, MOTOR 2: 0x58...0x5C)					
R/W	Addr	n	Register	Description / bit names	Range [Unit]
RW	0x38 0x58	12	ENCMODE	Encoder configuration and use of N channel <i>See separate table!</i>	
RW	0x39 0x59	32	<i>X_ENC</i>	Actual encoder position (signed)	-2 ³¹ ... +(2 ³¹)-1
W	0x3A 0x5A	32	<i>ENC_CONST</i>	Accumulation constant (signed) 16 bit integer part, 16 bit fractional part <i>X_ENC</i> accumulates +/- <i>ENC_CONST</i> / (2 ¹⁶ * <i>X_ENC</i>) (binary) or +/- <i>ENC_CONST</i> / (10 ⁴ * <i>X_ENC</i>) (decimal) <i>ENCMODE</i> bit <i>enc_sel_decimal</i> 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) <i>reset default =</i> 1.0 (=65536)
R+C	0x3B 0x5B	1	<i>ENC_STATUS</i>	bit 0: <i>n_event</i> 1: Encoder N event detected. Status bit is cleared on read: Read (R) + clear (C) This bit is ORed to the <i>interrupt output</i> signal.	
R	0x3C 0x5C	32	<i>ENC_LATCH</i>	Encoder position <i>X_ENC</i> latched on N event	

6.2.2.3 ENCMODE – Encoder Register

0x38, 0x58: ENCMODE – ENCODER REGISTER			
Bit	Name	Comment	
11	<i>latch_now</i>	1	Latch <i>X_ENC</i> (and <i>XACTUAL</i> if selected by bit <i>latch_x_act</i>) directly upon write access to <i>ENCMODE</i> . This allows checking the encoder deviation by comparing the <i>X_LATCH</i> and <i>ENC_LATCH</i> .
		0	No action
10	<i>enc_sel_decimal</i>	0	Encoder prescaler divisor binary mode: Counts <i>ENC_CONST(fractional part)</i> /65536
		1	Encoder prescaler divisor decimal mode: Counts in <i>ENC_CONST(fractional part)</i> /10000
9	<i>latch_x_act</i>	1: Also latch <i>XACTUAL</i> position together with <i>X_ENC</i> . Allows latching the ramp generator position upon an N channel event as selected by <i>pos_edge</i> and <i>neg_edge</i> .	
8	<i>clr_enc_x</i>	0	Upon N event, <i>X_ENC</i> becomes latched to <i>ENC_LATCH</i> only
		1	Latch and additionally clear encoder counter <i>X_ENC</i> at N-event
7	<i>neg_edge</i>	n p	N channel event sensitivity
6	<i>pos_edge</i>	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	<i>clr_once</i>	1: Latch or latch and clear <i>X_ENC</i> on the next N event following the write access	
4	<i>clr_cont</i>	1: Always latch or latch and clear <i>X_ENC</i> upon an N event (once per revolution, it is recommended to combine this setting with edge sensitive N event)	
3	<i>ignore_AB</i>	0	An N event occurs only when polarities given by <i>pol_N</i> , <i>pol_A</i> and <i>pol_B</i> match.
		1	Ignore A and B polarity for N channel event
2	<i>pol_N</i>	Defines active polarity of N (0=neg., 1=pos.)	
1	<i>pol_B</i>	Required B polarity for an N channel event (0=neg., 1=pos.)	
0	<i>pol_A</i>	Required A polarity for an N channel event (0=neg., 1=pos.)	

6.4 Microstep Table Registers

COMMON MICROSTEP TABLE REGISTERS (MOTOR 1/2: 0x60...0x69)					
R/W	Addr	n	Register	Description / bit names	Range [Unit]
W	0x60	32	<i>MSLUT[0]</i> microstep table entries 0...31	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding <i>MSLUTSEL</i> W bits: 0: W= %00: -1 %01: +0 %10: +1 %11: +2 1: W= %00: +0 %01: +1 %10: +2 %11: +3	32x 0 or 1 <i>reset default=</i> <i>sine wave</i> <i>table</i>
W	0x61 ... 0x67	7 x 32	<i>MSLUT[1...7]</i> microstep table entries 32...255	This is the differential coding for the first quarter of a wave. Start values for <i>CUR_A</i> and <i>CUR_B</i> are stored for <i>MSCNT</i> position 0 in <i>START_SIN</i> and <i>START_SIN90</i> . <i>ofs31, ofs30, ..., ofs01, ofs00</i> ... <i>ofs255, ofs254, ..., ofs225, ofs224</i>	7x 32x 0 or 1 <i>reset default=</i> <i>sine wave</i> <i>table</i>
W	0x68	32	<i>MSLUTSEL</i>	This register defines four segments within each quarter <i>MSLUT</i> wave. Four 2 bit entries determine the meaning of a 0 and a 1 bit in the corresponding segment of <i>MSLUT</i> . <i>See separate table!</i>	0<X1<X2<X3 <i>reset default=</i> <i>sine wave</i> <i>table</i>
W	0x69	8 + 8	<i>MSLUTSTART</i>	bit 7... 0: <i>START_SIN</i> bit 23... 16: <i>START_SIN90</i> <i>START_SIN</i> gives the absolute current at microstep table entry 0. <i>START_SIN90</i> gives the absolute current for microstep table entry at positions 256. Start values are transferred to the microstep registers <i>CUR_A</i> and <i>CUR_B</i> , whenever the reference position <i>MSCNT=0</i> is passed.	<i>START_SIN</i> <i>reset default</i> <i>=0</i> <i>START_SIN90</i> <i>reset default</i> <i>=247</i>

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

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

- *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

6.4.1 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 W_x . 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.
30			
29			
28			
27			
26			
25			
24			
23	X2	LUT segment 2 start	For defined response the values shall satisfy: $0 < X1 < X2 < X3$
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 Motor Driver Registers

MOTOR DRIVER REGISTER SET (MOTOR 1: 0x6A...0x6F, MOTOR 2: 0x7A...0x7F)					
R/W	Addr	n	Register	Description / bit names	Range [Unit]
R	0x6A 0x7A	10	<i>MSCNT</i>	Microstep counter. Indicates actual position in the microstep table for <i>CUR_A</i> . <i>CUR_B</i> uses an offset of 256. <i>Hint</i> : Move to a position where <i>MSCNT</i> is zero before re-initializing <i>MSLUTSTART</i> or <i>MSLUT</i> and <i>MSLUTSEL</i> .	0...1023
R	0x6B 0x7B	9 + 9	<i>MSCURACT</i>	bit 8... 0: <i>CUR_A</i> (signed): Actual microstep current for motor phase A as read from <i>MSLUT</i> (not scaled by current) bit 24... 16: <i>CUR_B</i> (signed): Actual microstep current for motor phase B as read from <i>MSLUT</i> (not scaled by current)	+/-0...255
RW	0x6C 0x7C	32	<i>CHOPCONF</i>	chopper and driver configuration <i>See separate table!</i>	
W	0x6D 0x7D	25	<i>COOLCONF</i>	coolStep smart current control register and stallGuard2 configuration <i>See separate table!</i>	
W	0x6E 0x7E	8 + 8	<i>DCCTRL</i>	dcStep (<i>DC</i>) automatic commutation configuration register: bit 7... 0: <i>DC_TIME</i> : Upper PWM on time limit for commutation ($DC_TIME * 1/f_{CLK}$). Set slightly above effective blank time <i>TBL</i> . bit 15... 8: <i>DC_SG</i> : 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	
R	0x6F 0x7F	32	<i>DRV_STATUS</i>	stallGuard2 value and driver error flags <i>See separate table!</i>	

6.5.1 CHOPCONF – Chopper Configuration

0x6C, 0x7C: CHOPCONF – CHOPPER CONFIGURATION			
Bit	Name	Function	Comment
31	-	reserved	set to 0
30	<i>diss2g</i>	short to GND protection disable	0: Short to GND protection is on 1: Short to GND protection is disabled
29	<i>dedge</i>	enable double edge step pulses	1: Enable step impulse at each step edge to reduce step frequency requirement. <i>Attention: Use only in Step/Dir mode.</i>
28	<i>intpol16</i>	16 microsteps with interpolation	1: In 16 microstep mode with Step/Dir interface, the microstep resolution becomes extrapolated to 256 microsteps for smoothest motor operation
27	<i>mres3</i>	MRES micro step resolution	%0000: Native 256 microstep setting. Use this setting when the IC is operated with the internal ramp generator. %0001 ... %1000: 128, 64, 32, 16, 8, 4, 2, FULLSTEP Reduced microstep resolution for Step/Dir operation. The resolution gives the number of microstep entries per sine quarter wave. Especially when switching to a low resolution of 8 microsteps and below, take care to switch at certain microstep positions. The switching position determines the sequence of patterns. step width=2^MRES [microsteps] <i>Hint: Reduced microstep resolutions are also useful in special cases to extend the acceleration or position range</i>
26	<i>mres2</i>		
25	<i>mres1</i>		
24	<i>mres0</i>		
23	-	reserved	set to 0
22	-		
21	-		
20	-		
19	<i>vhighchm</i>	high velocity chopper mode	This bit enables switching to <i>chm</i> =1 and <i>fd</i> =0, when <i>VHIGH</i> is exceeded. This way, a higher velocity can be achieved. Can be combined with <i>vhighfs</i> =1. If set, the <i>TOFF</i> setting automatically becomes doubled during high velocity operation in order to avoid doubling of the chopper frequency.
18	<i>vhighfs</i>	high velocity fullstep selection	This bit enables switching to fullstep, when <i>VHIGH</i> 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	<i>vsense</i>	sense resistor voltage based current scaling	0: Low sensitivity, high sense resistor voltage 1: High sensitivity, low sense resistor voltage
16	<i>tbl1</i>	TBL blank time select	%00 ... %11: Set comparator blank time to 16, 24, 36 or 54 clocks <i>Hint: %01 or %10 recommended for most applications</i>
15	<i>tbl0</i>		
14	<i>chm</i>	chopper mode	0 Standard mode (spreadCycle) 1 Constant off time with fast decay time. Fast decay time is also terminated when the negative nominal current is reached. Fast decay is after on time.
13	<i>rndtf</i>	random <i>TOFF</i> time	0 Chopper off time is fixed as set by <i>TOFF</i> 1 Random mode, <i>TOFF</i> is random modulated by $dN_{CLK} = -12 \dots +3$ clocks.
12	<i>disfdcc</i>	fast decay mode	<i>chm</i> =1: <i>disfdcc</i> =1 disables current comparator usage for termination of the fast decay cycle

Step/Dir options, set 0 when *steppdir_enable* = 0

0x6C, 0x7C: <i>CHOPCONF</i> – CHOPPER CONFIGURATION			
Bit	Name	Function	Comment
11	<i>fd3</i>	<i>TFD</i> [3]	<i>chm</i> =1: MSB of fast decay time setting <i>TFD</i>
10	<i>hend3</i>	<i>HEND</i> hysteresis low value <i>OFFSET</i> sine wave offset	<i>chm</i> =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	<i>hend2</i>		
8	<i>hend1</i>		
7	<i>hend0</i>		<i>chm</i> =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	<i>hstrt2</i>	<i>HSTRT</i> hysteresis start value added to <i>HEND</i>	<i>chm</i> =0 %000 ... %111: Add 1, 2, ..., 8 to hysteresis low value <i>HEND</i> (1/512 of this setting adds to current setting) <i>Attention: Effective HEND+HSTRT ≤ 16.</i> <i>Hint: Hysteresis decrement is done each 16 clocks</i>
5	<i>hstrt1</i>		
4	<i>hstrt0</i>		
		<i>TFD</i> [2..0] fast decay time setting	<i>chm</i> =1 Fast decay time setting (MSB: <i>fd3</i>): %0000 ... %1111: Fast decay time setting <i>TFD</i> with $N_{CLK} = 32 * HSTRT$ (%0000: slow decay only)
3	<i>toff3</i>	<i>TOFF</i> 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 $TBL ≥ 36$ clocks %0010 ... %1111: 2 ... 15
2	<i>toff2</i>		
1	<i>toff1</i>		
0	<i>toff0</i>		

6.5.2 COOLCONF – Smart Energy Control coolStep and stallGuard2

0x6D, 0x7D: COOLCONF – SMART ENERGY CONTROL COOLSTEP AND STALLGUARD2

Bit	Name	Function	Comment
...	-	reserved	set to 0
24	<i>sfilt</i>	stallGuard2 filter enable	0 Standard mode, high time resolution for stallGuard2
			1 Filtered mode, stallGuard2 signal updated for each four fullsteps only to compensate for motor pole tolerances
23	-	reserved	set to 0
22	<i>sgt6</i>	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	<i>sgt5</i>		
20	<i>sgt4</i>		
19	<i>sgt3</i>		
18	<i>sgt2</i>		
17	<i>sgt1</i>		
16	<i>sgt0</i>		
15	<i>seimin</i>	minimum current for smart current control	0: 1/2 of current setting (<i>IRUN</i>) 1: 1/4 of current setting (<i>IRUN</i>)
14	<i>sedn1</i>	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	<i>sedn0</i>		
12	-	reserved	set to 0
11	<i>semax3</i>	stallGuard2 hysteresis value for smart current control	If the stallGuard2 result is equal to or above (<i>SEMIN+SEMAX+1</i>)*32, the motor current becomes decreased to save energy. %0000 ... %1111: 0 ... 15
10	<i>semax2</i>		
9	<i>semax1</i>		
8	<i>semax0</i>		
7	-	reserved	set to 0
6	<i>seup1</i>	current up step width	Current increment steps per measured stallGuard2 value %00 ... %11: 1, 2, 4, 8
5	<i>seup0</i>		
4	-	reserved	set to 0
3	<i>semin3</i>	minimum stallGuard2 value for smart current control and smart current enable	If the stallGuard2 result falls below <i>SEMIN</i> *32, the motor current becomes increased to reduce motor load angle. %0000: smart current control coolStep off %0001 ... %1111: 1 ... 15
2	<i>semin2</i>		
1	<i>semin1</i>		
0	<i>semin0</i>		

6.5.3 DRV_STATUS – stallGuard2 Value and Driver Error Flags

0x6F, 0x7F: DRV_STATUS – STALLGUARD2 VALUE AND DRIVER ERROR FLAGS			
Bit	Name	Function	Comment
31	<i>stst</i>	standstill indicator	This flag indicates motor stand still in each operation mode. It is especially useful for step & dir mode.
30	<i>olb</i>	open load indicator phase B	1: Open load detected on phase A or B. <i>Hint:</i> 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 or after a motion, only.
29	<i>ola</i>	open load indicator phase A	
28	<i>s2gb</i>	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 (<i>TOFF=0</i>) or by the ENN input.
27	<i>s2ga</i>	short to ground indicator phase A	
26	<i>otpw</i>	overtemperature pre-warning flag	1: Overtemperature pre-warning threshold is exceeded. The overtemperature pre-warning flag is common for both drivers.
25	<i>ot</i>	overtemperature flag	1: Overtemperature limit has been reached. Drivers become disabled until <i>otpw</i> is also cleared due to cooling down of the IC. The overtemperature flag is common for both drivers.
24	<i>stallGuard</i>	stallGuard2 status	1: Motor stall detected (<i>SG_RESULT=0</i>) or dcStep stall in dcStep mode.
23	-	reserved	Ignore these bits
22			
21			
20	<i>CS ACTUAL</i>	actual motor current / smart energy current	
19			
18			
17			
16			
15	<i>fsactive</i>	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	<i>SG_RESULT</i>	stallGuard2 result respectively PWM on time for coil A in stand still for motor temperature detection	<p>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 <i>SGT</i> setting, this is an indicator for a motor stall. The stall detection compares <i>SG_RESULT</i> to 0 in order to detect a stall. <i>SG_RESULT</i> 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.</p> <p><i>SG_RESULT</i> is also applicable when dcStep is active. stallGuard2 works best with microstep operation.</p> <p>Temperature measurement: In standstill, no stallGuard2 result can be obtained. <i>SG_RESULT</i> 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.</p>
8			
7			
6			
5			
4			
3			
2			
1			
0			

6.6 Voltage PWM mode stealthChop

MOTOR DRIVER PWM REGISTER SET (MOTOR 1: 0x10...0x17, MOTOR 2: 0x18...0x1F)					
R/W	Addr	n	Register	Description / bit names	Range [Unit]
W	0x10 0x18	22	PWMCONF	Voltage PWM mode chopper configuration <i>See separate table!</i>	
R	0x11 0x19	8	PWM_ STATUS	Actual PWM scaler (255=max. Voltage)	0...255

6.6.1 PWMCONF – Voltage PWM mode stealthChop

0x10, 0x18: PWMCONF – VOLTAGE MODE PWM					
Bit	Name	Function	Comment		
...	-	reserved	Set to 0		
21	<i>freewheel1</i>	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	<i>freewheel0</i>				
19	-	reserved	Set to 0		
18	<i>pwm_autoscale</i>	PWM automatic amplitude scaling	0	User defined PWM amplitude. The current settings have no influence.	
			1	Enable automatic current control <i>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.</i>	
17	<i>pwm_freq1</i>	PWM frequency selection	%00: $f_{PWM}=2/1024 f_{CLK}$		
16	<i>pwm_freq0</i>		%01: $f_{PWM}=2/683 f_{CLK}$ %10: $f_{PWM}=2/512 f_{CLK}$ %11: $f_{PWM}=2/410 f_{CLK}$		
15	PWM_ GRAD	User defined regulation loop gradient (bits 15...12 currently unused, set to 0)	pwm_autoscale=0	0: stealthChop disabled	
14				1...15: stealthChop enabled (the actual value is not used)	
13			pwm_autoscale=1	0: stealthChop disabled	
12				1...15: User defined maximum PWM amplitude change per half wave (1 to 15)	
11				User defined PWM amplitude The resulting amplitude (0...255) is set by this value.	
10					
9					
8					
7	PWM_ AMPL	User defined amplitude	pwm_autoscale=0	User defined PWM amplitude The resulting amplitude (0...255) is set by this value.	
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					

7 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.

The internal resistance of 20mΩ will be increased by external trace resistance, 5mΩ are realistic.

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	0.21	0.12
0.82	0.26	0.15
0.75	0.28	0.16
0.68	0.31	0.18
0.50	0.42	0.24
0.47	0.45	0.25
0.33	0.63	0.35
0.27	0.76	0.43
0.22	0.91	0.51
0.15	1.29*)	0.72

*) Value exceeds upper current rating for single motor operation.

Hint

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

Parameter	Description	Setting	Comment
<i>IRUN</i>	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 <i>coolStep</i> .	0 ... 31	scaling factor 1/32, 2/32, ... 32/32
<i>IHOLD</i>	Identical to <i>IRUN</i> , but for motor in stand still.		
<i>IHOLD DELAY</i>	Allows smooth current reduction from run current to hold current. <i>IHOLDDELAY</i> controls the number of clock cycles for motor power down after <i>TZEROWAIT</i> in increments of 2 ¹⁸ clocks: 0=instant power down, 1..15: Current reduction delay per current step in multiple of 2 ¹⁸ clocks. <i>Example:</i> When using <i>IRUN</i> =31 and <i>IHOLD</i> =16, 15 current steps are required for hold current reduction. A <i>IHOLDDELAY</i> 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 1 ...15	instant <i>IHOLD</i> 1*2 ¹⁸ ... 15*2 ¹⁸ clocks per current decrement
<i>vsense</i>	Allows control of the sense resistor <i>voltage range</i> for full scale current.	0 1	V _{FS} = 0.32 V V _{FS} = 0.18 V

7.1 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 24.

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.

8 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.

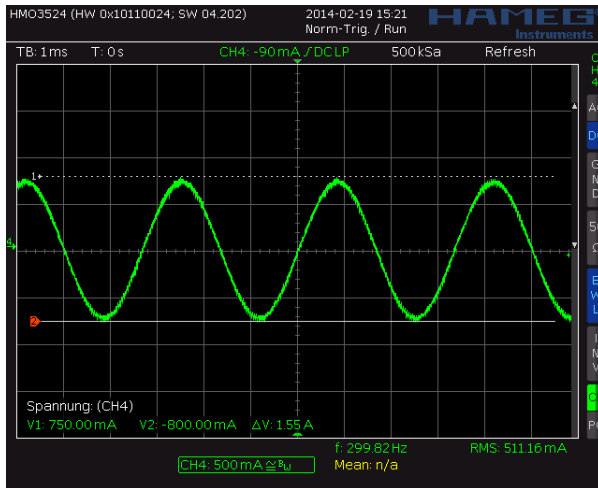


Figure 8.1 Motor coil sine wave current with stealthChop (measured with current probe)

8.1 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 fixed scale mode (*pwm_autoscale* = 0). The fixed scale mode will not react to a change of operating conditions like 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, velocity 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 $f_{PWM}=2/1024 f_{CLK}$	PWM_FREQ=%01 $f_{PWM}=2/683 f_{CLK}$	PWM_FREQ=%10 $f_{PWM}=2/512 f_{CLK}$	PWM_FREQ=%11 $f_{PWM}=2/410 f_{CLK}$
18MHz	35.2kHz	52.7kHz	70.3kHz	87.8kHz
16MHz	31.3kHz	46.9kHz	62.5kHz	78.0kHz
(internal)	~26kHz	~38kHz	~52kHz	~64kHz
12MHz	23.4kHz	35.1kHz	46.9kHz	58.5kHz
10MHz	19.5kHz	29.3kHz	39.1kHz	48.8kHz
8MHz	15.6kHz	23.4kHz	31.2kHz	39.0kHz

Table 8.1 Choice of PWM frequency – green: recommended

8.2 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 8.4). 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.

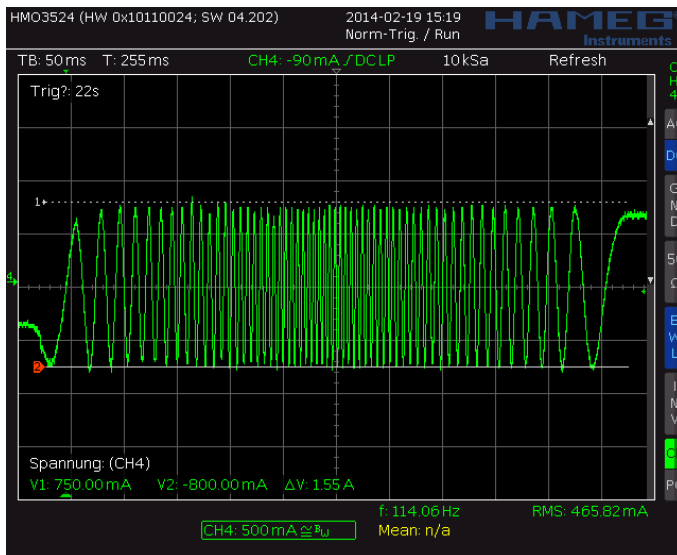


Figure 8.2 Scope shot: good setting for PWM_GRAD

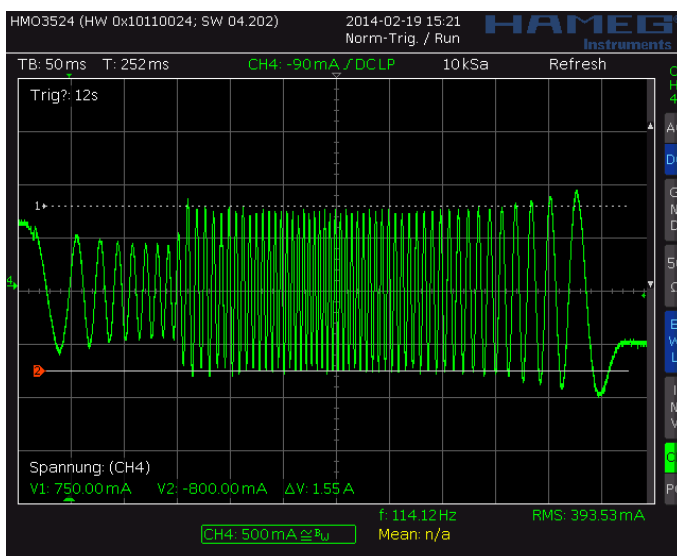


Figure 8.3 Scope shot: too small setting for PWM_GRAD

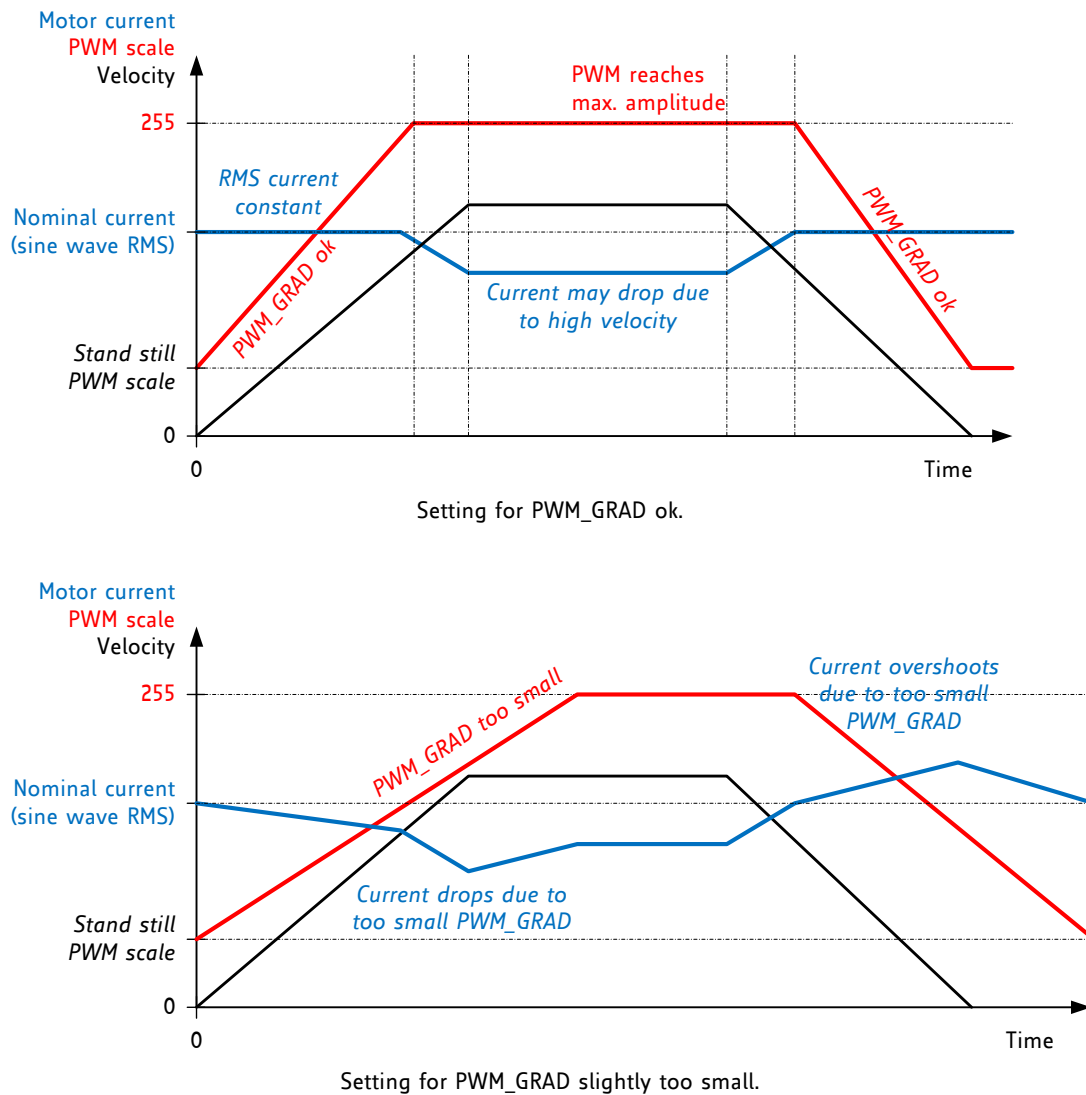


Figure 8.4 Good and too small setting for PWM_GRAD

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 18.

8.2.1 Lower Current Limit

The autoscaling function imposes a lower limit for motor current regulation. As the coil current can be measured in the shunt resistor during chopper on phase, only, a minimum chopper duty cycle allowing coil current regulation is given by the blank time as set by TBL and by the chopper frequency setting. Therefore, the motor specific minimum coil current in stealthChop autoscaling mode rises with the supply voltage and the chopper frequency. A lower blanking time allows a 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. The run current setting needs to be kept above the lower limit: 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. The lower motor coil current limit can be calculated from motor parameters and chopper settings:

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

With V_M the motor supply voltage and R_{COIL} the motor coil resistance.

$I_{Lower\ Limit}$ can be treated as a thumb value for the minimum possible motor current setting.

Example:

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

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

For *pwm_autoscale* mode, a lower coil current limit applies. This limit can be calculated or measured using a current probe. Keep the motor run-current setting *IRUN* well above this lower current limit.

8.2.2 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 *VCOOLTHRS*. 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 8.4 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, $\frac{1}{2}$ to $\frac{3}{4}$ 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 *VCOOLTHRS*), *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.

8.2.3 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.

8.3 Fixed Scaling

Non-automatic, fixed scaling scales the stealthChop amplitude based on the user defined value *PWM_AMPL*. 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* at low velocities 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. A higher scale value is necessary to compensate for this. When a higher value is chosen, it should be made sure, that the maximum driver current is not exceeded during the acceleration phase. This can be checked with the above formula. A short time excess current will not do harm to the motor.

Hint

The setting for PWM_AMPL 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.

8.4 Combining stealthChop with other Chopper Modes

The TMC5072 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. *VCOOLTHRS* 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 *VCOOLTHRS* 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 *VCOOLTHRS* and *VHIGH* to high values (above *VMAX*) 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.

8.5 Flags in stealthChop

8.5.1 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.

8.5.2 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).

8.6 Freewheeling and Passive Motor Braking

stealthChop provides different options for motor standstill. These options can be enabled by setting the standstill current I_{HOLD} to zero and choosing the desired option using the $FREEWHEEL$ setting. The desired option becomes enabled after a time period specified by $TZEROWAIT$ and I_{HOLD_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
$VCOOLTHRS$ $VHIGH$	Whichever is lower, specifies the upper velocity for operation in stealthChop voltage PWM mode.	0 ... $2^{23}-1$	
$pwm_autoscale$	Enable automatic current scaling using current measurement or use fixed scaling mode.	0 1	Fixed mode 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 1 2 3	$f_{PWM}=1/1024 f_{CLK}$ $f_{PWM}=1/683 f_{CLK}$ $f_{PWM}=1/512 f_{CLK}$ $f_{PWM}=1/410 f_{CLK}$
PWM_GRAD	Global enable and regulation loop gradient when $pwm_autoscale=1$.	0 1 ... 15	Do not use stealthChop stealthChop enabled
PWM_AMPL	User defined PWM amplitude for fixed scaling or amplitude limit for re-entry into stealthChop mode when $pwm_autoscale=1$.	0 ... 255	
$FREEWHEEL$	Stand still option when motor current setting is zero ($I_{HOLD}=0$). Only available with stealthChop enabled. The freewheeling option makes the motor easy movable, while both coil short options realize a passive brake. Mode 2 will brake more intensely than mode 3, because low side drivers (LS) have lower resistance than high side drivers.	0 1 2 3	Normal operation Freewheeling Coil shorted using LS drivers 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 1 ... 15	Driver off Driver enabled
TBL	Selects the comparator <i>blank time</i> . 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 1 2 3	$16 t_{CLK}$ $24 t_{CLK}$ $36 t_{CLK}$ $54 t_{CLK}$
$IRUN$ I_{HOLD}	Run and hold current setting for stealth Chop operation – only used with $pwm_autoscale=1$		See chapter on current setting for details

9 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 9.1 the different chopper phases are shown.

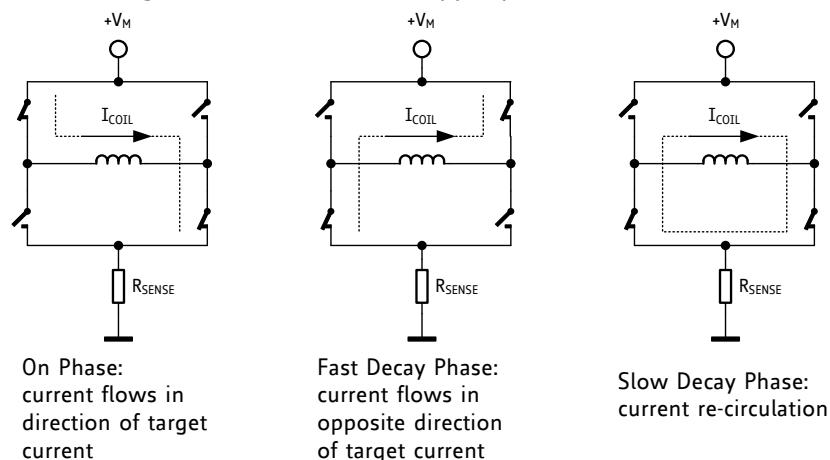


Figure 9.1 Chopper phases

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
<i>TOFF</i>	Sets the slow decay time (<i>off time</i>). This setting also limits the maximum chopper frequency. For operation with <i>stealthChop</i> , this parameter is not used, but it is required to enable the motor. In case of operation with <i>stealthChop</i> 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)
<i>TBL</i>	Selects the comparator <i>blank time</i> . 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}
<i>chm</i>	Selection of the <i>chopper mode</i>	0	spreadCycle
		1	classic const. off time

9.1 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 9.3). 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: 25kHz

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 9.2). 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.

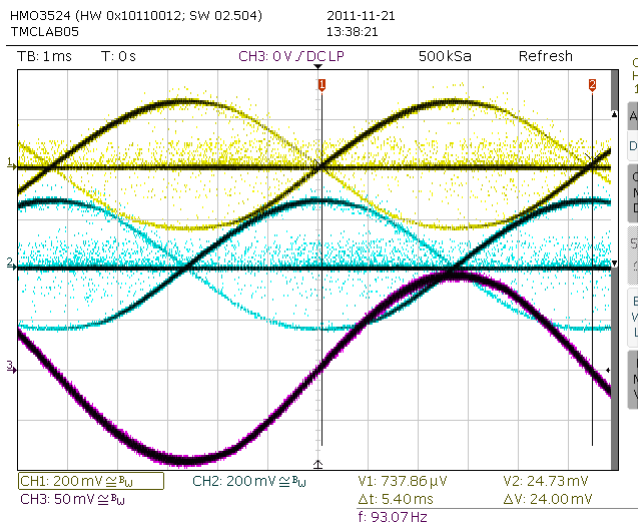


Figure 9.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)

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

Quick Start

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

For detail procedure see Application Note AN001 - *Parameterization of spreadCycle*

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

The hysteresis principle could in some cases lead to the chopper frequency becoming too low, e.g. when the coil resistance is high when compared to the supply voltage. This is avoided by splitting the hysteresis setting into a start setting (*HSTRT+HEND*) and an end setting (*HEND*). An automatic hysteresis decremter (HDEC) interpolates between both settings, by decremting 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 decremments 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.

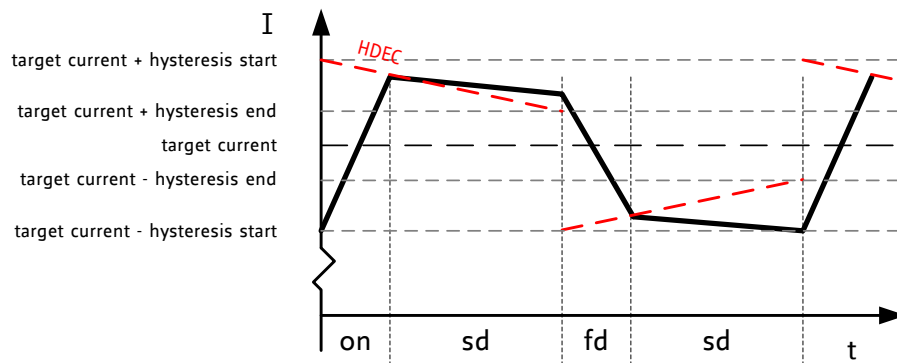


Figure 9.3 spreadCycle chopper scheme showing coil current during a chopper cycle

Two parameters control spreadCycle mode:

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

Even at *HSTRT*=0 and *HEND*=0, the TMC5072 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, 2 or 3 and a short *TBL* of 1 or 0.

9.2 Classic Constant Off Time Chopper

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

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

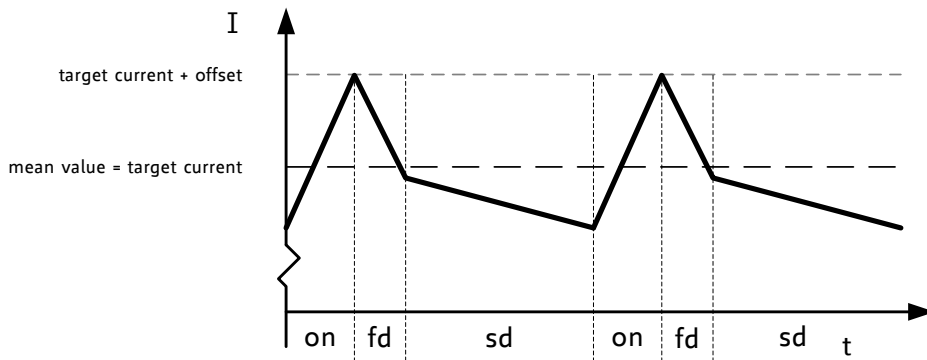
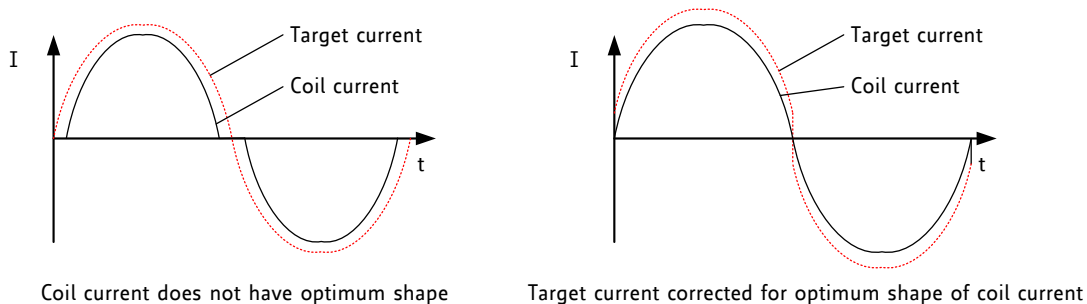


Figure 9.4 Classic const. off time chopper with offset showing coil current

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 9.5). If the zero offset is too low, the motor stands still for a short moment during current zero crossing. If it is set too high, it makes a larger microstep. Typically, a positive offset setting is required for smoothest operation.



Coil current does not have optimum shape

Target current corrected for optimum shape of coil current

Figure 9.5 Zero crossing with classic chopper and correction using sine wave offset

Three parameters control constant off-time mode:

Parameter	Description	Setting	Comment
<i>TFD</i> (<i>fd3</i> & <i>HSTR1</i>)	<i>Fast decay time</i> 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
<i>OFFSET</i> (<i>HEND</i>)	<i>Sine wave offset</i> . 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
<i>disfdcc</i>	Selects usage of the <i>current comparator</i> for termination of the <i>fast decay</i> cycle. If current comparator is enabled, it terminates the fast decay cycle in case the current reaches a higher negative value than the actual positive value.	0	enable comparator termination of fast decay cycle
		1	end by time only

9.3 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 layouts hint in chapter 24).

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
<i>rndtf</i>	This bit switches on a <i>random off time</i> generator, which slightly modulates the off time <i>TOFF</i> using a random polynomial.	0	disable
		1	random modulation enable

10 Driver Diagnostic Flags

The TMC5072 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.

10.1 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.

10.2 Short to GND Protection

The TMC5072 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.

10.3 Open Load Diagnostics

Interrupted cables are a common cause for systems failing, e.g. when connectors are not firmly plugged. The TMC5072 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.

11 Ramp Generator

The ramp generator allows motion based on target position or target velocity. It automatically calculates the optimum motion profile taking into account acceleration and velocity settings. The TMC5072 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 11.2.

11.1 Real World Unit Conversion

The TMC5072 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 TMC5072 register content are written as register[5072].

PARAMETER VS. UNITS		
Parameter / Symbol	Unit	calculation / description / comment
f_{CLK} [Hz]	[Hz]	clock frequency of the TMC5072 in [Hz]
s	[s]	second
US	μ step	
FS	fullstep	
μ step velocity v[Hz]	μ steps / s	$v[\text{Hz}] = v[5072] * (f_{CLK}[\text{Hz}]/2 / 2^{23})$
μ step acceleration a[Hz/s]	μ steps / s ²	$a[\text{Hz/s}] = a[5072] * f_{CLK}[\text{Hz}]^2 / (512*256) / 2^{24}$
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	$v[\text{rps}] = v[\mu\text{steps/s}] / \text{USC} / \text{FSC}$ FSC: motor fullsteps per rotation, e.g. 200
rps acceleration a[rps/s ²]	rotations / s ²	$a[\text{rps/s}^2] = a[\mu\text{steps/s}^2] / \text{USC} / \text{FSC}$
ramp steps[μ steps] = rs	μ steps	$rs = (v[5072])^2 / a[5072] / 2^8$ microsteps during linear acceleration ramp (assuming acceleration from 0 to v)

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* option *MRES=%0001* will double the motor velocity for the same speed setting and thus also double effective acceleration and deceleration. The motor will have half position resolution with this setting.

Quick Start

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

11.2 Motion Profiles

For the ramp generator register set, please refer to the chapter 6.2.

11.2.1 Ramp Mode

The ramp generator delivers two phase acceleration and two phase deceleration ramps with additional programmable start and stop velocities (see Figure 11.1).

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*.

11.2.2 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.

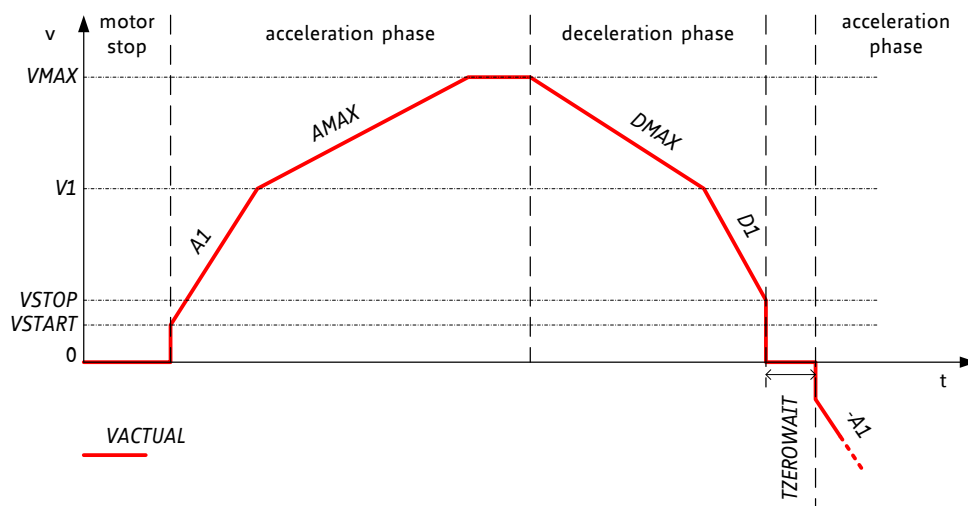


Figure 11.1 Ramp generator velocity trace showing consequent move in negative direction

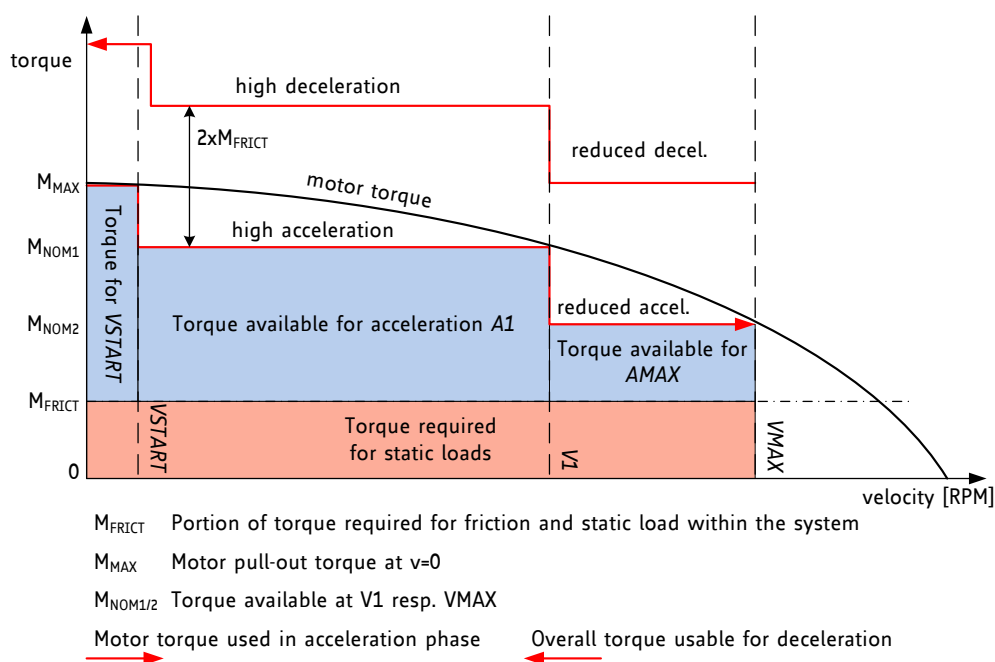


Figure 11.2 Illustration of optimized motor torque usage with TMC5072 ramp generator

11.2.3 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.

11.2.4 Early Ramp Termination

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

OPTIONS TO TERMINATE MOTION USING ACCELERATION SETTINGS:

- Switch to velocity mode, set $VMAX=0$ and $AMAX$ to the desired deceleration value. This will stop the motor using a linear ramp.
- For a stop in positioning mode, set $VSTART=0$ and $VMAX=0$. $VSTOP$ is not used in this case. The driver will use $AMAX$ and $A1$ (as determined by $V1$) for going to zero velocity.
- For a stop using $D1$, $DMAX$ and $VSTOP$, trigger the deceleration phase by copying $XACTUAL$ to $XTARGET$. Set $TZEROWAIT$ sufficiently to allow the CPU to interact during this time. The driver will decelerate and eventually come to a stop. Poll the actual velocity to terminate motion during $TZEROWAIT$ time using option a) or b).
- Activate a stop switch. This can be done by means of the hardware input, e.g. using a wired 'OR' to the REF switch input. If you do not use the hardware input and have tied the REFL and REFR to a fixed level, enable the stop function (*stop_l_enable*, *stop_r_enable*) and use the inverting function (*pol_stop_l*, *pol_stop_r*) to simulate the switch activation.

11.2.5 Application Example: Joystick Control

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

REALIZE JOYSTICK CONTROL

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

11.3 Interrupt Handling

The motion controllers provide the capability to issue an interrupt to the microcontroller, e.g. in order to react on a position reached event. In case more than one interrupt source is possible, it is necessary to carefully check for the actual event, without risking losing an event.

INTERRUPT HANDLING FOR 2 AXIS (EXAMPLE FOR TARGET_REACHED):

1. Read $RAMP_STAT1$ to clear the interrupt flags. This will turn off the interrupt source.
2. Check $X_{ACTUAL1}$ for reaching of the target position (and any other conditions you want to check for ramp 1).
3. Do the same for $RAMP_STAT2$ and $X_{ACTUAL2}$.

This way, you are sure that you will not miss any *target_reached* condition, because you first clear the flags, and afterwards read out the condition.

11.4 Velocity Thresholds

The ramp generator provides a number of velocity thresholds coupled to the actual velocity V_{ACTUAL} . The different ranges allow programming the motor to the optimum step mode, coil current and acceleration settings. For the range labeled "microstepping" in Figure 11.3, either stealthChop or spreadCycle can be used, if enabled.

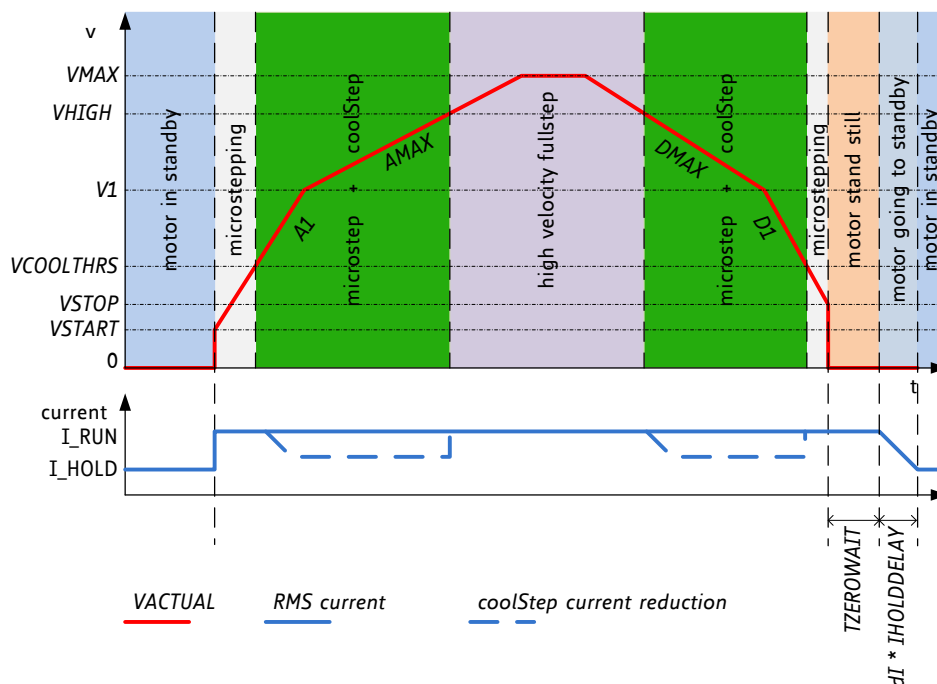


Figure 11.3 Ramp generator velocity dependent motor control

Note

Since it is not necessary to differentiate the velocity to the last detail, the velocity thresholds use a reduced number of bits for comparison and the lower eight bits of the compare values become ignored.

11.5 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 (D_{MAX}, 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.

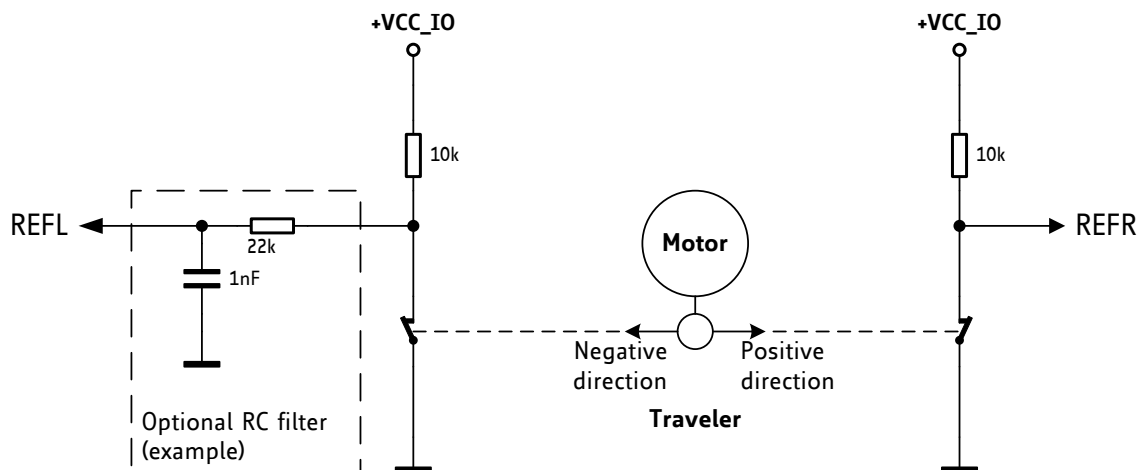


Figure 11.4 Using reference switches (example)

Normally open or normally closed switches can be used by programming the switch polarity or selecting the pull-up 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 TMC5072 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.

12 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 12.1. 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.

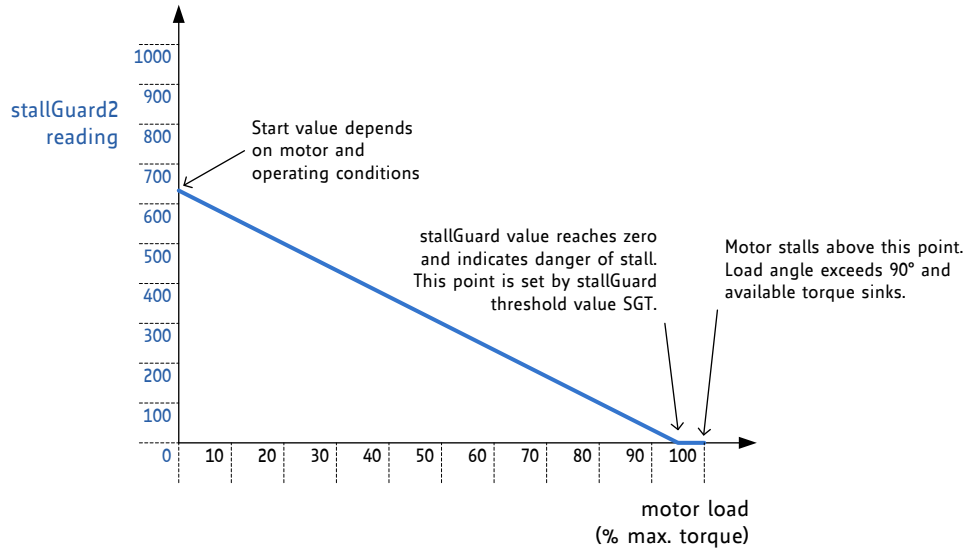


Figure 12.1 Function principle of stallGuard2

Parameter	Description	Setting	Comment
<i>SGT</i>	This signed value controls the stallGuard2 threshold level for stall detection and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value working with most motors. A higher value makes stallGuard2 less sensitive and requires more torque to indicate a stall.	0	indifferent value
		+1... +63	less sensitivity
		-1... -64	higher sensitivity
<i>sfilt</i>	Enables the stallGuard2 filter for more precision of the measurement. If set, reduces the measurement frequency to one measurement per electrical period of the motor (4 fullsteps).	0	standard mode
		1	filtered mode
Status word	Description	Range	Comment
<i>SG</i>	This is the <i>stallGuard2 result</i> . A higher reading indicates less mechanical load. A lower reading indicates a higher load and thus a higher load angle. Tune the <i>SGT</i> setting to show a <i>SG</i> 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!

12.1 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 *VCOOLTHRS* 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 18.

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

12.1.1 Variable Velocity Limits *VCOOLTHRS* and *VHIGH*

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.

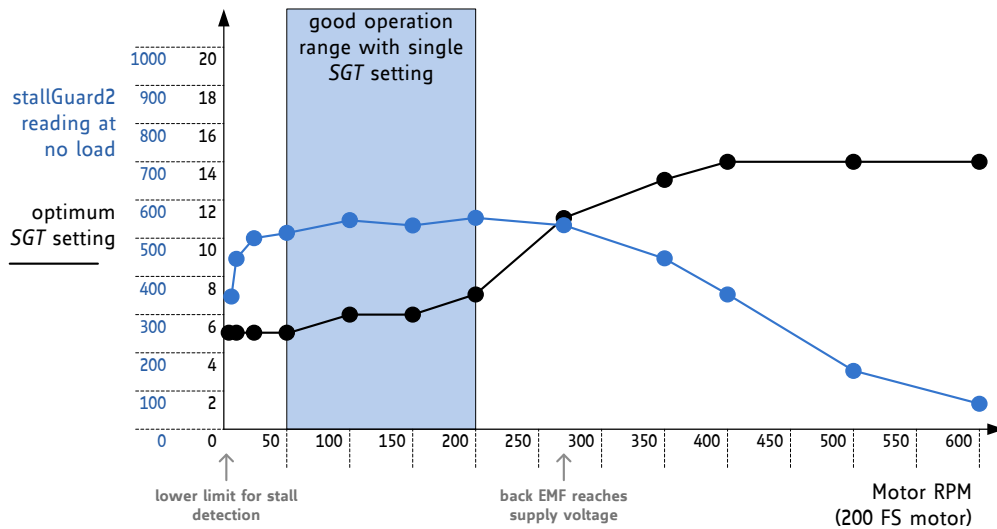


Figure 12.2 Example: Optimum *SGT* setting and stallGuard2 reading with an example motor

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

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

12.1.2 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.

12.1.3 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.

12.1.4 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:

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

12.2 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.

12.3 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.

12.4 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 *TZEROWAIT* in case the motion parameters have not been modified.

12.5 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.

13 coolStep Operation

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

13.1 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.

13.2 Setting up for coolStep

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

Parameter	Description	Range	Comment
<i>SEMIN</i>	4-bit unsigned integer that sets a <i>lower threshold</i> . If <i>SG</i> goes below this threshold, coolStep increases the current to both coils. The 4-bit <i>SEMIN</i> value is scaled by 32 to cover the lower half of the range of the 10-bit <i>SG</i> value. (The name of this parameter is derived from smartEnergy, which is an earlier name for coolStep.)	0 1...15	disable coolStep threshold is $SEMIN * 32$
<i>SEMAX</i>	4-bit unsigned integer that controls an <i>upper threshold</i> . If <i>SG</i> 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 13.1 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.

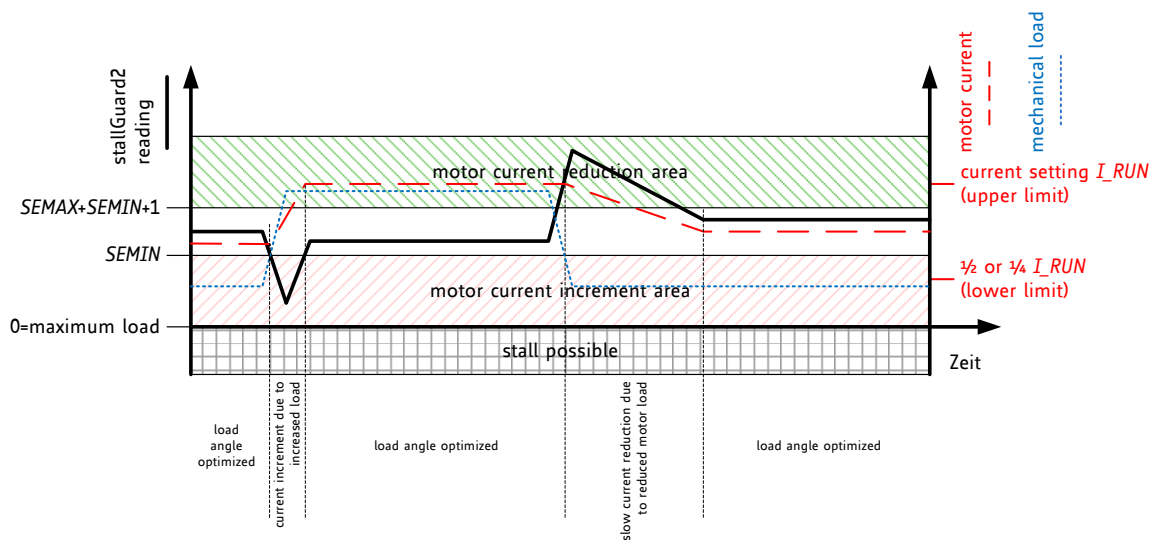


Figure 13.1 coolStep adapts motor current to the load

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

Parameter	Description	Range	Comment
<i>SEUP</i>	Sets the <i>current increment step</i> . The current becomes incremented for each measured stallGuard2 value below the lower threshold.	0...3	step width is 1, 2, 4, 8
<i>SEDN</i>	Sets the number of stallGuard2 readings above the upper threshold necessary for each <i>current decrement</i> of the motor current.	0...3	number of stallGuard2 measurements per decrement: 32, 8, 2, 1
<i>SEIMIN</i>	Sets the <i>lower motor current limit</i> for coolStep operation by scaling the <i>IRUN</i> current setting.	0 1	0: 1/2 of IRUN 1: 1/4 of IRUN
<i>VCOOL THRS</i>	Lower ramp generator velocity threshold. 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. <i>Hint:</i> May be adapted to disable coolStep during acceleration and deceleration phase by setting identical to <i>VMAX</i> .	1... 2 ²³	
<i>VHIGH</i>	Upper ramp generator velocity threshold value. Above this velocity coolStep becomes disabled (not used in Step/Dir mode). Adapt to the velocity range where stallGuard2 gives a stable result.	1... 2 ²³	Also controls additional functions like switching to fullstepping.
Status word	Description	Range	Comment
<i>CSACTUAL</i>	This status value provides the <i>actual motor current scale</i> as controlled by coolStep. The value goes up to the <i>IRUN</i> value and down to the portion of <i>IRUN</i> as specified by <i>SEIMIN</i> .	0...31	1/32, 2/32, ... 32/32

13.3 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.

13.3.1 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.


13.3.2 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.

14 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.

14.1 User Benefits

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

14.2 Designing-In dcStep

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

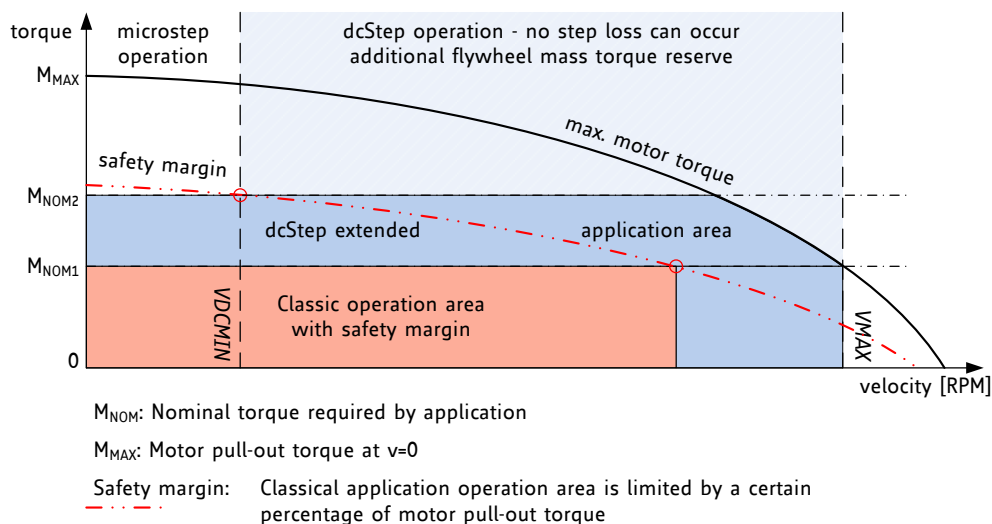


Figure 14.1 dcStep extended application operation area

Quick Start

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

For detail configuration procedure see Application Note AN003 - *dcStep*

14.3 Enabling dcStep

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 V_{ACTUAL} or at reduced velocity if the motor becomes overloaded. It requires setting the minimum operation velocity V_{DCMIN} . V_{DCMIN} 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 V_{DCMIN} . 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.

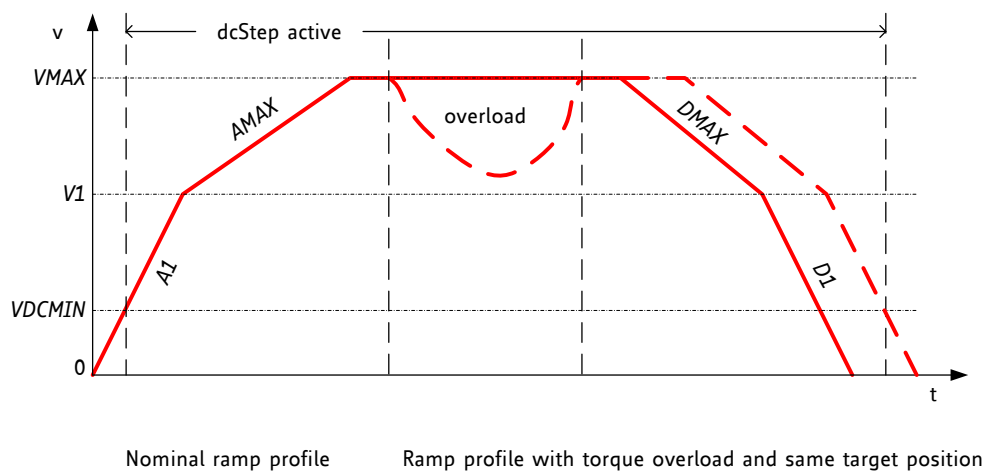


Figure 14.2 Velocity profile with impact by overload situation

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 15.2 for an initialization with the default table.

14.4 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 V_{ACTUAL} 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 $V_{COOLTHRS}$ to a velocity slightly above V_{DCMIN} or up to V_{MAX} .

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
<i>vhighfs</i> & <i>vhighchm</i>	These chopper configuration flags in <i>CHOPCONF</i> need to be set for dcStep operation. As soon as <i>VDCMIN</i> becomes exceeded, the chopper becomes switched to fullstepping.	0 / 1	set to 1 for dcStep
<i>TOFF</i>	dcStep often benefits from an increased off time value in <i>CHOPCONF</i> . Settings >2 should be preferred.	2... 15	Settings 8...15 do not make any difference to setting 8 for dcStep operation.
<i>VDCMIN</i>	This is the lower threshold for dcStep operation. Below this threshold, the motor operates in normal microstep mode. In dcStep operation, the motor operates at minimum <i>VDCMIN</i> , even when it is completely blocked. Tune together with <i>DC_TIME</i> setting. stealthChop is disabled in dcStep mode above <i>VDCMIN</i> .	0... 2 ²²	0: Disable dcStep Set to the lower velocity limit for dcStep operation.
<i>DC_TIME</i>	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 <i>VDCMIN</i> . 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 <i>TBL</i>) in clock cycles + 1
<i>DC_SG</i>	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 <i>sg_stop</i> 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 <i>DC_TIME</i> / 16

14.5 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 16.0MHz 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.

15 Sine-Wave Look-up Table

The TMC5072 driver provides a programmable look-up table for storing the microstep current wave. It is common to both drivers. 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.

15.1 User Benefits

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

15.2 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 W_x or W_{x+1} when advancing one step in the table. When W_x 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 W_x can be programmed to -1, 0, 1, or 2 using up to four flexible programmable segments within the quarter wave. This way even negative inclination can be realized. The four inclination segments are controlled by the position registers $X1$ to $X3$. Inclination segment 0 goes from microstep position 0 to $X1-1$ and its base inclination is controlled by $W0$, segment 1 goes from $X1$ to $X2-1$ with its base inclination controlled by $W1$, etc.

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

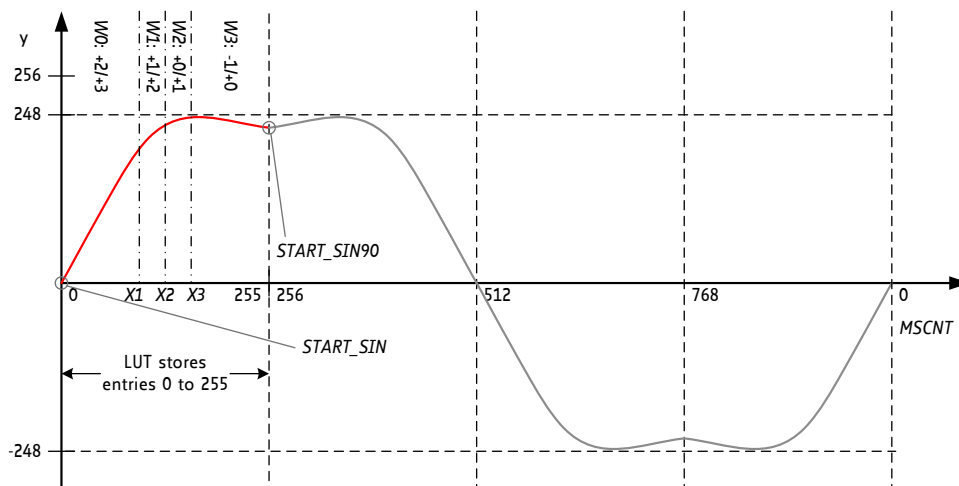


Figure 15.1 LUT programming example

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 6.4 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 TMC5072-EVAL comes with a calculation tool for own waves.

Initialization example for the default microstep table:

```
MSLUT[0]= %10101010101010101010101010101010 = 0xAAAAAB554
MSLUT[1]= %0100101010010101010101010010101010 = 0x4A9554AA
MSLUT[2]= %001001000100100100100100100101001 = 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
```

16 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*.

dcStep cannot be used in this mode, as the driver has no means to feedback taken steps to the external motion controller!

16.1 Timing

Figure 16.1 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.

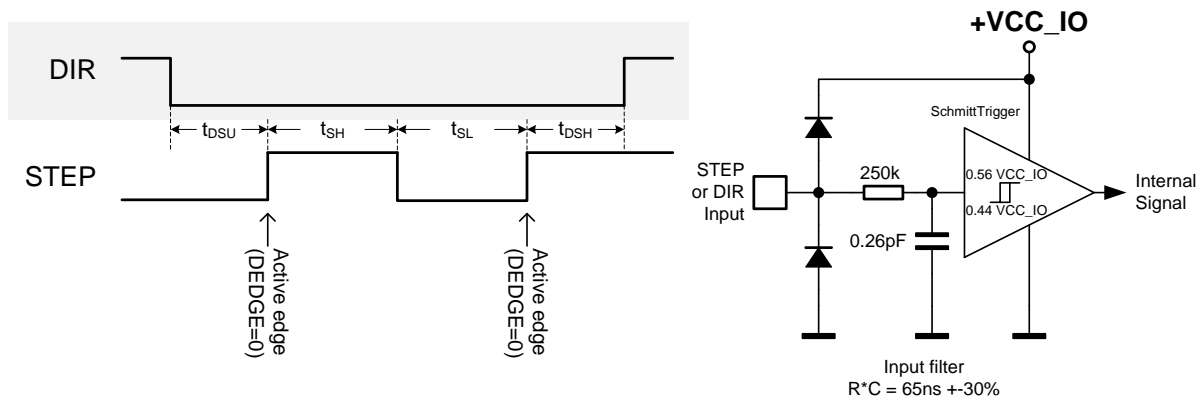


Figure 16.1 STEP and DIR timing, Input pin filter

STEP and DIR interface timing		AC-Characteristics				
		clock period is t_{CLK}				
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
step frequency (at maximum microstep resolution)	f_{STEP}	$dedge=0$			$\frac{1}{2} f_{CLK}$	
		$dedge=1$			$\frac{1}{4} f_{CLK}$	
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

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

16.2 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 TMC5072 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.

16.2.1 Working with Half- and Fullstep Resolution

Fullstepping is desirable in some applications, where maximum torque at maximum velocity with a given motor is desired. Especially at low microstep resolutions like full- or halfstepping, the absolute current values and thus the absolute positions in the table are important for best motor performance. Thus, a software which uses resolution switching in order to get maximum torque and velocity from the drive, should switch the resolution at or near certain positions, as shown in the following table.

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%

Table 16.1 Optimum position sequence for half- and full stepping

16.3 microPlyer Step Interpolator and Stand Still Detection

For each active edge on STEP, microPlyer produces 16 microsteps at 256x resolution, as shown in Figure 16.2.

Enable microPlyer by setting the *intpol16* bit in the *CHOPCONF* register. It only supports input at 16x setting, which it transforms into 256x resolution. Operation is only possible in STEP/DIR mode.

The step rate for the 16 microsteps is determined by measuring the time interval of the previous step period and dividing it into 16 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 (one fullstep per second). 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 $(\text{system clock frequency})/2^{16} - 256$ Hz. When a stand still is detected, the driver automatically switches the motor to holding current *IHOLD*.

Attention

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

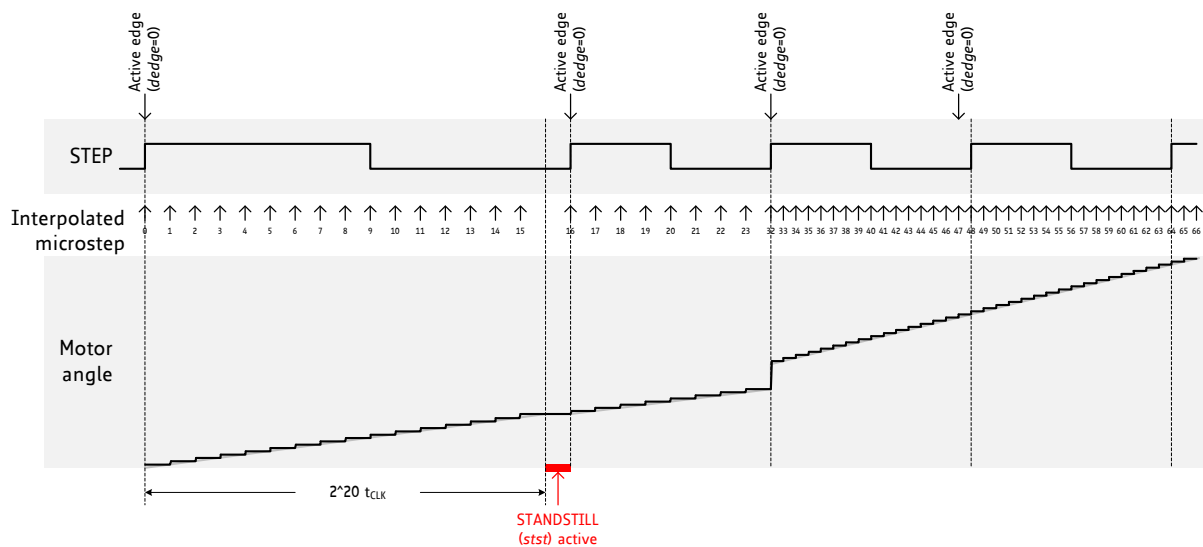


Figure 16.2 microPlyer microstep interpolation with rising STEP frequency

In Figure 16.2, 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.

17 ABN Incremental Encoder Interface

The TMC5072 is equipped with two incremental encoder interfaces 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.

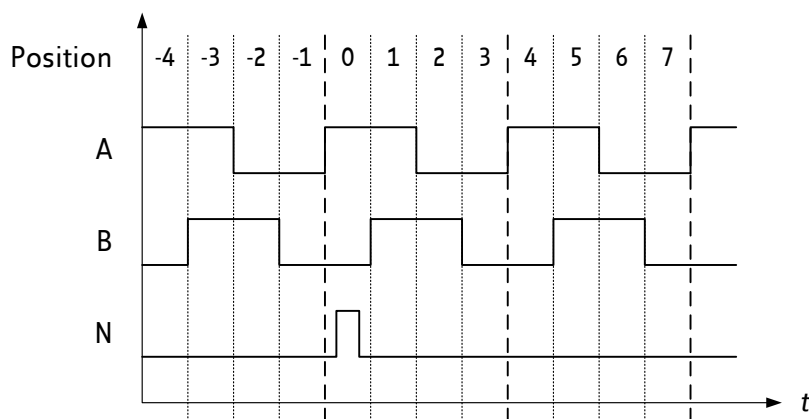


Figure 17.1 Outline of ABN signals of an incremental encoder

THE ENCODER CONSTANT *ENC_CONST*

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

Examples:

- Encoder factor of 1.0: $ENC_CONST = 0x0001.0x0000 = \text{FACTOR.FRACTION}$
- Encoder factor of -1.0: $ENC_CONST = 0xFFFF.0x0000$. This is the two's complement of $0x00010000$. It equals $(2^{16} - (\text{FACTOR} + 1)) \cdot (2^{16} - \text{FRACTION})$
- Decimal mode encoder factor 25.6: $00025.6000 = 0x0019.0x1770 = \text{FACTOR.DECIMALS}$
- Decimal mode encoder factor -25.6: $0xFFE6.4000 = 0xFFE6.0x0FA0$. This equals $(2^{16} - (\text{FACTOR} + 1)) \cdot (10000 - \text{DECIMALS})$

THE ENCODER COUNTER X_ENC

The encoder counter X_ENC holds the current encoder position ready for read out. Different modes concerning handling of the signals A, B, and N take into account active low and active high signals found with different types of encoders. For more details please refer to the register mapping in section 6.3.

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.

17.1 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		$3 t_{CLK}$		

17.2 Setting the Encoder to Match Motor Resolution

Encoder example settings for motor parameters: USC=256 μ steps, 200 fullstep motor
Factor = $FSC * USC / \text{encoder resolution}$

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
 $ENC_CONST = 51 * 2^{16} + 0.2 * 10000$

17.3 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.

18 Quick Configuration Guide

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

CURRENT SETTING AND FIRST STEPS WITH STEALTHCHOP

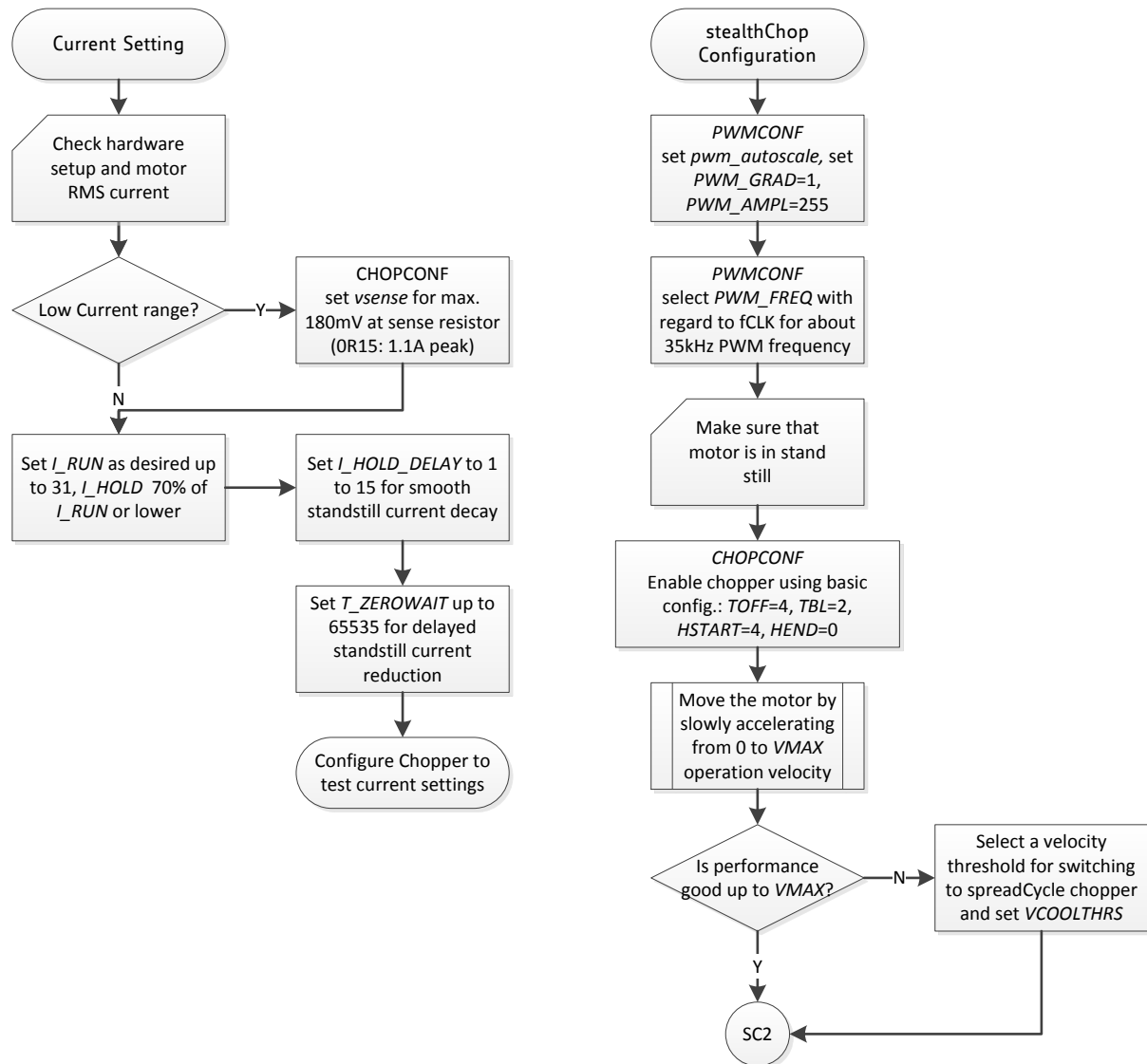


Figure 18.1 Current setting and first steps with stealthChop

TUNING STEALTHCHOP AND SPREADCYCLE

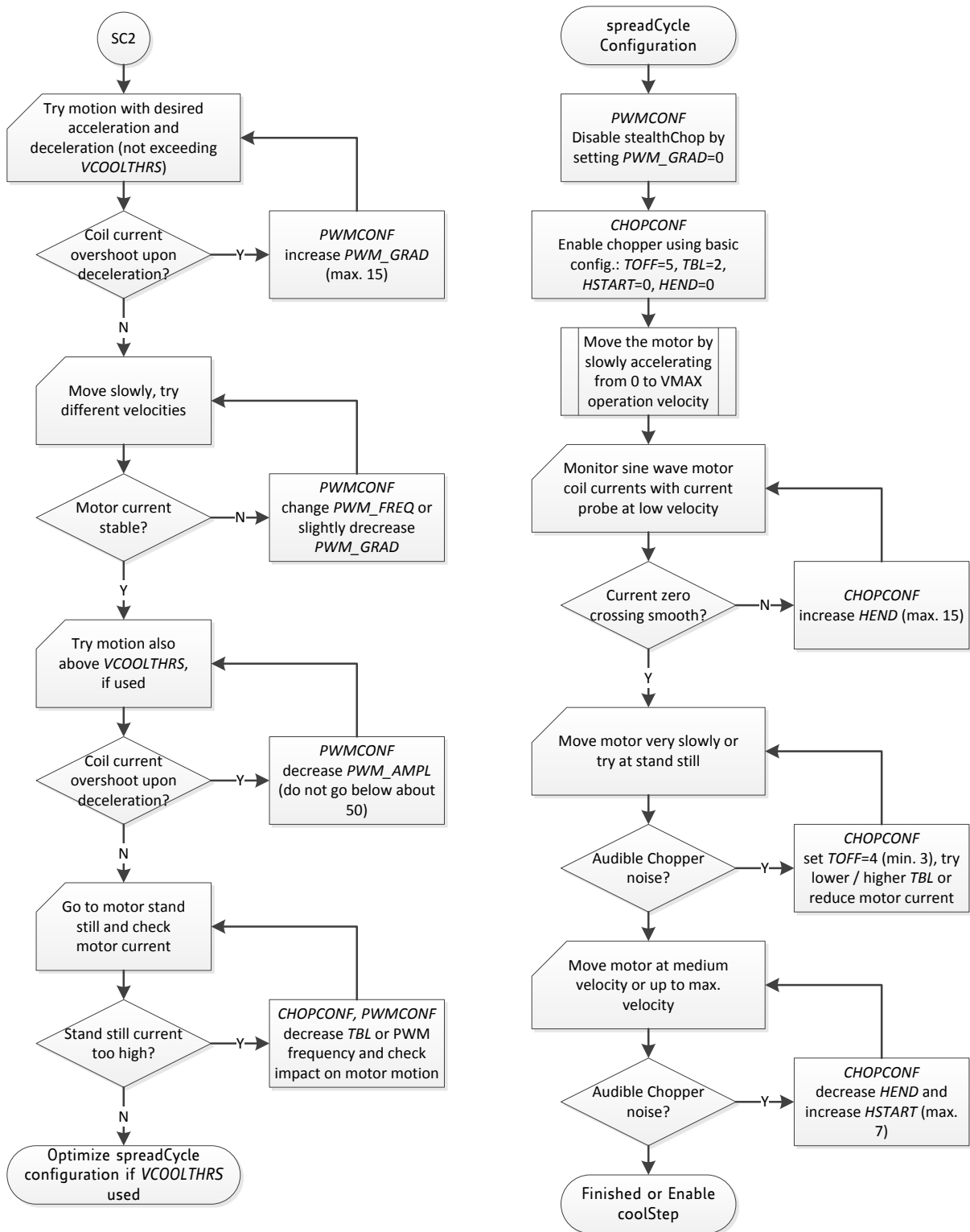


Figure 18.2 Tuning stealthChop and spreadCycle

MOVING THE MOTOR USING THE MOTION CONTROLLER

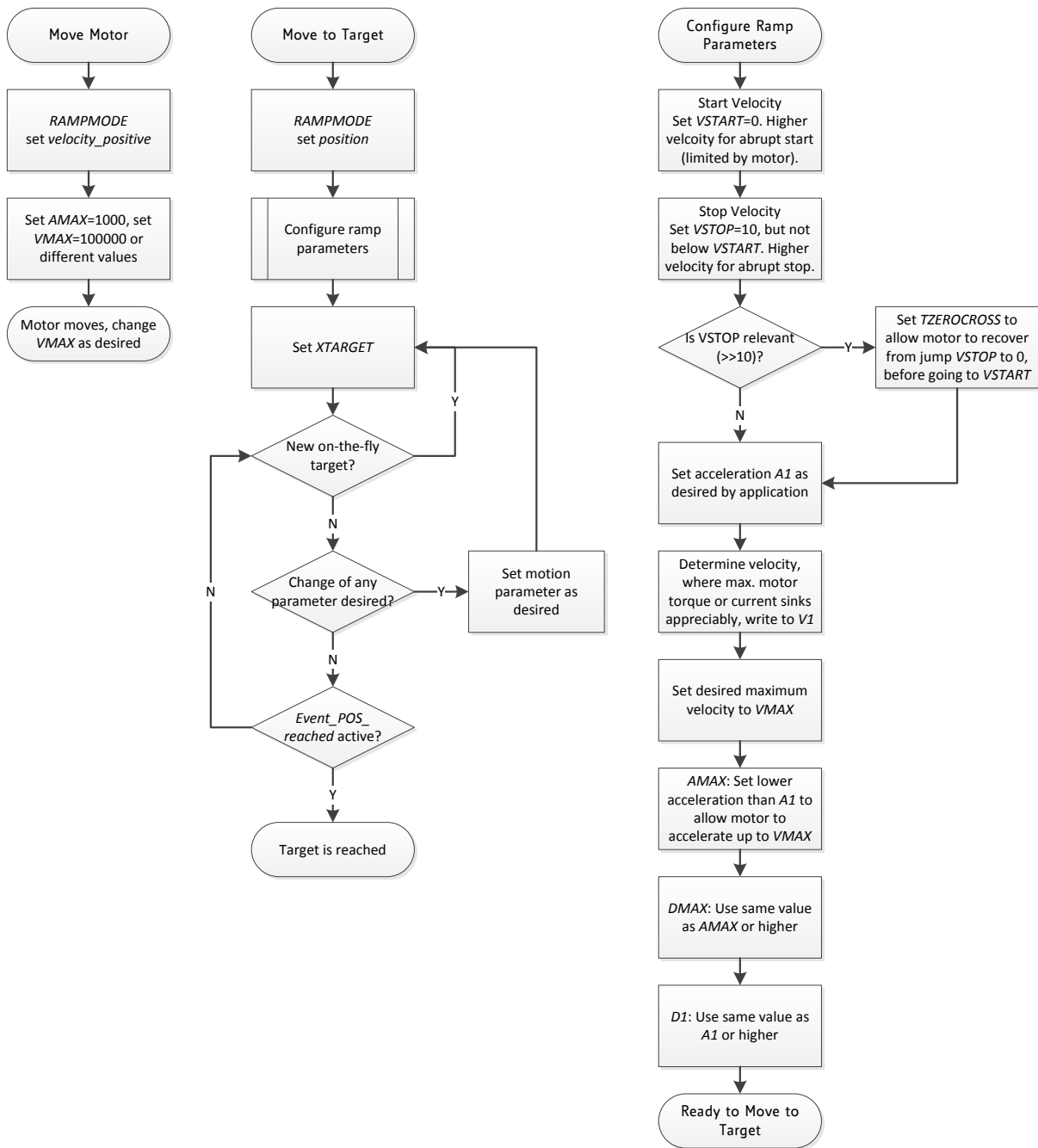


Figure 18.3 Moving the motor using the motion controller

ENABLING COOLSTEP (ONLY IN COMBINATION WITH SPREADCYCLE)

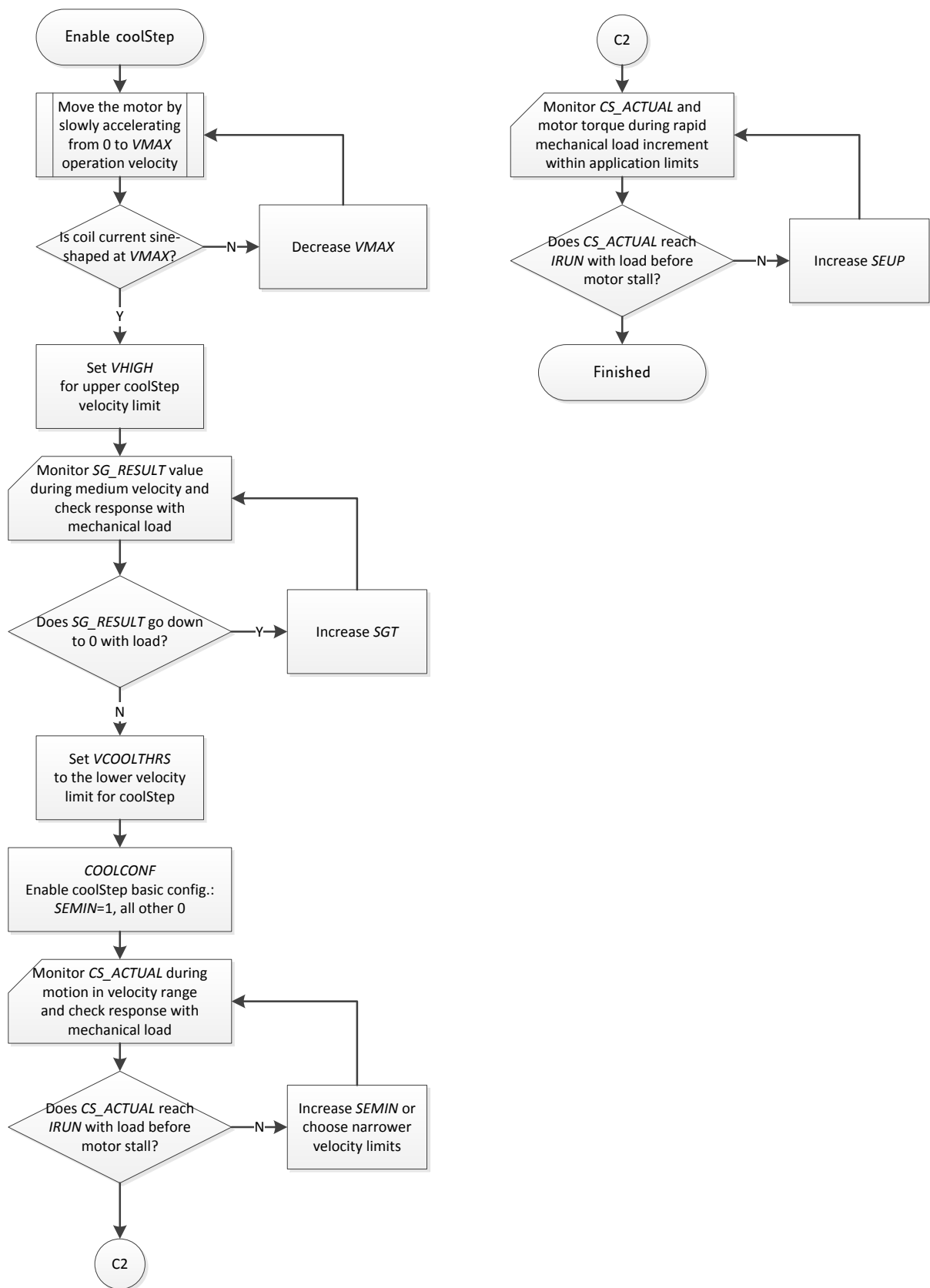


Figure 18.4 Enabling coolStep (only in combination with spreadCycle)

SETTING UP DCSTEP

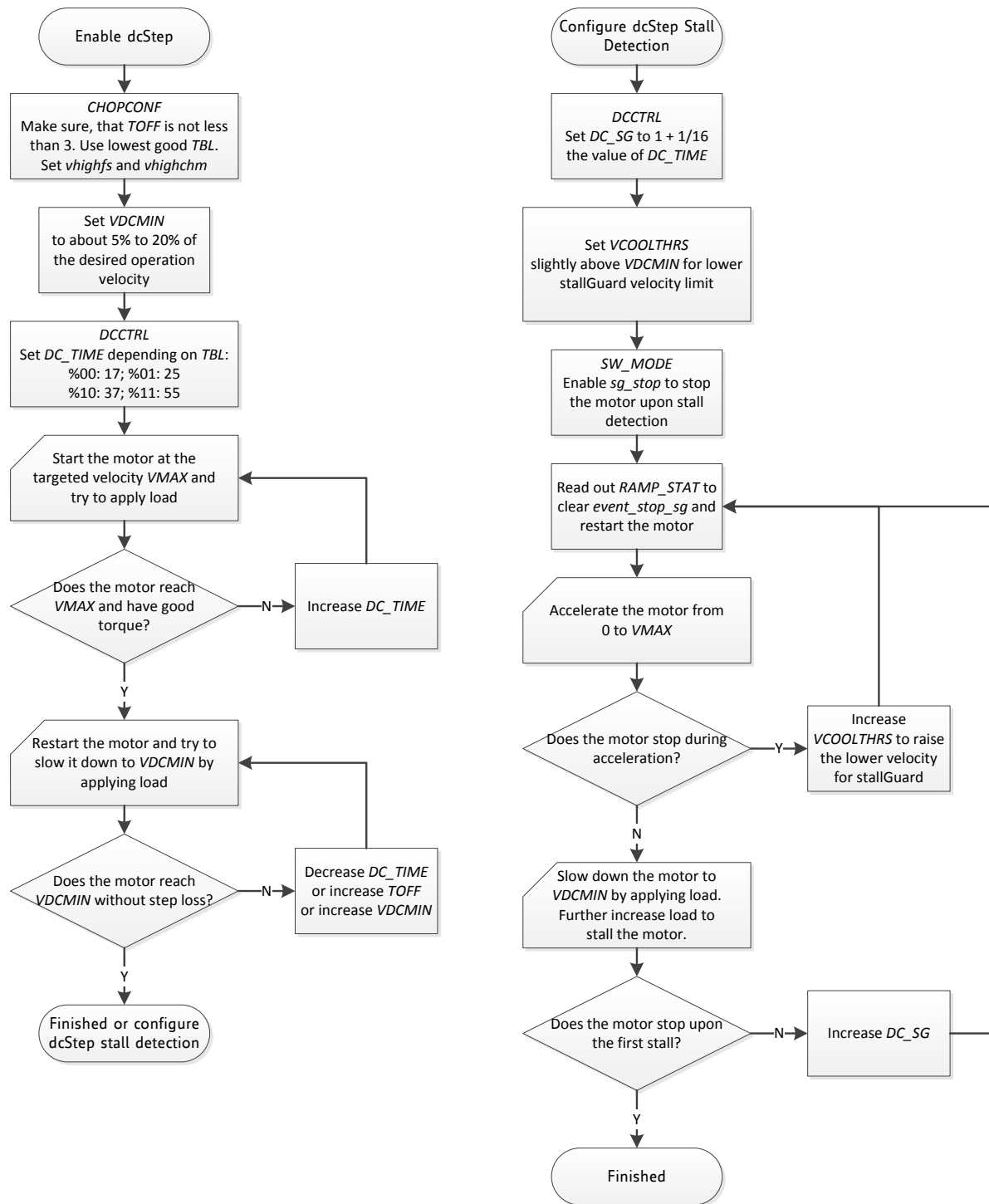


Figure 18.5 Setting up dcStep

19 Getting Started

Please refer to the TMC5072 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 18 will guide you through the process of correctly setting up all registers.

19.1 Initialization Examples

Initialization SPI datagram example sequence to enable driver 1 for step and direction operation and initialize the chopper for a 2 phase motor:

```
SPI send: 0xEC000100C5; // CHOPCONF: TOFF=5, HSTRT=4, HEND=1, TBL=2, CHM=0 (spreadCycle)
SPI send: 0xB000061F05; // IHOLD_IRUN: IHOLD=5, IRUN=31 (max. current), IHOLDDELAY=6
SPI send: 0xAC00002710; // TZEROWAIT=10000
SPI send: 0x8000000006; // GCONF=6: Switch both drivers to step and direction operation
```

SPI datagram example sequence to enable and initialize driver 1 for spreadCycle operation combined with stealthChop at low velocities. Ramp generator 1 moves the motor in velocity mode. Additional read access to the position register:

```
SPI send: 0x8000000008; // GCONF=8: Enable PP and INT outputs
SPI send: 0xEC000100C5; // CHOPCONF: TOFF=5, HSTRT=4, HEND=1, TBL=2, CHM=0 (spreadCycle)
SPI send: 0xB000011F05; // IHOLD_IRUN: IHOLD=5, IRUN=31 (max. current), IHOLDDELAY=1
SPI send: 0x90000401C8; // PWM_CONF: AUTO=1, 2/1024 Fclk, Switch amplitude limit=200, Grad=1
SPI send: 0xB200061A80; // VHIGH=400 000: Set VHIGH to a high value to allow stealthChop
SPI send: 0xB100007530; // VCOOLTHRS=30000: Set upper limit for stealthChop to about 30RPM
SPI send: 0xA600001388; // AMAX=5000
SPI send: 0xA700004E20; // VMAX=20000
SPI send: 0xA000000001; // RAMPMODE=1 (positive velocity)
```

// Now motor 1 should start rotating

```
SPI send: 0x2100000000; // Query X Actual – The next read access delivers X Actual
SPI read; // Read X Actual
```

Initialization SPI datagram example sequence to enable and initialize the motion controller and then move one rotation (51200 microsteps) using the ramp generator.

```
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: 0xA7000304D0; // 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))
```

For UART based operation it is important to make sure that the CRC byte is correct. The following example shows initialization for a TMC5072 with slave address 1 (NEXTADDR pin high). It programs driver 1 to spreadCycle mode and ramp generator 1 to move the motor in velocity mode and 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 0xB0 0x00 0x01 0x14 0x05 0x57; // 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 1 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.

20 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.

21 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.

21.1 Using the Internal Clock

Directly tie the CLK input to GND near to the TMC5072 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 14.5. This allows scaling acceleration and velocity settings as a result. The temperature dependency and ageing of the internal clock is comparatively low.

IMPLEMENTING FREQUENCY DEPENDENT SCALING

Frequency dependent scaling allows using the internal clock for a motion control application. The time reference of the external microcontroller is used to calculate a scaler for all velocity settings. The following steps are required:

1. You may leave the motor driver disabled during the calibration.
2. Start motor in velocity mode, with $V_{MAX}=10000$ and $A_{MAX}=60000$ (for quick acceleration). The acceleration phase is ended after a few ms.
3. Read out X_{ACTUAL} twice, at time point t_1 and time point t_2 , e.g. 100ms later ($dt=0.1s$). The time difference between both read accesses shall be exactly timed by the external microcontroller.
4. Stop the motion ramp by setting $V_{MAX}=0$.
5. The number of steps done in between of t_1 and t_2 now can be used to calculate the factor

$$f = \frac{V_{MAX} * dt}{X_{ACTUAL}(t_2) - X_{ACTUAL}(t_1)} = \frac{1000}{X_{ACTUAL}(t_2) - X_{ACTUAL}(t_1)}$$
6. Now multiply each velocity value with this factor f , to normalize the velocity to steps per second. At a nominal value of the internal clock frequency, 780 steps will be done in 100ms.

Hint

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

21.2 Using an External Clock

When an external clock is available, a frequency of 10MHz to 16MHz 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 18MHz 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.

21.3 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 TMC5072 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 can be considered.

22 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	27	V
Supply and bridge voltage max. *)	V_{VS}	-0.5	28	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}		2.0	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 (tbd.)	kV
ESD-Protection for handling (Human body model, HBM)	V_{ESD}		1 (tbd.)	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.

23 Electrical Characteristics

23.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	T_J	-40	125	°C
Supply voltage (using internal +5V regulator)	V_{VS}	5.5	26	V
Supply voltage (internal +5V regulator bridged: $V_{VCC}=V_{VSA}$)	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}		0.8	A
Peak output current per motor coil output (sine wave peak)	I_{Ox}		1.1	A
Peak output current per motor coil output (sine wave peak) Limit $T_J \leq 105^\circ\text{C}$, e.g. for 100ms short time acceleration phase below 50% duty cycle.	I_{Ox}		1.5	A

23.2 DC Characteristics 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				
		$V_{VS} = 24.0V$				
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Total supply current, driver disabled $I_{VS} + I_{VSA} + I_{VCC}$	I_S	$f_{CLK}=16MHz$		25	40	mA
Total supply current, operating, $I_{VS} + I_{VSA} + I_{VCC}$	I_S	$f_{CLK}=16MHz, 40kHz$ chopper		28		mA
Static supply current	I_{VS0}	$f_{CLK}=0Hz$	3	4.5	7	mA
Supply current, driver disabled, dependency on CLK frequency	I_{VSX}	f_{CLK} variable, additional to I_{VS0}		1.3		mA/MHz
Internal current consumption from 5V supply on VCC pin	I_{VCC}	$f_{CLK}=16MHz, 40kHz$ chopper		25	40	mA
IO supply current	I_{VIO}	no load on outputs, inputs at V_{IO} or GND		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	50	120	250	ns
slope, MOSFET turning off	t_{SLPOFF}	measured at 700mA load current	50	220	450	ns
Current sourcing, driver off	I_{IDLE}	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	$V_{SVOUT} - 0.4$	V_{SVOUT}	V
Charge pump voltage threshold for undervoltage detection	$V_{VCP}-V_{VS}$	using internal 5V regulator voltage	3.1	3.6	3.9	V
Charge pump frequency	f_{CP}			1/16 f_{CLKOSC}		

Linear regulator		DC-Characteristics				
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Output voltage	V_{SVOUT}	$I_{SVOUT} = 0mA$ $T_J = 25^\circ C$	4.75	5.0	5.25	V
Output resistance	R_{SVOUT}	Static load		3		Ω
Deviation of output voltage over the full temperature range	$V_{SVOUT(DEV)}$	$I_{SVOUT} = 30mA$ $T_J = \text{full range}$		30	100	mV

Clock oscillator and input	Timing-Characteristics					
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Clock oscillator frequency	f_{CLKOSC}	$t_j = -50^\circ C$	9	12.4		MHz
Clock oscillator frequency	f_{CLKOSC}	$t_j = 50^\circ C$	10.1	13.2	17.2	MHz
Clock oscillator frequency	f_{CLKOSC}	$t_j = 150^\circ C$		13.4	18	MHz
External clock frequency (operating)	f_{CLK}		4	10-16	18	MHz
External clock high / low level time	t_{CLKL} / t_{CLKH}	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_{VS} 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	1.9	2.55	3.0	V
V_{VCC_IO} undervoltage detector hysteresis	$V_{UV_VIOHYST}$		0.1	0.3	0.5	V
Short to GND detector threshold ($V_{VSP} - V_{OX}$)	V_{OS2G}		1.5	2.2	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	$^\circ C$
Overtemperature shutdown	t_{OT}	Temperature rising	135	150	170	$^\circ C$

Sense resistor voltage levels	DC-Characteristics					
Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Sense input peak threshold voltage (low sensitivity)	V_{SRTL}	$vsense=0$ $csactual=31$ $sin_x=248$ $Hyst.=0; I_{BRxy}=0$		320		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	I_{COIL}	$vsense=0$	-5		+5	%
Internal resistance from pin BRxy to internal sense comparator (additional to sense resistor)	R_{BRxy}			20		$m\Omega$

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_{OUTH}	$I_{OUTH} = -2mA$	$V_{VIO}-0.2$			V
Input leakage current	I_{LEAK}		-10		10	μA
Digital pin capacitance	C			3.5		pF

23.3 Thermal Characteristics

The following table shall give an idea on the thermal resistance of the QFN-48 package. The thermal resistance for a four layer board will provide a good idea on a typical application. The single layer board example is kind of a worst case condition, as the typical application will require a 4 layer board. Actual thermal characteristics will depend on the PCB layout, PCB type and PCB size.

A thermal resistance of 23°C/W for a typical board means, that the package is capable of continuously dissipating 4W at an ambient temperature of 25°C with the die temperature staying below 125°C.

Parameter	Symbol	Conditions	Typ	Unit
Typical power dissipation One motor active, one motor in standby at low current	P_D	One motor 1.00A RMS 115°C (125°C)	3.7	W
		One motor 0.71A RMS 85°C (93°C)	2.4	W
		Surface temperature at package center (peak surface temperature), board 55mm x 85mm, 25°C environment stealthChop or spreadCycle, sinewave, 40 or 20kHz chopper, 24V, 16MHz, internal supply for VCC Motors: QSH4218-035-10-027		
Typical power dissipation Two motors active	P_D	Two motors 0.71A RMS 113°C (119°C)	3.7	W
		Two motors 0.35A RMS 64°C (68°C)	1.4	W
Thermal resistance junction to ambient on a single layer board	R_{TJA}	Single signal layer board (1s) as defined in JEDEC EIA JESD51-3 (FR4, 76.2mm x 114.3mm, d=1.6mm)	80	K/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, 76.2mm x 114.3mm, d=1.6mm)	23	K/W
Thermal resistance junction to ambient on a multilayer board with air flow	R_{TMJA1}	Identical to R_{TMJA} , but with air flow 1m/s	20	K/W
Thermal resistance junction to board	R_{TJB}	PCB temperature measured within 1mm distance to the package	10	K/W
Thermal resistance junction to case	R_{TJC}	Junction temperature to heat slug of package	3	K/W

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 TMC5072 power dissipation is available on www.trinamic.com.

24 Layout Considerations

24.1 Exposed Die Pad

The TMC5072 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.

24.2 Wiring GND

All signals of the TMC5072 are referenced to their respective GND. Directly connect all GND pins under the TMC5072 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 TMC5072. Optimally place them close to the TMC5072, 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.

24.3 Supply Filtering

The 5VOUT output voltage ceramic filtering capacitor (4.7 μ F recommended) should be placed as close as possible to the 5VOUT pin, with its GND return going directly to the GNDA pin. Use as short and as thick connections as possible. For best microstepping performance and lowest chopper noise an additional filtering capacitor can 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 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 μ F 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.

24.4 Single Driver Connection

In a parallel connection setup, where the TMC5072 drives one motor with double current, take into account, that driver 1 takes over the complete control. Thus, the driver 1 layout should be optimized concerning sense resistor placement, etc. Connect driver 2 bridge outputs and BR pins in parallel to the corresponding driver 1 pins. Especially for the BR pins of driver 2, it is important to use low inductivity interconnection lines to driver 1.

24.5 Layout Example

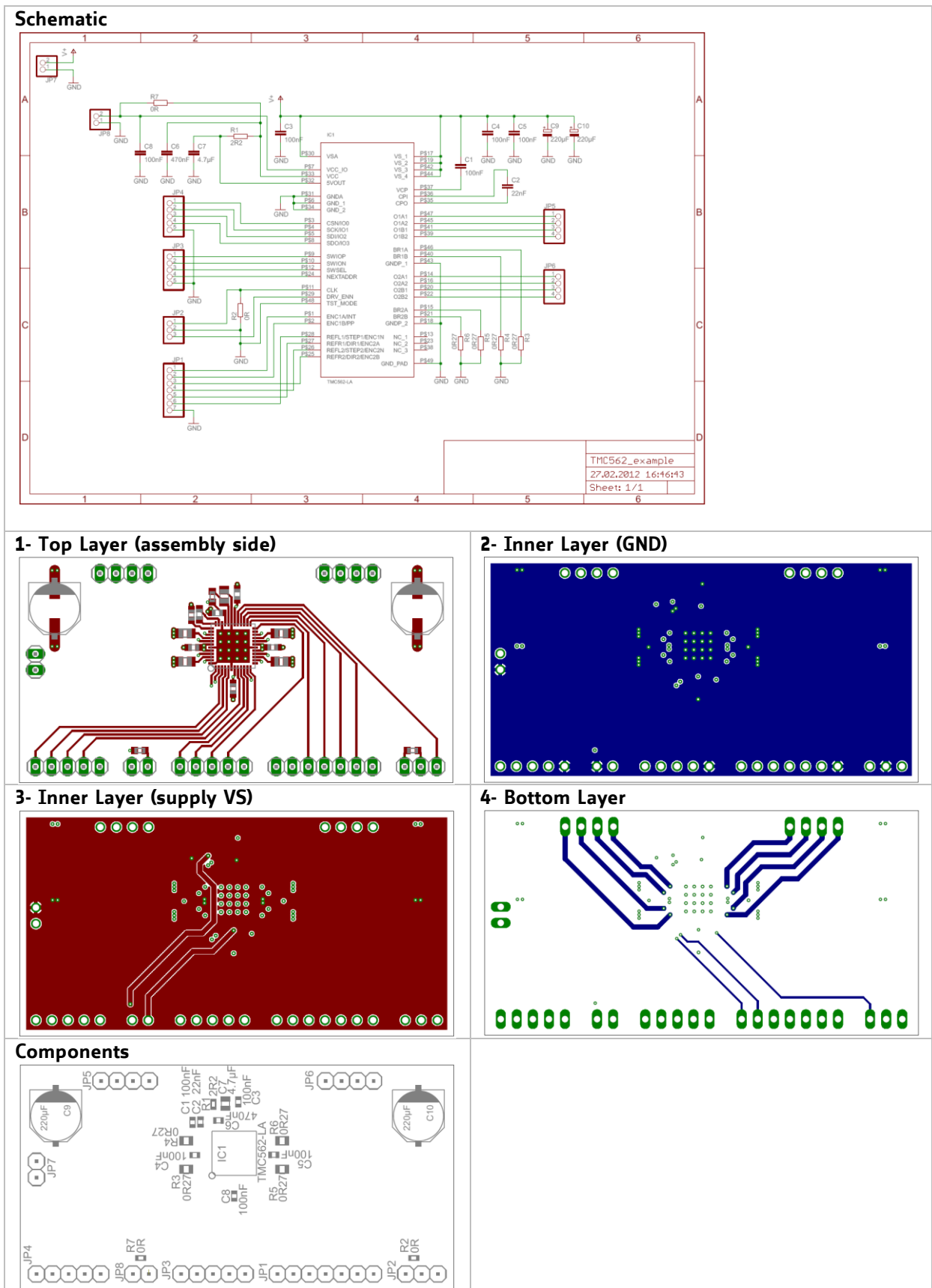


Figure 24.1 Layout example

25 Package Mechanical Data

25.1 Dimensional Drawings

Attention: Drawings not to scale.

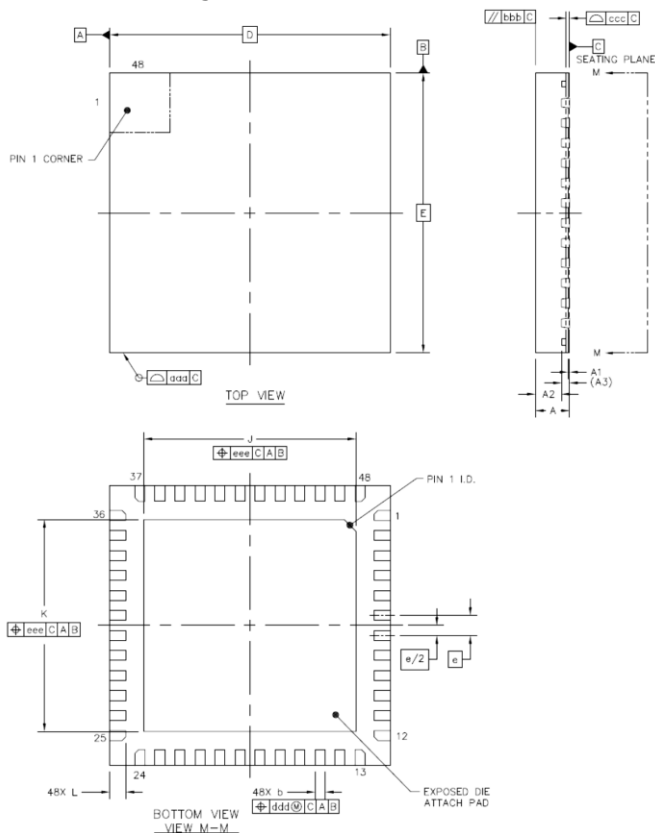


Figure 25.1 Dimensional drawings

Parameter	Ref	Min	Nom	Max
total thickness	A	0.80	0.85	0.90
stand off	A1	0.00	0.035	0.05
mold thickness	A2	-	0.65	0.67
lead frame thickness	A3		0.203	
lead width	b	0.2	0.25	0.3
body size X	D		7.0	
body size Y	E		7.0	
lead pitch	e		0.5	
exposed die pad size X	J	5.2	5.3	5.4
exposed die pad size Y	K	5.2	5.3	5.4
lead length	L	0.35	0.4	0.45
package edge tolerance	aaa			0.1
mold flatness	bbb			0.1
coplanarity	ccc			0.08
lead offset	ddd			0.1
exposed pad offset	eee			0.1

25.2 Package Codes

Type	Package	Temperature range	Code & marking
TMC5072-LA	QFN48 (RoHS)	-40°C ... +125°C	TMC5072-LA

26 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. Integration at street level cost was possible by squeezing know-how into a few mm² of layout using one of the most modern smart power processes. The 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

27 Disclaimer

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

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

Specifications are subject to change without notice.

All trademarks used are property of their respective owners.

28 ESD Sensitive Device

The TMC5072 is an ESD sensitive CMOS device sensitive to electrostatic discharge. Take special care to use adequate grounding of personnel and machines in manual handling. After soldering the devices to the board, ESD requirements are more relaxed. Failure to do so can result in defect or decreased reliability.



29 Table of Figures

Figure 1.1 Basic application and block diagram.....	5
Figure 1.2 Energy efficiency with coolStep (example).....	8
Figure 2.1 TMC5072 pin assignments.....	9
Figure 3.1 Standard application circuit.....	12
Figure 3.2 5V only operation.....	13
Figure 3.3 Driving a single motor with high current.....	14
Figure 3.4 Using an external 5V supply for digital circuitry of driver (different options).....	15
Figure 3.5 Using an external 5V supply to bypass internal regulator	15
Figure 3.6 RC-Filter on VCC for reduced ripple	16
Figure 3.7 Simple ESD enhancement and more elaborate motor output protection	17
Figure 4.1 SPI timing.....	20
Figure 5.1 Addressing multiple TMC5072 via single wire interface using chaining	24
Figure 5.2 Addressing multiple TMC5072 via differential interface, additional filtering for NEXTADDR ...	25
Figure 5.3 Ring Mode Example	26
Figure 8.1 Motor coil sine wave current with stealthChop (measured with current probe)	49
Figure 8.2 Scope shot: good setting for PWM_GRAD.....	50
Figure 8.3 Scope shot: too small setting for PWM_GRAD	50
Figure 8.4 Good and too small setting for PWM_GRAD.....	51
Figure 9.1 Chopper phases	57
Figure 9.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)	59
Figure 9.3 spreadCycle chopper scheme showing coil current during a chopper cycle	60
Figure 9.4 Classic const. off time chopper with offset showing coil current.....	61
Figure 9.5 Zero crossing with classic chopper and correction using sine wave offset.....	61
Figure 11.1 Ramp generator velocity trace showing consequent move in negative direction.....	65
Figure 11.2 Illustration of optimized motor torque usage with TMC5072 ramp generator.....	66
Figure 11.3 Ramp generator velocity dependent motor control.....	67
Figure 11.4 Using reference switches (example).....	68
Figure 12.1 Function principle of stallGuard2	70
Figure 12.2 Example: Optimum SGT setting and stallGuard2 reading with an example motor	72
Figure 13.1 coolStep adapts motor current to the load	75
Figure 14.1 dcStep extended application operation area	77
Figure 14.2 Velocity profile with impact by overload situation	78
Figure 15.1 LUT programming example	80
Figure 16.1 STEP and DIR timing, Input pin filter	82
Figure 16.2 microPlyer microstep interpolation with rising STEP frequency.....	84
Figure 17.1 Outline of ABN signals of an incremental encoder	85
Figure 18.1 Current setting and first steps with stealthChop	88
Figure 18.2 Tuning stealthChop and spreadCycle	89
Figure 18.3 Moving the motor using the motion controller	90
Figure 18.4 Enabling coolStep (only in combination with spreadCycle).....	91
Figure 18.5 Setting up dcStep.....	92
Figure 24.1 Layout example.....	103
Figure 25.1 Dimensional drawings.....	104

30 Revision History

Version	Date	Author BD – Bernhard Dwersteg SD – Sonja Dwersteg	Description
1.00	2012-JUL-20	SD	New design (based on V2 silicon)
1.10	2013-DEC-01	BD	Revision for V3 silicon - 8 Bit addressing for UART, Ring option - no three phase option - single sinewave table for both motors: Changed table for motor driver register (common microstep table) - additional stealthChop option for 2 phase chopper - blanking of stallGuard - dcCoolStep - UART ring option: IO3 as config input, Address=0 selects no forwarding - Sense resistor table
	2014-AUG-26	BD	Silicon V3.0A
1.12	2014-OCT-15	BD	- Second stallGuard tuning algorithm - No 3 Phase chopper - Adapted hint for optimum chopper frequency - Added Parts from Rhino Documentation V0.47
1.18	2015-FEB-24	BD	stallGuard Stop details: Improved homing algorithm in 14.4, Added 12.4, Text for event_stop_sg, improved 19.1, Limits VCC_IO UV, VCP UV, Detail wording in many chapters, 320mV VSRTL, SPI example, Added chapter Closing the Loop. Added UART interface errata. Explanation VACTUAL sign, improved blue blocks, dcStep Description & Procedure
1.19	2015-MAR-25	BD	Removed preliminary, slight corrections in wording
1.20	2015-OCT-13	BD	Correct SPI write access example, SPI mode 3, added TCLK1 data, corrected TOFF calculation example, comments in GSTAT, comment on SPI_STATUS, 5V only +-5%, X1=128 in microstep table defaults
1.21	2016-APR-22	BD	More details on: Setting negative encoder factors, stealthChop lower current limit, Ramp generator Joystick control, Terminate Ramp, Adaptation to internal fCLK, Interrupt handling Corrected: effective stealthChop PWM frequency is $2 \cdot \text{divider}$ setting, Wording V1 and VMAX register, ESD schematic w. varistors instead of snubber

Table 30.1 Documentation revisions

31 References

[TMC5072-EVAL] TMC5072-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™ with TMC5072, www.trinamic.com
Calculation sheet [TMC50XX_Calculations.xlsx](#)