TMC5062 DATASHEET

Dual, cost-effective controller and driver for up to two 2-phase bipolar stepper motors.
Integrated motion controller with SPI interface.

dcStep™

coolStep™

stallGuard2™

Applications

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

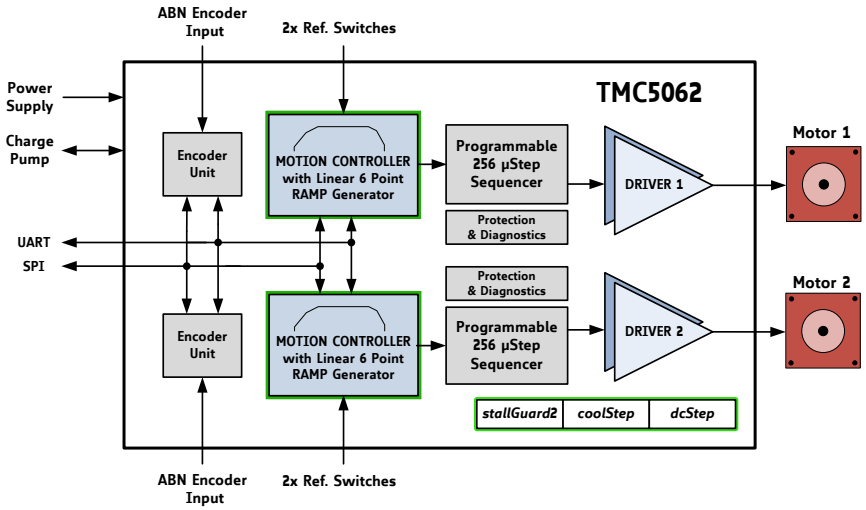
Features and Benefits

Two 2-phase stepper motors
Drive Capability up to 2 x 1.1A coil current
Motion Controller with **sixPoint™** ramp
Voltage Range 4.75... 20V DC
SPI & Single Wire UART
Dual ABN Encoder Interface
2x Ref.-Switch input per axis
Highest Resolution 256 microsteps per full step
Full Protection & Diagnostics
dcStep™ load dependent speed control – no step loss
stallGuard2™ high precision sensorless motor load detection
coolStep™ load dependent current saves up to 75% energy
spreadCycle™ high-precision chopper for best current sine wave form and zero crossing with additional **chopSync2™**
Compact Size 7x7mm² QFN48 package

Description

The TMC5062 is a high performance motion controller and driver for up to two stepper motors. It combines two flexible ramp motion controllers with energy efficient stepper motor drivers. The drivers support two-phase stepper motors and offer an industry-leading feature set, including high-resolution microstepping, sensorless mechanical load measurement, load-adaptive velocity and power optimization, and low-resonance chopper operation. Standard SPI™ interface and an optional UART based single wire interface simplify communication. Integrated protection and diagnostic features support robust and reliable operation. High integration, high energy efficiency and small form factor enable miniaturized designs with low external component count for cost-effective and highly competitive solutions.

Block Diagram



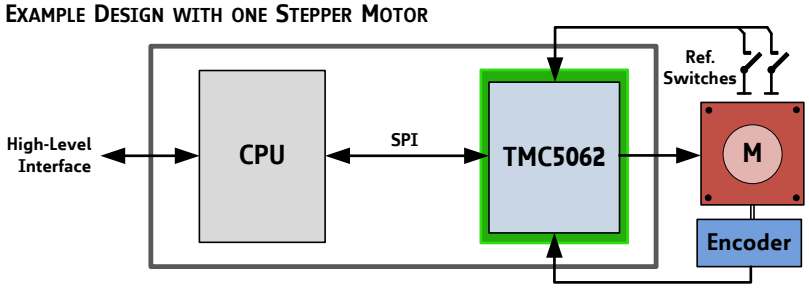
TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

TRINAMIC
MOTION CONTROL

APPLICATION EXAMPLES: HIGH FLEXIBILITY – MULTIPURPOSE USE

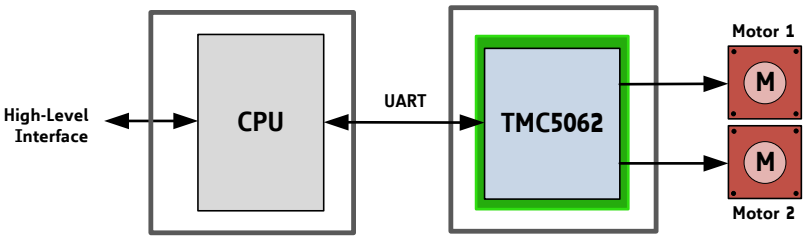
The TMC5062 scores with power density, complete motion controlling features and integrated power stages. It offers a versatility that covers a wide spectrum of applications from battery systems up to embedded applications with 1.1A RMS motor current per coil. The small form factor keeps costs down and 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 from TRINAMIC's coolStep and dcStep technologies deliver cost savings in related systems such as power supplies and cooling.

EXAMPLE DESIGN WITH ONE STEPPER MOTOR

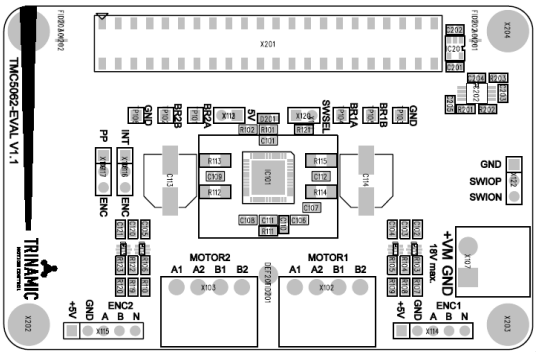


The stepper motor driver outputs are switched in parallel. A dual ABN encoder interface and two reference switch inputs are used.

COMPACT DESIGN FOR TWO STEPPER MOTORS



An application with two stepper motors is shown. Additionally the ABN encoder interface and two reference switches can be used for each motor. A single CPU controls the whole system. The CPU-board and controller / driver boards are highly economical and space saving. A UART interface can be used as an option to SPI for pin count limited controller or remote drives.



TMC5062-EVAL EVALUATION BOARD

EVALUATION & DEVELOPMENT PLATFORM

The TMC5062-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 TMC5062 evaluation board system consists of three parts: STARTRAMPE (base board), ESELSBRÜCKE (connector board including several test points), and TMC5062-EVAL.

ORDER CODES

Order code	Description	Size [mm²]
TMC5062-LA	Dual dcStep™ and coolStep™ controller/driver, QFN48	7 x 7
TMC5062-EVAL	Evaluation board for TMC5062	85 x 55
STARTRAMPE	Baseboard for TMC5062-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

1.1 KEY CONCEPTS 5
1.2 CONTROL INTERFACES 6
1.3 SOFTWARE 6
1.4 MOVING AND CONTROLLING THE MOTOR 7
1.5 PRECISION DRIVER WITH PROGRAMMABLE MICROSTEPPING WAVE 7
1.6 STALLGUARD2 – MECHANICAL LOAD SENSING 7
1.7 COOLSTEP – LOAD ADAPTIVE CURRENT CONTROL 7
1.8 DCSTEP – LOAD DEPENDENT SPEED CONTROL 8
1.9 ENCODER INTERFACES 8

2 PIN ASSIGNMENTS 9

2.1 PACKAGE OUTLINE 9
2.2 SIGNAL DESCRIPTIONS 9

3 SAMPLE CIRCUITS 12

3.1 STANDARD APPLICATION CIRCUIT 12
3.2 5 V ONLY SUPPLY 14
3.3 EXTERNAL VCC SUPPLY 15
3.4 OPTIMIZING ANALOG PRECISION 16
3.5 DRIVER PROTECTION AND EME CIRCUITRY 17

4 SPI INTERFACE 18

4.1 SPI DATAGRAM STRUCTURE 18
4.2 SPI SIGNALS 19
4.3 TIMING 20

5 UART SINGLE WIRE INTERFACE 21

5.1 DATAGRAM STRUCTURE 21
5.2 CRC CALCULATION 23
5.3 UART SIGNALS 24

6 REGISTER MAPPING 25

6.1 GENERAL CONFIGURATION REGISTERS 26
6.2 RAMP GENERATOR REGISTERS 28
6.3 ENCODER REGISTERS 34
6.4 MOTOR DRIVER REGISTERS 36

7 CURRENT SETTING 43

7.1 SENSE RESISTORS 44

8 CHOPPER OPERATION 45

8.1 SPREADCYCLE CHOPPER 46
8.2 CLASSIC 2-PHASE MOTOR CONSTANT OFF TIME CHOPPER 49
8.3 RANDOM OFF TIME 50
8.4 CHOPSYNC2 FOR QUIET MOTORS 51

9 DRIVER DIAGNOSTIC FLAGS 52

9.1 TEMPERATURE MEASUREMENT 52
9.2 SHORT TO GND PROTECTION 52
9.3 OPEN LOAD DIAGNOSTICS 52

10 RAMP GENERATOR 53

10.1 REAL WORLD UNIT CONVERSION 53
10.2 MOTION PROFILES 54
10.3 INTERRUPT HANDLING 55
10.4 VELOCITY THRESHOLDS 55
10.5 REFERENCE SWITCHES 57
10.6 RESTRICTIONS OF RAMP GENERATOR (ERRATA) 58

11 STALLGUARD2 LOAD MEASUREMENT 61

11.1 TUNING THE STALLGUARD2 THRESHOLD SGT 62
11.2 STALLGUARD2 UPDATE RATE AND FILTER 64
11.3 DETECTING A MOTOR STALL 64
11.4 HOMING WITH STALLGUARD 64
11.5 LIMITS OF STALLGUARD2 OPERATION 64

12 COOLSTEP OPERATION 65

12.1 USER BENEFITS 65
12.2 SETTING UP FOR COOLSTEP 65
12.3 TUNING COOLSTEP 67

13 DCSTEP 68

13.1 USER BENEFITS 68
13.2 DESIGNING-IN DCSTEP 68
13.3 ENABLING DCSTEP 69
13.4 STALL DETECTION IN DCSTEP MODE 69
13.5 MEASURING ACTUAL MOTOR VELOCITY IN DCSTEP OPERATION 70

14 SINE-WAVE LOOK-UP TABLE 71

14.1 USER BENEFITS 71
14.2 MICROSTEP TABLE 71

15 ABN INCREMENTAL ENCODER INTERFACE 73

15.1 ENCODER TIMING 74
15.2 SETTING THE ENCODER TO MATCH MOTOR RESOLUTION 74
15.3 CLOSING THE LOOP 74

16 QUICK CONFIGURATION GUIDE 76

17 GETTING STARTED 80

17.1 INITIALIZATION EXAMPLES 80

18 CLOCK OSCILLATOR AND CLOCK INPUT 81

18.1 USING THE INTERNAL CLOCK 81
18.2 USING AN EXTERNAL CLOCK 81
18.3 CONSIDERATIONS ON THE FREQUENCY 81

19 ABSOLUTE MAXIMUM RATINGS 83

20 ELECTRICAL CHARACTERISTICS 83

20.1 OPERATIONAL RANGE 83
20.2 DC CHARACTERISTICS AND TIMING CHARACTERISTICS 84
20.3 THERMAL CHARACTERISTICS 86

21 LAYOUT CONSIDERATIONS 87

21.1 EXPOSED DIE PAD 87
21.2 WIRING GND 87
21.3 SUPPLY FILTERING 87

21.4 LAYOUT EXAMPLE 88
22 PACKAGE MECHANICAL DATA 89
22.1 DIMENSIONAL DRAWINGS 89
22.2 PACKAGE CODES 89
23 DISCLAIMER 90
24 ESD SENSITIVE DEVICE 90
25 TABLE OF FIGURES 91
26 REVISION HISTORY 92
27 REFERENCES 92

1 Principles of Operation

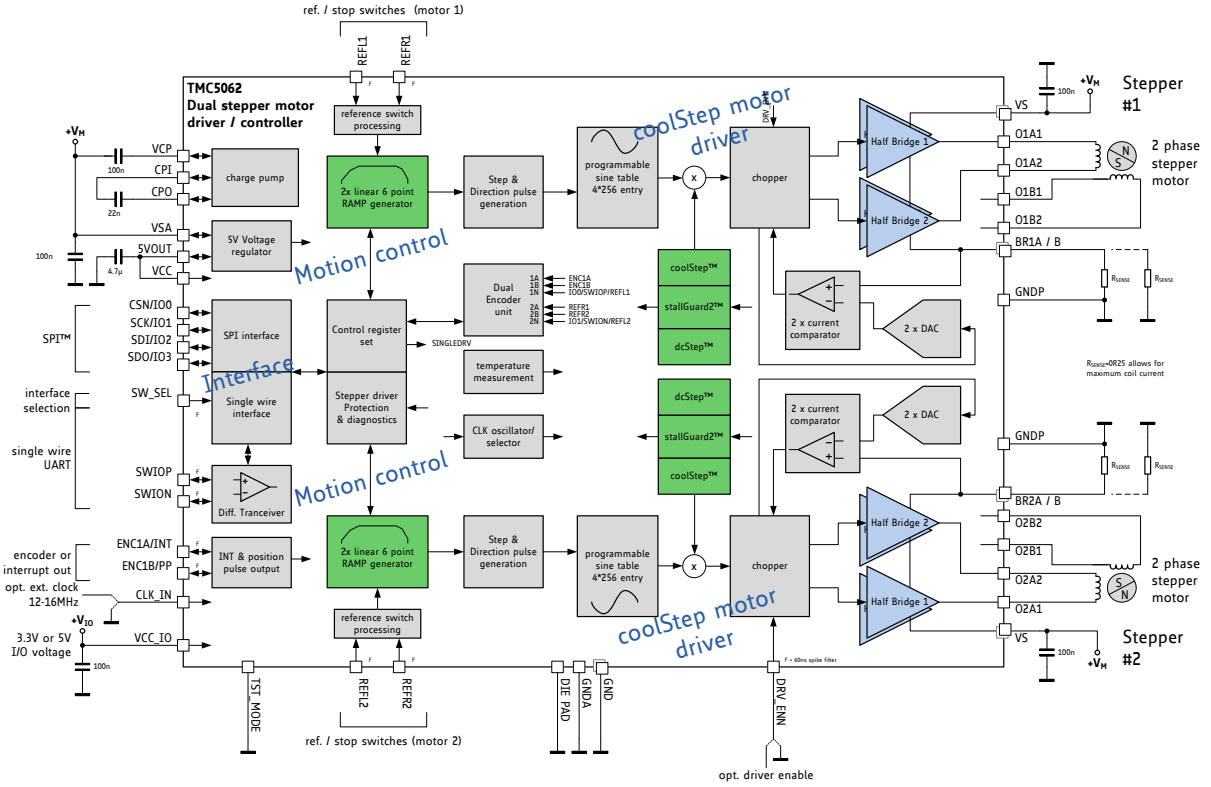


Figure 1.1 Basic application and block diagram

The TMC5062 motion controller and driver chip is an intelligent power component interfacing between the CPU and up to two stepper motors. All stepper motor logic is completely within the TMC5062. No software is required to control the motor – just provide target positions. The TMC5062 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 TMC5062 uses dcStep, coolStep, and stallGuard2 automatically to optimize every motor movement: TRINAMICs special features contribute toward lower system cost, greater precision, greater energy efficiency, smoother motion, and cooler operation in stepper motor applications. The clear concept and the comprehensive solution save design-in time.

1.1 Key Concepts

The TMC5062 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.

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 also 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 TMC5062 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 TMC5062 slave always consists of sending one 40-bit command word and receiving one 40-bit status word.

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

1.2.2 UART Interface

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

1.3 Software

From a software point of view the TMC5062 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 motors 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

1.5 Precision Driver with Programmable Microstepping Wave

Current into the motor coils is controlled using a cycle-by-cycle chopper mode. Two chopper modes are available: a traditional constant off-time mode and the new spreadCycle mode. Constant off-time mode provides higher torque at the 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. Programmable microstep shapes allow optimizing the motor performance.

Benefits are:

- Significantly improved microstepping with low cost motors
- Motor runs smooth and quiet
- 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, making the drive an efficient and precise solution.

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.

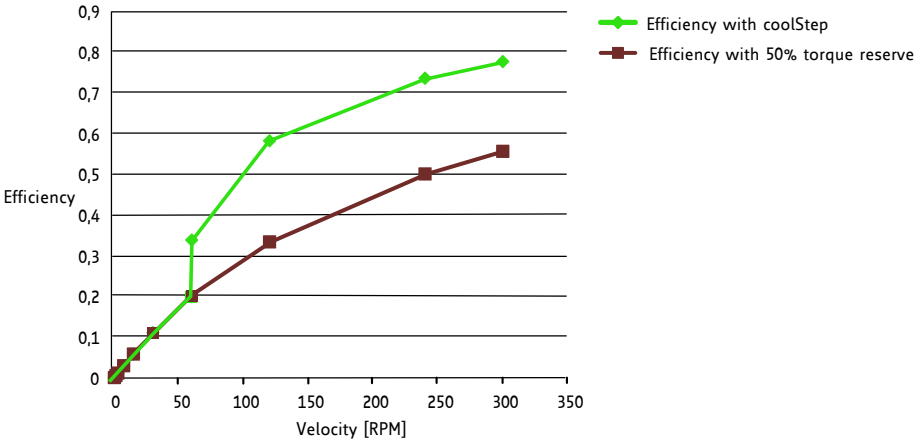


Figure 1.2 Energy efficiency with coolStep (example)

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 loose steps in overload conditions
- Application works as fast as possible
- Highest possible acceleration automatically
- Highest energy efficiency at speed limit
- Highest possible motor torque using fullstep drive
- Cheaper motor does the job

1.9 Encoder 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

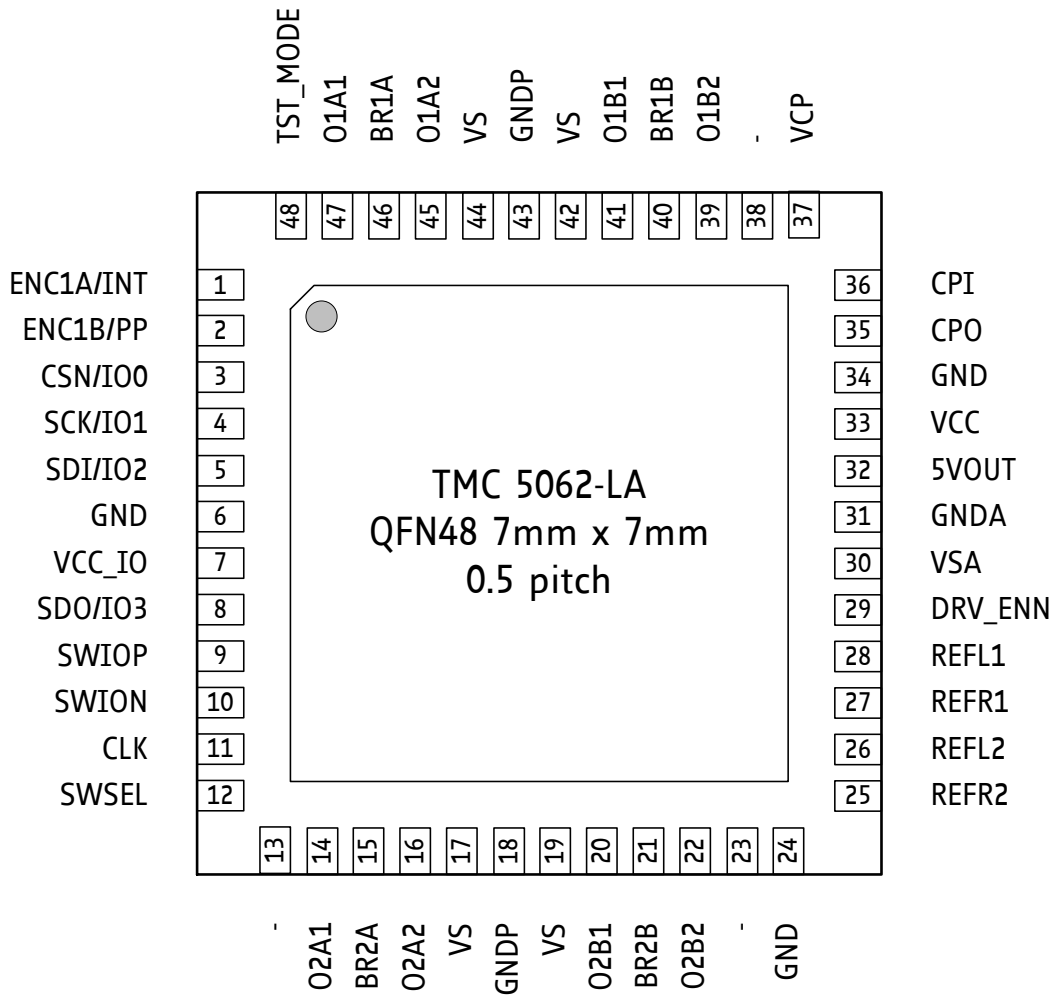


Figure 2.1 TMC5062 pin assignments.

2.2 Signal Descriptions

Pin	Number	Type	Function
GND	6, 24, 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 2.2µF 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.
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 22 nF / 50 V capacitor to CPO.
VCP	37		Output of charge pump. Provide external 100 nF 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 positive active interrupt output based on ramp generator flags *RAMP_STAT* bits 4, 5, 6 & 7 and encoder null event status *ENC_STATUS* bit 0 (*poscmp_enable*=1).
ENC1B/PP	2	I/O	Input B for incremental encoder 1. Can be programmed to provide position compare output for motor 1 (*poscmp_enable*=1).
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/IO3	8	I/O	Data output of SPI interface (Tristate, enabled with CSN=0), programmable IO in UART mode
SWIOP (ENC1N)	9	I/O	Single wire UART interface I/O. Has internal 100K pulldown resistor. Multi-purpose input in SPI mode or encoder 1 N input.
SWION (ENC2N)	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.
REFR2 (ENC2B)	25	I	Right reference switch input for motor 2 or encoder 2 B input
REFL2	26	I	Left reference switch input for motor 2
REFR1 (ENC2A)	27	I	Right reference switch input for motor 1 or encoder 2 A input
REFL1	28	I	Left reference switch input for 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.
-	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)

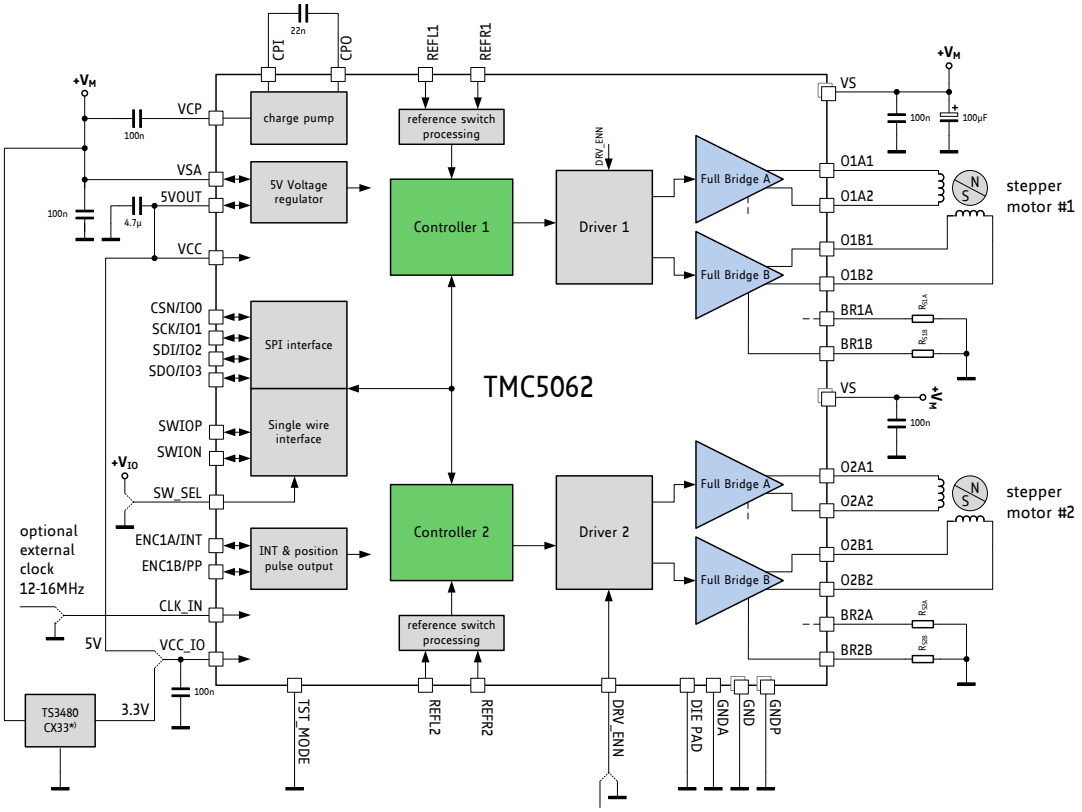
Pin	Number	Type	Function
O2A1	14	O (VS)	Motor 2 coil A output 1
BR2A	15		Sense resistor connection for motor 2 coil A. Place sense resistor to GND near pin.
O2A2	16	O (VS)	Motor 2 coil A output 2
VS	17, 19		Motor supply voltage. Provide filtering capacity near pin with shortest loop to nearest GNDP pin (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



*) For a reliable start-up it is essential that VCC_IO comes up to a minimum of 1.5V before the TMC5062 leaves the reset condition. Therefore, TRINAMIC recommends using a fast-start-up voltage regulator (e.g. TS3480CX33) in a 3.3V environment.

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 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 a fast startup 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 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 GNDA pin. See layout hints for more details. Low ESR electrolytic capacitors are recommended for VS filtering.

Attention
In case VSA is supplied by a different voltage source, make sure that VSA does not exceed VS by more than one diode drop upon power up or power down.

3.1.1 VCC_IO Requirements

For a reliable start-up it is essential that VCC_IO comes up to a minimum of 1.5V before the TMC5062 leaves the reset condition. The reset condition ends earliest 50µs after the time when VSA exceeds its undervoltage threshold of typically 4.2V, or when 5VOUT exceeds its undervoltage threshold of typically 3.5V, whichever comes last.

THERE ARE THREE WAYS TO COME UP TO VCC_IO REQUIREMENTS

- 5VOUT can be used directly to supply VCC_IO. In this case there are no further requirements.

- An external low drop regulator can be used in a 3.3V environment. Note, that most voltage regulators are not suitable for this application because they show a delayed boot up. The following external regulators are proven by TRINAMIC:

TS3480CX33	This regulator can be used within the full supply voltage range when tied to the motor supply voltage.
LD1117-3.3	This regulator can be used to supply VCC_IO from 5VOUT, or from a supply voltage of up to 15V.

- VCC_IO can be supplied externally as shown in Figure 3.2 . In this case it is mandatory to connect the Schottky diode to the logic supply of the external circuitry. Please note, that the 2K resistor is not to be used with 5V I/O voltage.

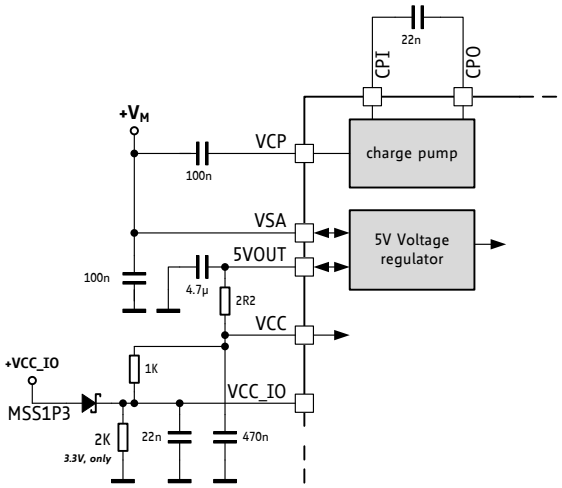


Figure 3.2 External supply of VCC_IO (showing optional filtering for VCC)

Refer to application note no. 028 *Supply Voltage Considerations: VCC_IO in TMC50xx Designs* (www.trinamic.com). Here you will find complete information about connecting VCC_IO.

3.2 5 V Only Supply

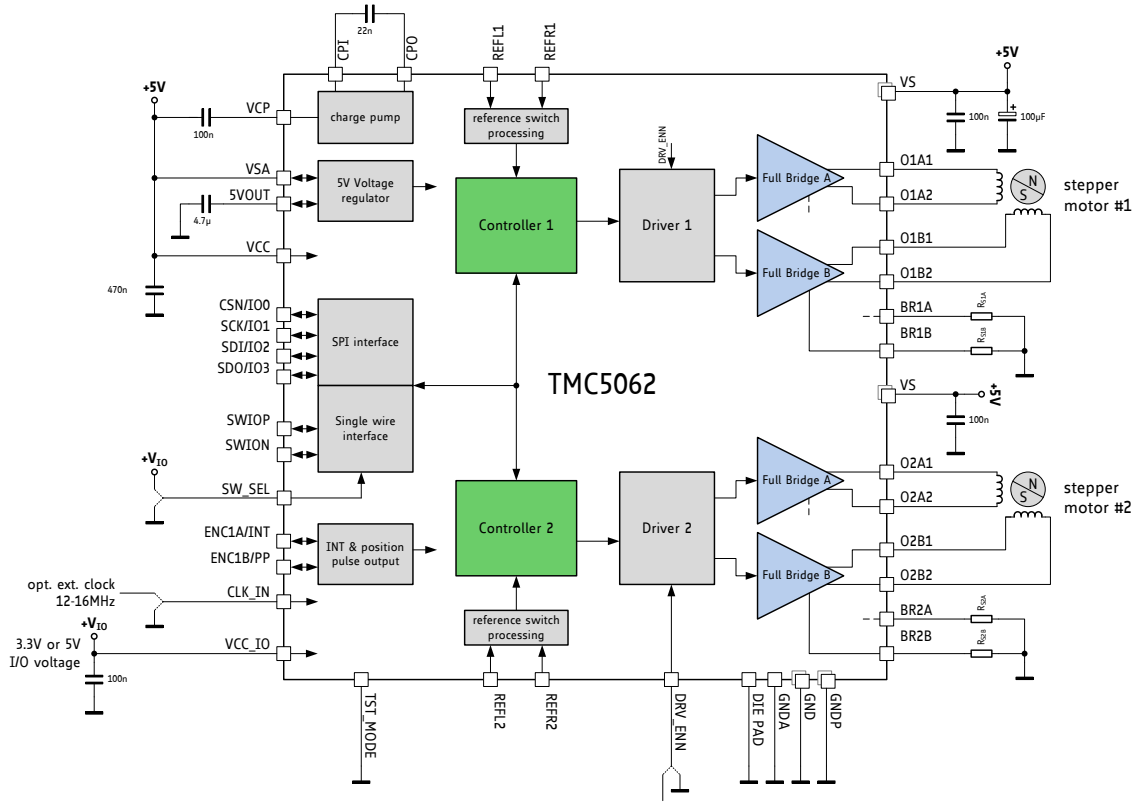


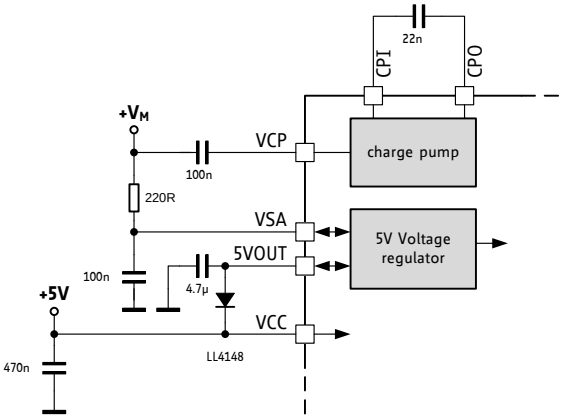
Figure 3.3 5V only operation

While the standard application circuit is limited to roughly 5.5 V lower supply voltage, a 5 V only application lets the IC run from a normal 5 V +/-5% supply. In this application, linear regulator drop must be minimized. Therefore, the major 5 V 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 External VCC Supply

Supplying VCC from an external supply is advised, when cooling of the chip is critical, e.g. at high environment temperatures in combination with high supply voltages (20 V), as the linear regulator is a major source of on-chip power dissipation. It must be made sure that the external VCC supply comes up before or synchronously with the 5VOUT supply, because otherwise the power-up reset event may be missed by the TMC5062. A diode from 5VOUT to VCC ensures this, in case the external voltage regulator is not a low drop type linear regulator. In order to prevent overload of the internal 5V regulator when using this diode, an additional series resistor has been added to VSA.

An alternative for reduced power dissipation is using a lower supply voltage for VSA, e.g. 6V to 12V. If power dissipation is critical, but no external supply is available, the clock frequency can be reduced as a first step by supplying external 12 MHz clock.



The diode is mandatory to satisfy power-up conditions!

Figure 3.4 Using an external 5V supply to reduce linear regulator power dissipation

3.3.1 Internal Regulator Bridged

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

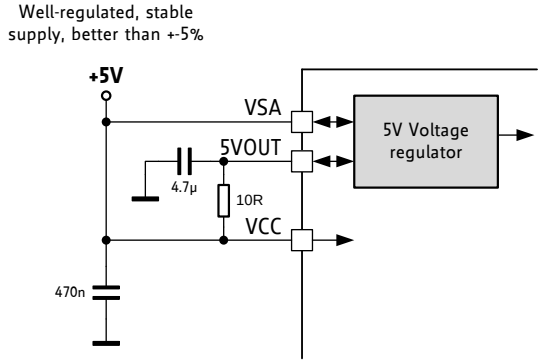


Figure 3.5 Using an external 5V supply to bypass internal regulator

3.4 Optimizing Analog Precision

The 5VOUT pin is used as an analog reference for operation of the TMC5062. Performance will degrade when there is voltage ripple on this pin. Most of the high frequency ripple in a TMC5062 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 1 MHz 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 2.2 Ω to 3.3 Ω. The voltage drop on this resistor will be roughly 100 mV (I_{VCC} * R).

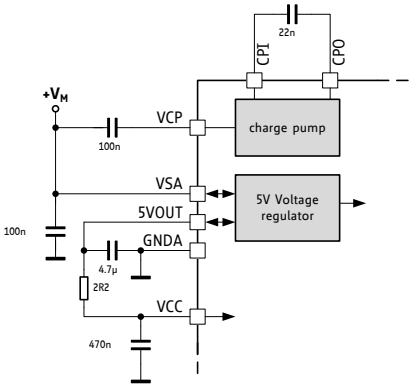


Figure 3.6 Adding an RC-Filter on VCC for reduced ripple

3.5 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.

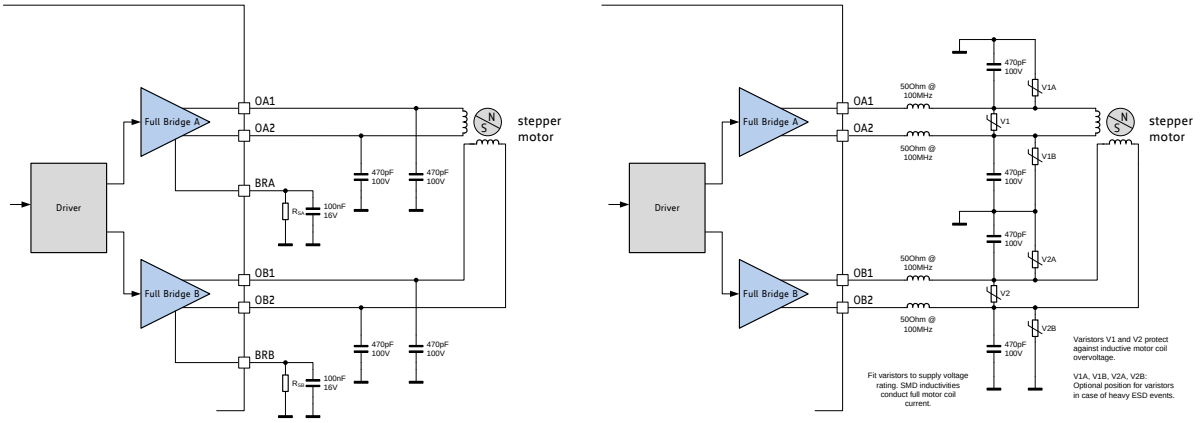


Figure 3.7 Simple ESD enhancement and more elaborate motor output protection

4 SPI Interface

4.1 SPI Datagram Structure

The TMC5062 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 TMC5062 must be handled in a way, that it stays active (low) for the complete duration of the datagram transmission.

Each datagram sent to the TMC5062 is composed of an address byte followed by four data bytes. This allows direct 32 bit data word communication with the register set of the TMC5062. 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.

TMC5062 SPI DATAGRAM STRUCTURE

MSB (transmitted first)	40 bit	LSB (transmitted last)
39 …		… 0
→ 8 bit address **← 8 bit SPI status**	**← → 32 bit data**	
39 … 32	31 … 0	
→ to TMC5062: RW + 7 bit address ← from TMC5062: 8 bit SPI status	8 bit data	8 bit data	8 bit data	8 bit data
39 / 38 … 32	31 … 24	23 … 16	15 … 8	7 … 0

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

4.1.1 Selection of Write / Read (WRITE_notREAD)

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

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

Whenever data is read from or written to the TMC5062, 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 (*VMAX*), the address byte has to be set to 0x80 + 0x27 = 0xA7. For read access, the data bit might have any value (-). So, one can set them to 0.

action	data sent to TMC5062	data received from TMC5062
read *XACTUAL*	→ 0x2100000000	← 0xSS & unused data
read *XACTUAL*	→ 0x2100000000	← 0xSS & XACTUAL
write *VMAX*:= 0x00ABCDEF	→ 0xA700ABCDEF	← 0xSS & XACTUAL
write *VMAX*:= 0x00123456	→ 0xA700123456	← 0xSS00ABCDEF

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

4.1.2 SPI Status Bits Transferred with Each Datagram Read Back

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

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

4.1.3 Data Alignment

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

4.2 SPI Signals

The SPI bus on the TMC5062 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 TMC5062.

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.

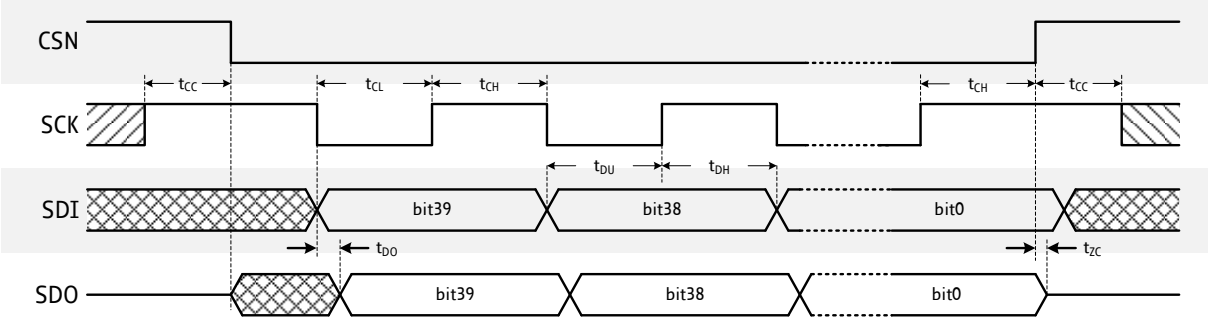


Figure 4.1 SPI timing

Hint
Usually this SPI timing is referred to as SPI MODE 3 (CPOL=1 and CPHA=1).

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 TMC5062 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 makes this interface easy and flexible to use.

5.1 Datagram Structure

5.1.1 Write Access

<table>
<tr><th colspan="17">TMC5062 UART WRITE ACCESS DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="17">each byte is LSB…MSB, highest byte transmitted first</td></tr>
<tr><td colspan="9">0 …</td><td colspan="8">… 55</td></tr>
<tr><td colspan="8">synchronization</td><td colspan="3">RW + 7 bit register address</td><td colspan="3">32 bit data</td><td colspan="3">CRC</td></tr>
<tr><td colspan="8">0…7</td><td colspan="3">8…15</td><td colspan="3">16…47</td><td colspan="3">48…55</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="2">register address</td><td>1</td><td colspan="3">data bytes 3, 2, 1, 0 (high byte to low byte)</td><td colspan="3">crc</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>…</td><td>15</td><td>16</td><td>…</td><td>47</td><td>48</td><td>…</td><td>55</td></tr>
</table>

A sync nibble precedes each transmission to and from the TMC5062 and is embedded into the first transmitted byte. The second nibble is all zero. 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 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

TMC5062 UART READ ACCESS REQUEST DATAGRAM STRUCTURE																								
each byte is LSB...MSB, highest byte transmitted first																								
synchronization								**RW + 7 bit register address**								**CRC**							
0...7								8...15								16...23							
1	0	1	0	0	0	0	0	register address							0	crc							
0	1	2	3	4	5	6	7	8	⋮						15	16	⋮						23

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 desired register for the read access. The TMC5062 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 TMC5062 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.

TMC5062 UART READ ACCESS REPLY DATAGRAM STRUCTURE																								
each byte is LSB...MSB, highest byte transmitted first																								
0 ...																								... 55
synchronization								**R + 7 bit register address**								**32 bit read data**							**CRC**
0...7								8...15								16...47							48...55
1	0	1	0	1	1	1	1	register							0	data bytes 3, 2, 1, 0 (high byte to low byte)							crc
0	1	2	3	4	5	6	7	8	⋮						15	16	⋮					47	48	⋮	55

The read response is sent to the master. The transmitter becomes switched inactive four bit times after the last bit is sent.

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 register addressing byte. The synchronization byte is assumed to always be correct. The TMC5062 responds only to correctly transmitted datagrams. 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 TMC5062-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 TMC5062 has two signals:

TMC5062 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 UART mode the slave checks the serial wire SWIOP and SWION for correctly received datagrams 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.

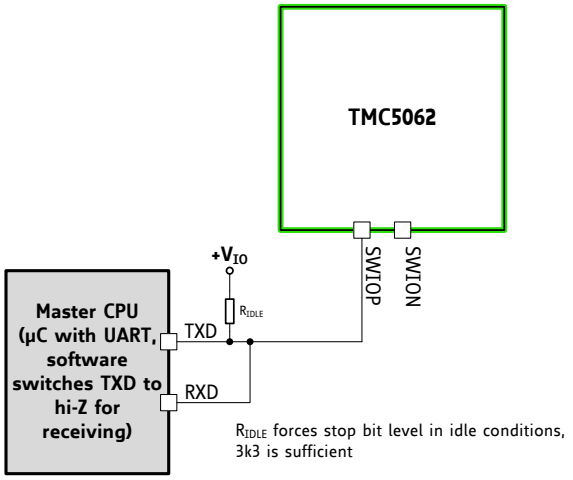


Figure 5.1 Connecting to a master via single wire UART interface

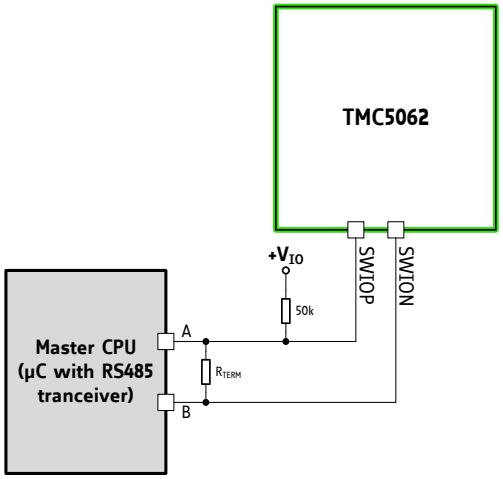


Figure 5.2 Connecting to a master via differential UART interface

6 Register Mapping

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

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

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

NOTATION OF R/W FIELD	
R	Read only
W	Write only
R/W	Read- and writable register
R+C	Clear upon read

OVERVIEW REGISTER MAPPING

REGISTER	**DESCRIPTION**
General Configuration Registers	These registers contain - global configuration - global status flags - slave address configuration - and I/O configuration
Ramp Generator Motion Control Register Set	This register set offers registers for - choosing a ramp mode - choosing velocities - homing - acceleration and deceleration - target positioning
Ramp Generator Driver Feature Control Register Set	This register set offers registers for - driver current control - setting thresholds for coolStep operation - setting thresholds for different chopper modes - setting thresholds for dcStep operation - 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 - 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…0X1F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
RW	0x00	11	*GCONF*	**Bit**	***GCONF* – Global configuration flags**
				0..2	Reserved, set to 0
				3	*poscmp_enable* 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	*enc1_refsel* 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	*enc2_enable* 0: Right reference switches are available. 1: Encoder 2 A and B signals are mapped to REFR1 and REFR2 inputs.
				6	*enc2_refsel* 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	*test_mode* 0: Normal operation 1: Enable analog test output on pin REFR2 *TEST_SEL* selects the function of REFR2: 0…4: T120, DAC1, VDDH1, DAC2, VDDH2 *Attention: Not for user, set to 0 for normal operation!*
				8	*shaft1* 1: Inverse motor 1 direction
				9	*shaft2* 1: Inverse motor 2 direction
				10	*lock_gconf* 1: GCONF is locked against further write access.
R+C	0x01	4	*GSTAT*	**Bit**	***GSTAT* – Global status flags**
				0	*reset* 1: Indicates that the IC has been reset since the last read access to *GSTAT.*
				1	*drv_err1* 1: Indicates, that driver 1 has been shut down due to overtemperature or short circuit detection since the last read access. Read *DRV_STATUS1* for details. The flag can only be reset when all error conditions are cleared.
				2	*drv_err2* *1:* Indicates, that driver 2 has been shut down due to overtemperature or short circuit detection since the last read access. Read *DRV_STATUS2* for details. The flag can only be reset when all error conditions are cleared.
				3	*uv_cp* 1: Indicates an undervoltage on the charge pump. The driver is disabled in this case.
R	0x02	8	*IFCNT*	Interface transmission counter. This register becomes	

GENERAL CONFIGURATION REGISTERS (0X00…0X1F)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	
				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	4 + 4	*SLAVECONF*	**Bit**	***SLAVECONF***
				3..0	*TEST_SEL*: selects the function of REFR2 in test mode: 0…4: T120, DAC1, VDDH1, DAC2, VDDH2 *Attention: Not for user, set to 0 for normal operation!*
				7..4	*SENDDELAY*: 0, 1: 8 bit times 2, 3: 3*8 bit times 4, 5: 5*8 bit times 6, 7: 7*8 bit times 8, 9: 9*8 bit times 10, 11: 11*8 bit times 12, 13: 13*8 bit times 14, 15: 15*8 bit times
R	0x04	8 + 8	*INPUT*	**Bit**	***INPUT***
					Reads the state of all input pins available plus the state of IO pins set to output.
				0	*io0_in*: IO0 polarity
				1	*io1_in*: IO1 polarity
				2	*io2_in*: IO2 polarity
				3	*io3_in*: IO3 polarity
				4	*iop_in:* IOP pin polarity (always input in SPI mode)
				5	*ion_in*: ION pin polarity (always input in SPI mode)
				6	Reserved, ignore this bit
				7	DRV_ENN
				31..24	*VERSION*: 0x01=first version of the IC Identical numbers mean full digital compatibility.
W		4 + 4	*OUTPUT*	**Bit**	***OUTPUT***
					Sets the IO output pin polarity and data direction.
				0	*io0_out*: IO0 output polarity
				1	*io1_out*: IO1 output polarity
				2	*io2_out*: IO2 output polarity
				3	*Io3_out*: IO3 output polarity
				8	*ioddr0*: IO0 data direction: 0=input, 1=output
				9	*ioddr1*: IO1 data direction: 0=input, 1=output
				10	*ioddr2*: IO2 data direction: 0=input, 1=output
				11	*ioddr3*: IO3 data direction: 0=input, 1=output
W	0x05	32	*X_COMPARE*	Position comparison register for motor 1 position strobe. Activate *poscmp_enable* to get position pulse on output PP. *XACTUAL = X_COMPARE*: - 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*	*RAMPMODE:* 0: Positioning mode (using all A, D and V parameters) 1: Velocity mode to positive *VMAX* (using *AMAX* acceleration) 2: Velocity mode to negative *VMAX* (using *AMAX* acceleration) 3: Hold mode (velocity remains unchanged, unless stop event occurs)	0…3
RW	0x21 0x41	32	*XACTUAL*	Actual motor position (signed) *Hint:* This value normally should only be modified, when homing the drive. In positioning mode, modifying the register content will start a motion.	-2^31… +(2^31)-1
R	0x22 0x42	24	*VACTUAL*	Actual motor velocity from ramp generator (signed) The sign matches the motion direction. A negative sign means motion to lower XACTUAL.	+-(2^23)-1 [µsteps / t]
W	0x23 0x43	18	*VSTART*	Motor start velocity (unsigned) Set *VSTOP ≥ VSTART!*	0…(2^18)-1 [µsteps / t]
W	0x24 0x44	16	*A1*	First acceleration between *VSTART* and *V1* (unsigned)	0…(2^16)-1 [µsteps / ta²]
W	0x25 0x45	20	*V1*	First acceleration / deceleration phase target velocity (unsigned) 0: Disables *A1* and *D1* phase, use *AMAX*, *DMAX* only	0…(2^20)-1 [µsteps / t]
W	0x26 0x46	16	*AMAX*	Second acceleration between *V1* and *VMAX* (unsigned) This is the acceleration and deceleration value for velocity mode.	0…(2^16)-1 [µsteps / ta²]
W	0x27 0x47	23	*VMAX*	Motion ramp target velocity (for positioning ensure *VMAX ≥ VSTART*) (unsigned) This is the target velocity in velocity mode. It can be changed any time during a motion.	0…(2^23)-512 [µsteps / t]
W	0x28 0x48	16	*DMAX*	Deceleration between *VMAX* and *V1* (unsigned)	0…(2^16)-1 [µsteps / ta²]
W	0x2A 0x4A	16	*D1*	Deceleration between *V1* and *VSTOP* (unsigned) *Attention: Do not set 0 in positioning mode, even if V1=0!*	1…(2^16)-1 [µsteps / ta²]

RAMP GENERATOR MOTION CONTROL REGISTER SET (MOTOR 1: 0X20…0X2D, MOTOR 2: 0X40…0X4D)					
R/W	**Addr**	**n**	***Register***	**Description */ bit names***	**Range [Unit]**
W	0x2B 0x4B	18	*VSTOP*	Motor stop velocity (unsigned) *Attention: Set VSTOP ≥ VSTART!* *Attention: Do not set 0 in positioning mode.*	1…(2^18)-1 [µsteps / t]
W	0x2C 0x4C	16	*TZEROWAIT*	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. This setting avoids excess acceleration e.g. from *VSTOP* to *-VSTART*.	0…(2^16)-1 * 512 t_{CLK}
RW	0x2D 0x4D	32	*XTARGET*	Target position for ramp mode (signed). Write a new target position to this register in order to activate the ramp generator positioning in *RAMPMODE*=0. Initialize all velocity, acceleration and deceleration parameters before. *Hint:* The position is allowed to wrap around, thus, *XTARGET* value optionally can be treated as an unsigned number. *Hint:* The maximum possible displacement is *+/-*((2^31)-1). *Hint:* When increasing *V1, D1 or DMAX during* a motion, rewrite *XTARGET* afterwards in order to trigger a second acceleration phase, if desired.	-2^31… +(2^31)-1

6.2.2 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)
				12..8	*IRUN* Motor run current (0=1/32…31=32/32) *Hint:* Choose sense resistors in a way, that normal *IRUN* is 16 to 31 for best microstep performance.
				19..16	*IHOLDDELAY* Controls the number of clock cycles for motor power down after a motion as soon as *T_ZEROWAIT* 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*	This is the lower threshold velocity for switching on smart energy coolStep. (unsigned) Set this parameter to disable coolStep at low speeds, where it cannot work reliably. *VHIGH* ≥ |*VACT*| ≥ VCOOLTHRS: - coolStep is enabled, if configured (Only bits 22..8 are used for value and for comparison)	
W	0x32 0x52	23	*VHIGH*	This velocity setting allows velocity dependent switching into a different chopper mode and fullstepping to maximize torque. (unsigned) |*VACT*| ≥ *VHIGH*: - coolStep is disabled (motor runs with normal current scale) - If *vhighchm* is set, the chopper switches to *chm*=1 with *TFD*=0 (constant off time with slow decay, only). - chopSync2 is switched off (*SYNC*=0) - If *vhighfs* is set, the motor operates in fullstep mode. (Only bits 22..8 are used for value and for comparison)	

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*	Automatic commutation dcStep becomes enabled above velocity V*DCMIN* (unsigned) In this mode, the actual position is determined by the sensor-less motor commutation and becomes fed back to *XACTUAL*. In case the motor becomes heavily loaded, *VDCMIN* also is used as the minimum step velocity. 0: Disable, dcStep off *|VACT|* ≥ *VDCMIN* ≥ 256: - Triggers the same actions as exceeding *VHIGH*. - Switches on automatic commutation dcStep *Hint:* Also set bits *vhighfs* and *vhighchm* and set *DCCTRL* parameters in order to operate dcStep. (Only bits 22… 8 are used for value and for comparison)
RW	0x34 0x54	11	***SW_MODE***	Switch mode configuration *See separate table!*
R+C	0x35 0x55	14	***RAMP_STAT***	Ramp status and switch event status *See separate table!*
R	0x36 0x56	32	*XLATCH*	Ramp generator latch position, latches *XACTUAL* upon a programmable switch event (see *SW_MODE*). *Hint:* The encoder position can be latched to *ENC_LATCH* together with *XLATCH* to allow consistency checks.

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

6.2.2.1 *SW_MODE* – Reference Switch and stallGuard2 Event Configuration Register

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

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

6.3 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	11	***ENCMODE***	Encoder configuration and use of N channel *See separate table!*	
RW	0x39 0x59	32	*X_ENC*	Actual encoder position (signed)	-2^31... +(2^31)-1
W	0x3A 0x5A	32	*ENC_CONST*	Accumulation constant (signed) 16 bit integer part, 16 bit fractional part *X_ENC* accumulates *+/- ENC_CONST /* (2^16**X_ENC*) (binary) or *+/-ENC_CONST /* (10^4**X_ENC*) (decimal) *ENCMODE* bit *enc_sel_decimal* switches between decimal and binary setting. Use the sign, to match rotation direction!	binary: ± [µsteps/2^16] ±(0 ... 32767.9999847) decimal: ±(0 ... 32767.9999) *reset default = 1.0 (=65536)*
R+C	0x3B 0x5B	1	*ENC_STATUS*	bit 0: *n_event* 1: Encoder N event detected. Status bit is cleared on read: Read (R) + clear (C) This bit is ORed to the *interrupt output* signal.	
R	0x3C 0x5C	32	*ENC_LATCH*	Encoder position *X_ENC* latched on N event	

6.2.2.3 *ENCMODE* – Encoder Register

0x38, 0x58: *ENCMODE* – ENCODER REGISTER			
Bit	**Name**	**Comment**	
10	*enc_sel_decimal*	0	Encoder prescaler divisor binary mode: Counts *ENC_CONST(fractional part)* /65536
		1	Encoder prescaler divisor decimal mode: Counts in *ENC_CONST(fractional part)* /10000
9	*latch_x_act*	1: Also latch *XACTUAL* position together with *X_ENC*. Allows latching the ramp generator position upon an N channel event as selected by *pos_edge* and *neg_edge*.	
8	*clr_enc_x*	0	Upon N event, *X_ENC* becomes latched to *ENC_LATCH* only
		1	Latch and additionally clear encoder counter *X_ENC* at N-event
7	*neg_edge*	***n p***	**N channel event sensitivity**
6	*pos_edge*	0 0	N channel event is active during an active N event level
		0 1	N channel is valid upon active going N event
		1 0	N channel is valid upon inactive going N event
		1 1	N channel is valid upon active going and inactive going N event
5	*clr_once*	1: Latch or latch and clear *X_ENC* on the next N event following the write access	
4	*clr_cont*	1: Always latch or latch and clear *X_ENC* upon an N event (once per revolution, it is recommended to combine this setting with edge sensitive N event)	
3	*ignore_AB*	0	An N event occurs only when polarities given by *pol_N*, *pol_A* and *pol_B* match.
		1	Ignore A and B polarity for N channel event
2	*pol_N*	Defines active polarity of N (0=neg., 1=pos.)	
1	*pol_B*	Required B polarity for an N channel event (0=neg., 1=pos.)	
0	*pol_A*	Required A polarity for an N channel event (0=neg., 1=pos.)	

6.4 Motor Driver Registers

MOTOR DRIVER REGISTER SET (MOTOR 1: 0X60...0X6F, MOTOR 2: 0X70...0X7F)					
R/W	**Addr**	**n**	***Register***	**Description *I bit names***	**Range [Unit]**
W	0x60 0x70	32	*MSLUT1[0]* *MSLUT2[0]* microstep table entries 0...31	Each bit gives the difference between microstep x and x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90_120*. *ofs31*, *ofs30*, ..., *ofs01*, *ofs00* ... *ofs255*, *ofs254*, ..., *ofs225*, *ofs224*	32x 0 or 1 *reset default= sine wave table*
W	0x61 ... 0x67 0x71 ... 0x77	7 x 32	*MSLUT1[1...7]* *MSLUT2[1...7]* microstep table entries 32...255		7x 32x 0 or 1 *reset default= sine wave table*
W	0x68 0x78	32	***MSLUTSEL1*** ***MSLUTSEL2***	This register defines four segments within each quarter *MSLUT* wave. Four 2 bit entries determine the meaning of a 0 and a 1 bit in the corresponding segment of *MSLUT*. *See separate table!*	*0<X1<X2<X3* *reset default= sine wave table*
W	0x69 0x79	8 + 8	*MSLUTSTART*	bit 7... 0: *START_SIN* bit 23... 16: *START_SIN90_120* *START_SIN* gives the absolute current at microstep table entry 0. *START_SIN90_120* gives the absolute current for microstep table entry at positions 256. Start values are transferred to the microstep registers *CUR_A* and *CUR_B*, whenever the reference position *MSCNT*=0 is passed.	*START_SIN* *reset default =0* *START_SIN90_120* *reset default =247*
R	0x6A 0x7A	10	*MSCNT*	Microstep counter. Indicates actual position in the microstep table for *CUR_A*. *CUR_B* uses an offset of 256. *Hint:* Move to a position where *MSCNT* is zero before re-initializing *MSLUTSTART* or *MSLUT* and *MSLUTSEL*.	
R	0x6B 0x7B	9 + 9	*MSCURACT*	bit 8... 0: *CUR_A* (signed): Actual microstep current for motor phase A as read from MSLUT (not scaled by current) bit 24... 16: *CUR_B* (signed): Actual microstep current for motor phase B as read from *MSLUT* (not scaled by current)	
RW	0x6C 0x7C	32	***CHOPCONF***	chopper and driver configuration *See separate table!*	
W	0x6D 0x7D	25	***COOLCONF***	coolStep smart current control register and stallGuard2 configuration *See separate table!*	

W	0x6E 0x7E	8 + 8	*DCCTRL*	dcStep (*DC*) automatic commutation configuration register: bit 7... 0: *DC_TIME*: Upper PWM on time limit for commutation (*DC_TIME* * $1/f_{CLK}$). Set slightly above effective blank time *TBL*. bit 15... 8: *DC_SG*: Max. PWM on time for step loss detection using dcStep stallGuard2 in dcStep mode. (*DC_SG* * $16/f_{CLK}$) Set slightly higher than *DC_MAXI*/16 0=disable	
R	0x6F 0x7F	32	***DRV_ STATUS***	stallGuard2 value and driver error flags *See separate table!*	

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

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

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

6.4.1 *MSLUTSEL* – Look up Table Segmentation Definition

0x68, 0x78: *MSLUTSEL* – LOOK UP TABLE SEGMENTATION DEFINITION			
Bit	**Name**	**Function**	**Comment**
31	*X3*	LUT segment 3 start	The sine wave look up table can be divided into up to four segments using an individual step width control entry *Wx*. The segment borders are selected by *X1*, *X2* and *X3*. Segment 0 goes from 0 to *X1*-1. Segment 1 goes from *X1* to *X2*-1. Segment 2 goes from *X2* to *X3*-1. Segment 3 goes from *X3* to 255. For defined response the values shall satisfy: 0<*X1*<*X2*<*X3*
30			
29			
28			
27			
26			
25			
24			
23	*X2*	LUT segment 2 start	
22			
21			
20			
19			
18			
17			
16			
15	*X1*	LUT segment 1 start	
14			
13			
12			
11			
10			
9			
8			
7	*W3*	LUT width select from *ofs(X3)* to *ofs255*	Width control bit coding *W0*…*W3*: %00: MSLUT entry 0, 1 select: -1, +0 %01: MSLUT entry 0, 1 select: +0, +1 %10: MSLUT entry 0, 1 select: +1, +2 %11: MSLUT entry 0, 1 select: +2, +3
6			
5	*W2*	LUT width select from *ofs(X2)* to *ofs(X3-1)*	
4			
3	*W1*	LUT width select from *ofs(X1)* to *ofs(X2-1)*	
2			
1	*W0*	LUT width select from *ofs00* to *ofs(X1-1)*	
0			

6.4.2 *CHOPCONF* – Chopper Configuration

0x6C, 0x7C: CHOPCONF – CHOPPER CONFIGURATION			
Bit	**Name**	**Function**	**Comment**
31	-	reserved	set to 0
30	*diss2g*	short to GND protection disable	0: Short to GND protection is on 1: Short to GND protection is disabled
29	-	reserved	set to 0
28	-	reserved	set to 0
27	-	reserved	set to 0
26	-	reserved	set to 0
25	-	reserved	set to 0
24	-	reserved	set to 0
23	*sync3*	SYNC PWM synchronization clock	This register allows synchronization of the chopper for both phases of a two phase motor in order to avoid the occurrence of a beat, especially at low motor velocities. It is automatically switched off above VHIGH. %0000: Chopper sync function chopSync off %0001 … %1111: Synchronization with $f_{SYNC} = f_{CLK}/(sync*64)$ Hint: Set TOFF to a low value, so that the chopper cycle is ended, before the next sync clock pulse occurs. Set for the double desired chopper frequency for chm=0, for the desired base chopper frequency for chm=1.
22	*sync2*		
21	*sync1*		
20	*sync0*		
19	*vhighchm*	high velocity chopper mode	This bit enables switching to *chm*=1 and *fd*=0, when *VHIGH* is exceeded. This way, a higher velocity can be achieved. Can be combined with *vhighfs*=1. If set, the *TOFF* setting automatically becomes doubled during high velocity operation in order to avoid doubling of the chopper frequency.
18	*vhighfs*	high velocity fullstep selection	This bit enables switching to fullstep, when *VHIGH* is exceeded. Switching takes place only at 45° position. The fullstep target current uses the current value from the microstep table at the 45° position.
17	*vsense*	sense resistor voltage based current scaling	0: Low sensitivity, high sense resistor voltage 1: High sensitivity, low sense resistor voltage
16	*tbl1*	TBL blank time select	%00 … %11: Set comparator blank time to 16, 24, 36 or 54 clocks *Hint:* %01 or %10 recommended for most applications
15	*tbl0*		
14	*chm*	chopper mode	0: Standard mode (spreadCycle) 1: Constant off time with fast decay time. Fast decay time is also terminated when the negative nominal current is reached. Fast decay is after on time.
13	*rndtf*	random *TOFF* time	0: Chopper off time is fixed as set by *TOFF* 1: Random mode, *TOFF* is random modulated by d_{NCLK}= -12 … +3 clocks.
12	*disfdcc*	fast decay mode	chm=1: disfdcc=1 disables current comparator usage for termination of the fast decay cycle

11	*fd3*	*TFD* [3]	chm=1: MSB of fast decay time setting TFD	
10 9 8 7	*hend3* *hend2* *hend1* *hend0*	*HEND* hysteresis low value *OFFSET* sine wave offset	*chm*=0	%0000 … %1111: Hysteresis is -3, -2, -1, 0, 1, …, 12 (1/512 of this setting adds to current setting) This is the hysteresis value which becomes used for the hysteresis chopper.
			chm=1	%0000 … %1111: Offset is -3, -2, -1, 0, 1, …, 12 This is the sine wave offset and 1/512 of the value becomes added to the absolute value of each sine wave entry.
6 5 4	*hstrt2* *hstrt1* *hstrt0*	*HSTRT* hysteresis start value added to *HEND*	*chm*=0	%000 … %111: Add 1, 2, …, 8 to hysteresis low value *HEND* (1/512 of this setting adds to current setting) *Attention: Effective HEND+HSTRT ≤ 16.* *Hint*: Hysteresis decrement is done each 16 clocks
		TFD [2..0] fast decay time setting	*chm*=1	Fast decay time setting (MSB: *fd3*): %0000 … %1111: Fast decay time setting *TFD* with *NCLK*= 32**HSTRT* (%0000: slow decay only)
3 2 1 0	*toff3* *toff2* *toff1* *toff0*	*TOFF* off time and driver enable	Off time setting controls duration of slow decay phase *NCLK*= 12 + 32**TOFF* %0000: Driver disable, all bridges off %0001: 1 – use only with *TBL* ≥ 36 clocks %0010 … %1111: 2 … 15	

6.4.3 *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	*sfilt*	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	*sgt6*	stallGuard2 threshold value	This signed value controls stallGuard2 level for stall output and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value working with most motors. -64 to +63: A higher value makes stallGuard2 less sensitive and requires more torque to indicate a stall.
21	*sgt5*		
20	*sgt4*		
19	*sgt3*		
18	*sgt2*		
17	*sgt1*		
16	*sgt0*		
15	*seimin*	minimum current for smart current control	0: 1/2 of current setting (*IRUN*) 1: 1/4 of current setting (*IRUN*)
14	*sedn1*	current down step speed	%00: For each 32 stallGuard2 values decrease by one %01: For each 8 stallGuard2 values decrease by one %10: For each 2 stallGuard2 values decrease by one %11: For each stallGuard2 value decrease by one
13	*sedn0*		
12	-	reserved	set to 0
11	*semax3*	stallGuard2 hysteresis value for smart current control	If the stallGuard2 result is equal to or above (*SEMIN*+*SEMAX*+1)*32, the motor current becomes decreased to save energy. %0000 … %1111: 0 … 15
10	*semax2*		
9	*semax1*		
8	*semax0*		
7	-	reserved	set to 0
6	*seup1*	current up step width	Current increment steps per measured stallGuard2 value %00 … %11: 1, 2, 4, 8
5	*seup0*		
4	-	reserved	set to 0
3	*semin3*	minimum stallGuard2 value for smart current control and smart current enable	If the stallGuard2 result falls below *SEMIN**32, the motor current becomes increased to reduce motor load angle. %0000: smart current control coolStep off %0001 … %1111: 1 … 15
2	*semin2*		
1	*semin1*		
0	*semin0*		

6.4.4 DRV_STATUS – stallGuard2 Value and Driver Error Flags

0x6F, 0x7F: DRV_STATUS – STALLGUARD2 VALUE AND DRIVER ERROR FLAGS			
Bit	**Name**	**Function**	**Comment**
31	*stst*	standstill indicator	This flag indicates motor stand still in each operation mode.
30	*olb*	open load indicator phase B	1: Open load detected on phase A or B *Hint:* This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion or after a motion, only.
29	*ola*	open load indicator phase A	
28	*s2gb*	short to ground indicator phase B	1: Short to GND detected on phase A or B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (*TOFF*=0) or by the ENN input.
27	*s2ga*	short to ground indicator phase A	
26	*otpw*	overtemperature pre-warning flag	1: Overtemperature pre-warning threshold is exceeded. The overtemperature pre-warning flag is common for both drivers.
25	*ot*	overtemperature flag	1: Overtemperature limit has been reached. Drivers become disabled until *otpw* is also cleared due to cooling down of the IC. The overtemperature flag is common for both drivers.
24	*stallGuard*	stallGuard2 status	1: Motor stall detected (*SG_RESULT*=0) or dcStep stall in dcStep mode.
23	-	reserved	Ignore these bits
22			
21			
20	*CS ACTUAL*	actual motor current / smart energy current	Actual current control scaling, for monitoring smart energy current scaling controlled via settings in register *COOLCONF*, or for monitoring the function of the automatic current scaling.
19			
18			
17			
16			
15	*fsactive*	full step active indicator	1: Indicates that the driver has switched to fullstep as defined by chopper mode settings and velocity thresholds.
14	-	reserved	Ignore these bits
13			
12			
11			
10			
9	*SG_ RESULT*	stallGuard2 result respectively PWM on time for coil A in stand still for motor temperature detection	Mechanical load measurement: The stallGuard2 result gives a means to measure mechanical motor load. A higher value means lower mechanical load. A value of 0 signals highest load. With optimum *SGT* setting, this is an indicator for a motor stall. The stall detection compares *SG_RESULT* to 0 in order to detect a stall. *SG_RESULT* is used as a base for coolStep operation, by comparing it to a programmable upper and a lower limit. *SG_RESULT* is not applicable when dcStep is active. stallGuard2 works best with microstep operation. Temperature measurement: In standstill, no stallGuard2 result can be obtained. *SG_RESULT* shows the chopper on-time for motor coil A instead. If the motor is moved to a determined microstep position at a certain current setting, a comparison of the chopper on-time can help to get a rough estimation of motor temperature. As the motor heats up, its coil resistance rises and the chopper on-time increases.
8			
7			
6			
5			
4			
3			
2			
1			
0			

7 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.

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

Parameter	Description	Setting	Comment
IRUN	Current scale when motor is running. Scales coil current values as taken from the internal sine wave table. For high precision motor operation, work with a current scaling factor in the range 16 to 31, because scaling down the current values reduces the effective microstep resolution by making microsteps coarser. This setting also controls the maximum current value set by coolStep.	0 … 31	scaling factor 1/32, 2/32, … 32/32
IHOLD	Identical to *IRUN*, but for motor in stand still.		
IHOLD DELAY	Allows smooth current reduction from run current to hold current. *IHOLDDELAY* controls the number of clock cycles for motor power down after *TZEROWAIT* in increments of 2^18 clocks: 0=instant power down, 1..15: Current reduction delay per current step in multiple of 2^18 clocks. *Example:* When using *IRUN*=31 and *IHOLD*=16, 15 current steps are required for hold current reduction. A *IHOLDDELAY* setting of 4 thus results in a power down time of 4*15*2^18 clock cycles, i.e. roughly one second at 16MHz.	0	instant *IHOLD*
		1 …15	$1*2^{18}$ … $15*2^{18}$ clocks per current decrement
vsense	Allows control of the sense resistor *voltage range* for full scale current.	0	V_{FS} = 0.32 V
		1	V_{FS} = 0.18 V

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 20.3.

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 sees a bit 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 Chopper Operation

The currents through both motor coils are controlled using choppers. The choppers work independently of each other. In Figure 8.1 the different chopper phases are shown.

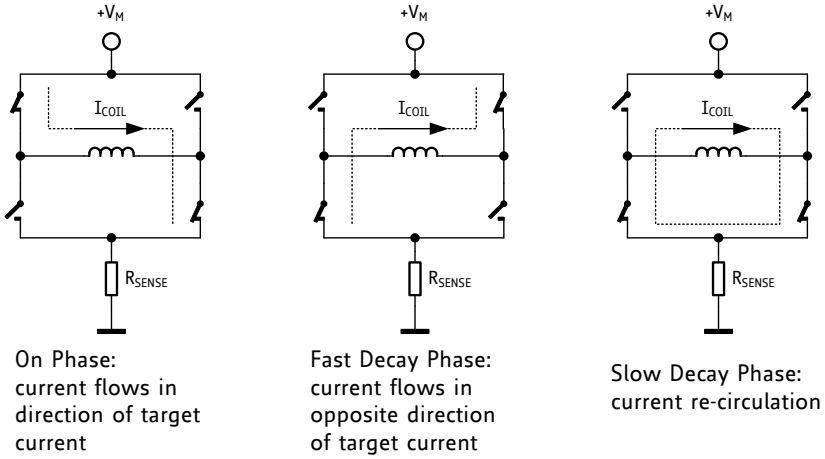


Figure 8.1 Chopper phases

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

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

There are two 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 20 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. A higher frequency leads to increased switching losses. It is advised to check the resulting frequency and to work below 50 kHz.

Three parameters are used for controlling both chopper modes:

Parameter	Description	Setting	Comment
TOFF	Sets the slow decay time (*off time*). This setting also limits the maximum chopper frequency. Setting this parameter to zero completely disables all driver transistors and the motor can free-wheel.	0	chopper off
		1...15	off time setting N_{CLK}= 12 + 32**TOFF* (1 will work with minimum blank time of 24 clocks)
TBL	Selects the comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing on the sense resistor. For most applications, a setting of 1 or 2 is good. For highly capacitive loads, e.g. when filter networks are used, a setting of 2 or 3 will be required.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	36 t_{CLK}
		3	54 t_{CLK}
chm	Selection of the *chopper mode*	0	spreadCycle
		1	classic const. off time

8.1 spreadCycle Chopper

The spreadCycle (pat. fil.) chopper algorithm is a precise and simple to use chopper mode which automatically determines the optimum length for the fast-decay phase. Several parameters are available to optimize the chopper to the application.

Each chopper cycle is comprised of an on phase, a slow decay phase, a fast decay phase and a second slow decay phase (see Figure 8.3). The two slow decay phases and the two blank times per chopper cycle put an upper limit to the chopper frequency. The slow decay phases typically make up for about 30%-70% of the chopper cycle in standstill and are important for low motor and driver power dissipation.

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

Assumptions:
Target Chopper frequency: 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 8.2). Checking the sine wave shape near zero transition will show a small ledge between both half waves in case the hysteresis setting is too small. At medium velocities (i.e.

100 to 400 fullsteps per second), a too low hysteresis setting will lead to increased humming and vibration of the motor.

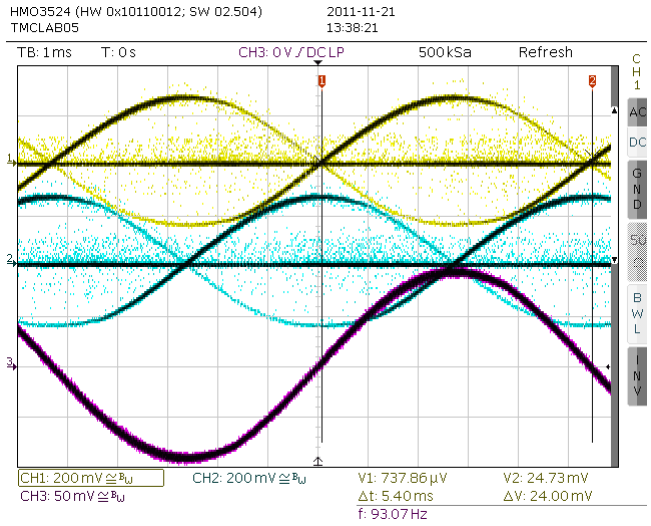


Figure 8.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B)

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

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 16.
For detail procedure see Application Note AN001 - *Parameterization of spreadCycle*

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

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

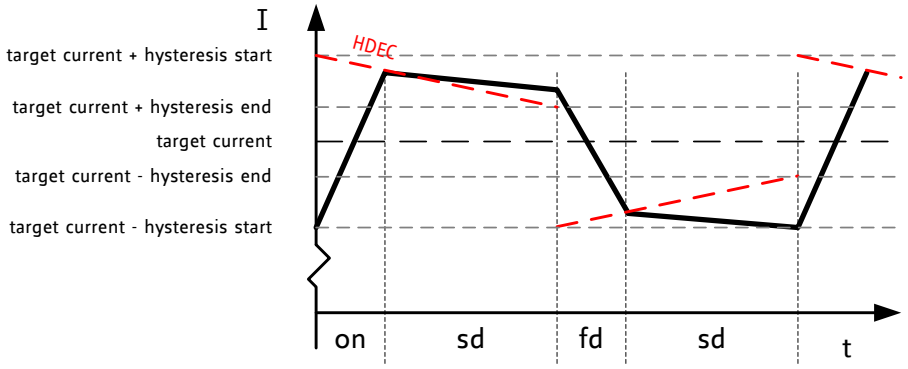


Figure 8.3 spreadCycle chopper scheme showing coil current during a chopper cycle

Two parameters control spreadCycle mode:

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

Even at HSTRT=0 and HEND=0, the TMC5062 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.

8.2 Classic 2-Phase Motor Constant Off Time Chopper

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

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

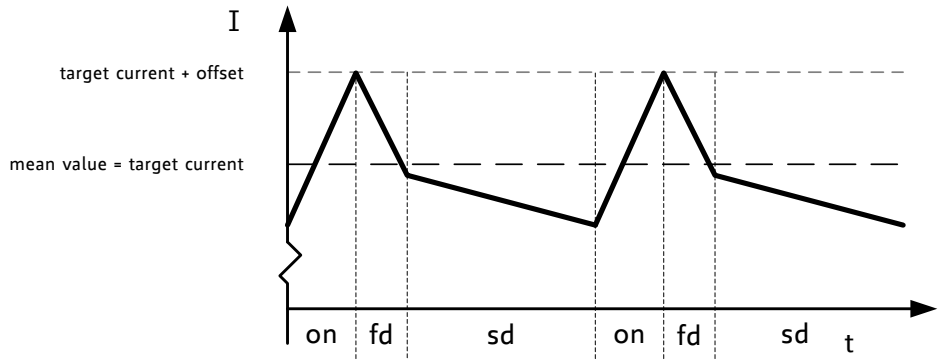


Figure 8.4 Classic const. off time chopper with offset showing coil current

After tuning the fast decay time, the offset should be tuned for a smooth zero crossing. This is necessary because the fast decay phase makes the absolute value of the motor current lower than the target current (see Figure 8.5). If the zero offset is too low, the motor stands still for a short moment during current zero crossing. If it is set too high, it makes a larger microstep. Typically, a positive offset setting is required for smoothest operation.

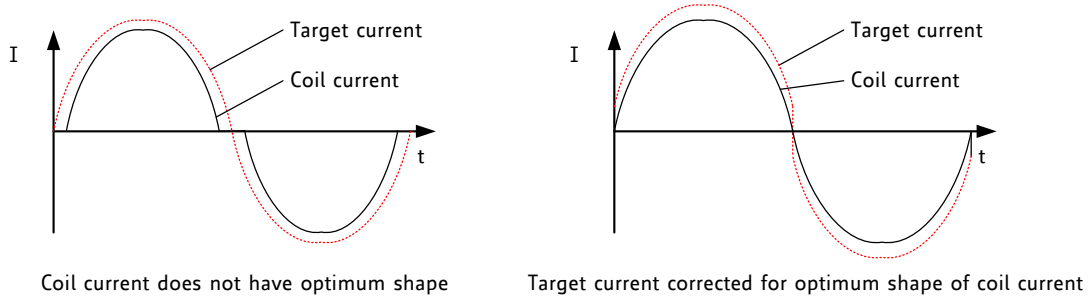


Figure 8.5 Zero crossing with classic chopper and correction using sine wave offset

Three parameters control constant off-time mode:

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

8.3 Random Off Time

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

Hint
A common factor, which can cause motor noise, is a bad PCB layout causing coupling of both sense resistor voltages (please refer layouts hint in chapter 20.3).

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

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

8.4 chopSync2 for Quiet Motors

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

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

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

Example:

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

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

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

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

9 Driver Diagnostic Flags

The TMC5062 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.

9.1 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.

9.2 Short to GND Protection

The TMC5062 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.

9.3 Open Load Diagnostics

Interrupted cables are a common cause for systems failing, e.g. when connectors are not firmly plugged. The TMC5062 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.

10 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 TMC5062 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 10.2.

10.1 Real World Unit Conversion

The TMC5062 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 TMC5062 register content are written as register[5062].

PARAMETER VS. UNITS		
Parameter / Symbol	**Unit**	**calculation / description / comment**
f_{CLK}[Hz]	[Hz]	clock frequency of the TMC5062 in [Hz]
s	[s]	second
US	µstep	
FS	fullstep	
µstep velocity v[Hz]	µsteps / s	v[Hz] = v[5062] * (f_{CLK}[Hz]/2 / 2^23)
µstep acceleration a[Hz/s]	µsteps / s^2	a[Hz/s] = a[5062] * f_{CLK}[Hz]^2 / (512*256) / 2^24
USC microstep count	counts	microstep resolution in number of microsteps (i.e. the number of microsteps between two fullsteps – normally 256)
rotations per second v[rps]	rotations / s	v[rps] = v[µsteps/s] / USC / FSC FSC: motor fullsteps per rotation, e.g. 200
rps acceleration a[rps/s^2]	rotations / s^2	a[rps/s^2] = a[µsteps/s^2] / USC / FSC
ramp steps[µsteps] = rs	µsteps	rs = (v[5062])^2 / a[5062] / 2^8 microsteps during linear acceleration ramp (assuming acceleration from 0 to v)

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 16.

10.2 Motion Profiles

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

10.2.1 Ramp Mode

The ramp generator delivers two phase acceleration and two phase deceleration ramps with additional programmable start and stop velocities (see Figure 10.1).

> *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.*

10.2.2 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.

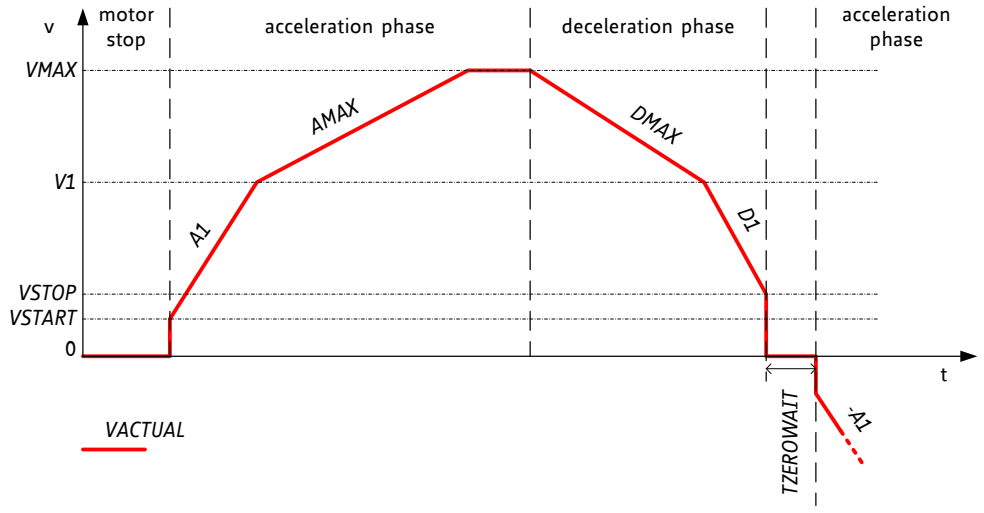


Figure 10.1 Ramp generator velocity trace showing consequent move in negative direction

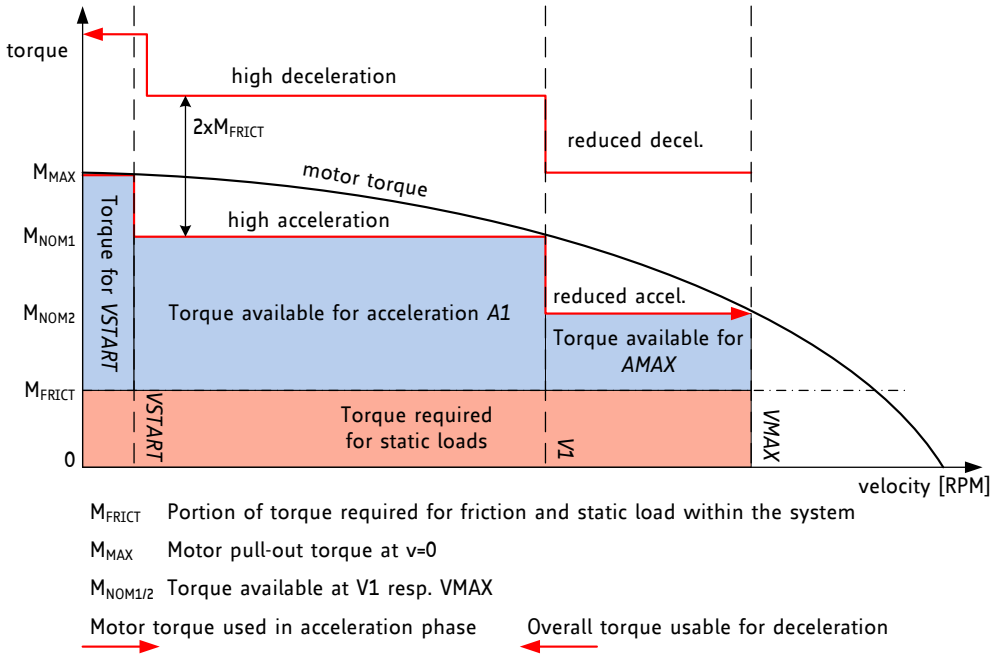


Figure 10.2 Illustration of optimized motor torque usage with TMC5062 ramp generator

10.2.3 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.

Please see chapter 10.6 for a known restriction of the velocity mode.

10.3 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 POSITION_REACHED):

1. Read *RAMP_STAT1* to clear the interrupt flags. This will turn off the interrupt source.
2. Check *XACTUAL1* 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 *XACTUAL2*.

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

10.4 Velocity Thresholds

The ramp generator provides a number of velocity thresholds coupled to the actual velocity *VACTUAL*. The different ranges allow programming the motor to the optimum step mode, coil current and acceleration settings.

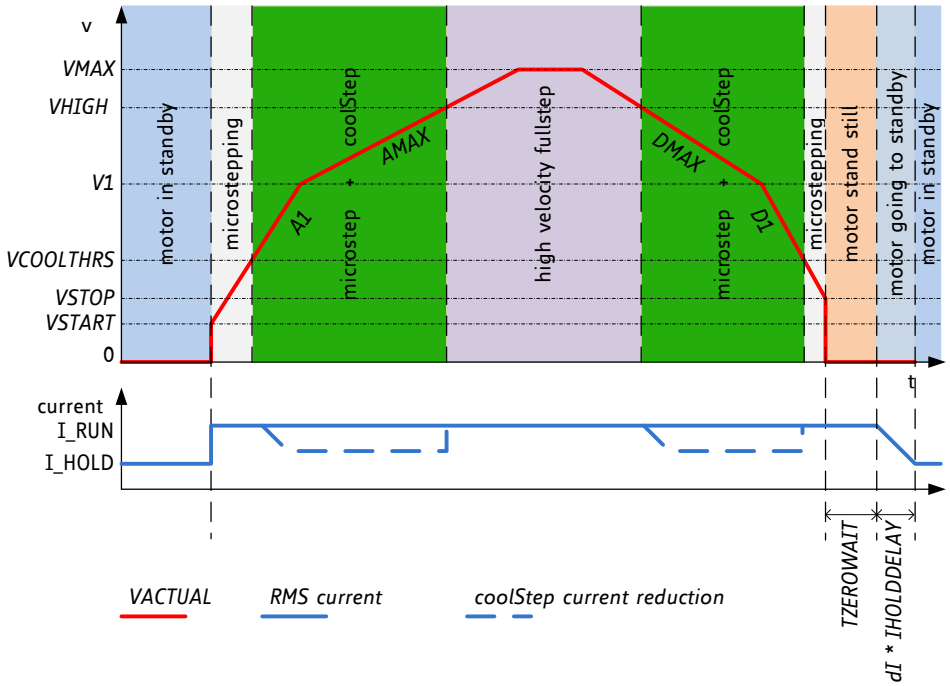


Figure 10.3 Ramp generator velocity dependent motor control

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.

10.5 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 (*DMAX*, *V1*, *D1*).

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

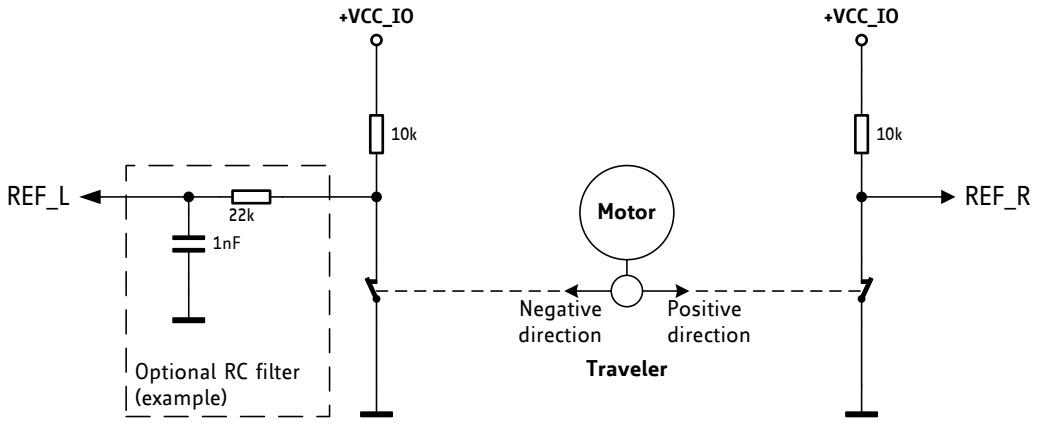


Figure 10.4 Using reference switches (example)

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 TMC5062 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.

10.6 Restrictions of Ramp Generator (Errata)

When the TMC5062 becomes stopped in velocity mode, there is an irregularity of the position counter failing and counting continuously with clock frequency until the next move is commanded.

Failure condition:

1. Motor is moving in velocity mode
2. Master sets *VMAX*=0 to stop the motion
3. Upon reaching of *VACTUAL*=0, the position counter may start counting with clock frequency (The deterministically probability for this behavior occurring is about 1/16 Million.)

In this situation the motor is correctly in standstill and also the ramp state reports the motor to be stopped. When starting the motor again, the position counter continues from the new (wrong) position. This behavior leads to a loss of the synchronization between the position counter and the motor position.

Background:
The restriction is caused by a failure state, which involves the state of the internal velocity pulse generator and the actual point of time, when the velocity becomes zero. When the velocity *VACTUAL* becomes decreased from one to zero with the 24 bit ramp generator register in a certain state, the *XACTUAL* position counter gets to a state where it counts up despite the velocity now being zero. This can occur in velocity mode only, because in this mode the internal change of the velocity register is not coupled to an advance in the actual position. The statistical probability for the occurrence of the failure is given by the combination of 2^24 (i.e. 16M) possible states of the accumulation register, with one of the states leading to a fail. If the one state of the accumulation register, which leads to an overflow of the register in case of an accumulation of the last velocity value (1) before reaching zero occurs at exactly the same moment where the velocity actually goes to zero, the *XACTUAL* counter gets caught in an endless loop.

10.6.1 Velocity Mode Workaround

There are two alternatives for a workaround. The first workaround is recommended for most applications which require the use of velocity mode. Therefore the application software must allow polling a register on a deterministic, regular time interval. The second workaround has less real time relevance, as it just requires a read-modify-write instruction to execute within limited time.

First Software Workaround for Applications Using Velocity Mode Intensively

The velocity mode can be used, but in order to stop the motor, do not directly set *VMAX*=0.

Workaround for stopping the motor:

1. Set *VMAX* to a low velocity, in the range 1 to 2000 (e.g. 100). Even if *VMAX* has been lower before, this ensures a quick termination of the stop procedure. Exit the stop procedure, in case *VMAX* already had been set to 0 before (motor is stopped).
2. Check the *velocity_reached* flag to become active. Alternatively, check if the absolute value of *VACTUAL* is at or below the value selected for step 1.
3. Poll *XACTUAL* until a new step has been executed (i.e. *XACTUAL* has changed) (with *VMAX*=100 this will need at maximum about 10ms, with *VMAX*=1000, about 1ms) (*)

4. Set *AMAX* to 65535 (0xFFFF) and set *VMAX* to zero to finally stop the motor. This will stop the motor within a few microseconds.
5. Wait until the motor is actually stopped (*vzero* flag active) before starting a new motion. Remember to set *AMAX* back to the original value before starting the next motion.

Step (3.) and (4.) are time critical:
Make sure that the delay between detection of the step execution by reading *XACTUAL* and setting *VMAX*=0 is significantly lower than the time between each two steps. No additional step shall be executed between (3.) and (4.). For example, when *XACTUAL* can be checked once each 5ms, use a step frequency of max. 100Hz (10ms) for *VMAX* in step (1.). You can test the procedure by checking that no further position change has been executed until step 5.

Do not switch between *RAMPMODE* 1 and 2 (velocity in positive direction and velocity in negative direction), without stopping the motion as described above before changing the direction.

Second Software Workaround Avoiding Velocity Mode

Operate the device in positioning mode instead of velocity mode.
Use a target position far away to simulate a velocity mode movement, e.g. *XTARGET*:=*XACTUAL*+2^30 to yield a positive motion direction, or *XTARGET*:=*XACTUAL*-2^30 for a negative direction. A smaller increment down to the span of the deceleration ramp also can be used, depending on how often the procedure is called. The target position this way can be increased in regular intervals in order to have an infinite running (even longer than the 32 bit position range).

In order to stop the motor, cease incrementing *XTARGET*. The motor will continue turning and decelerate in time to stop as commanded by the last increment.

In order to stop the motor at the next possible position:

1. Set *VMAX* to a low velocity, in the range of minimum equal to *VSTOP* or up to about 1000 (e.g. 100). Depending on the speed of execution of step 4 (mostly limited by communication between MCU and TMC), higher values can be chosen to speed up the motor stop process.
2. Check the *velocity_reached* flag to become active. In case the *position_reached* flag becomes active, exit the procedure as the motion has finished normally.
3. Read out *XACTUAL*. For a motion in positive direction, increase it by 2 (or more, e.g. 10 or 100, if desired), and write it to *XTARGET*, for a motion in negative direction, decrease it accordingly. Increase *VSTOP* to the same value which was selected for *VMAX* in step 1. This will stop the motor within two steps (or 10, or 100) of the write access to *XTARGET*. (with *VMAX*=100 this will need at maximum about 10-20ms, with *VMAX*=1000, about 1-2ms)
4. Wait until the motor is actually stopped (*vzero* flag active) before starting a new motion. Remember to set *VSTOP* and *VMAX* back to the original values before starting the next positioning move.

The read-modify-write access in step (3.) is time critical: Make sure that the delay between reading *XACTUAL* and writing to *XTARGET* and *VSTOP* is significantly lower than the time required doing the remaining 2 steps (or more, as decided for the increment in step (3.)). Otherwise the motor might reverse before stopping). With *VSTOP*=10, the remaining motion ramp will need about 200ms (*), with *VSTOP*=100 it will need about 20ms.

(*) The time delays given relate to a clock frequency of about 16MHz. At 12MHz they are 25% longer.

Optional Detection and Correction

This option risks the occurrence of the error and detects and corrects it. The irregularity of the position counter can easily be detected by reading the counter twice whenever the motor is brought to standstill (VZERO flag set). In case, two subsequent read accesses of *XACTUAL* show a different result during standstill, the position is lost. Trigger a new homing sequence.

This solution will work well for applications with a low sequence of motion tasks, which allow doing a new homing sequence. In case only one critical motion command per minute is issued, the mean time to failure and automatic correction will be > 10 years.

10.6.2 TZEROWAIT and VSTART Restriction

This restriction applies in case that positioning mode is used with alternation of target-positions on-the-fly, i.e. when a reversal of the motion direction can occur due to a change of the target position, while the motor is moving.

In this case, set *TZEROWAIT*=0. Set *VSTART* to minimum 1 (or to a higher value).

Hint: Take care, that *VSTOP* is always required to be higher than *VSTART*, i.e. *VSTOP* must be minimum 2.

10.6.3 Stop Switch Handling Restriction

In case a stop switch is used for homing in conjunction with the automatic motor stop (*stop_l_enable*=1 or *stop_r_enable*=1), a soft stop shall be used (set *en_softstop*=1). Set the deceleration parameters to the desired value.

Hint: In any case, a homing requires use of the soft stop, as a hard stop might lead to motor step loss. When reaching the reference switch, use the automatic position latch register in order to have an exact reference of where the stop switch became active.

Use hard stop only for emergency stop. After a hard stop, initiate a new homing sequence, because position might be lost.

Hint: There is no restriction of using a hard stop in conjunction with stallGuard2 (*sg_stop*=1). Hard stop should be used with stallGuard in any case, as a stall event means, that the motor is forced into stop.

11 stallGuard2 Load Measurement

stallGuard2 provides an accurate measurement of the load on the motor. It can be used for stall detection as well as other uses at loads below those which stall the motor, such as coolStep load-adaptive current reduction. The stallGuard2 measurement value changes linearly over a wide range of load, velocity, and current settings, as shown in Figure 11.1. At maximum motor load, the value goes to zero or near to zero. This corresponds to a load angle of 90° between the magnetic field of the coils and magnets in the rotor. This also is the most energy-efficient point of operation for the motor.

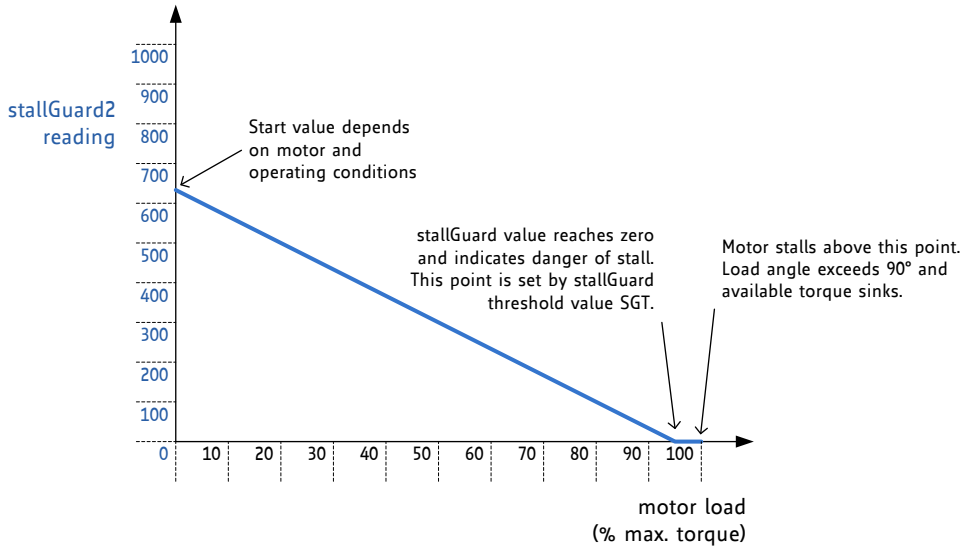


Figure 11.1 Function principle of stallGuard2

Parameter	Description	Setting	Comment
SGT	This signed value controls the stallGuard2 threshold level for stall detection and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value working with most motors. A higher value makes stallGuard2 less sensitive and requires more torque to indicate a stall.	0	indifferent value
		+1... +63	less sensitivity
		-1... -64	higher sensitivity
sfilt	Enables the stallGuard2 filter for more precision of the measurement. If set, reduces the measurement frequency to one measurement per electrical period of the motor (4 fullsteps).	0	standard mode
		1	filtered mode
Status word	**Description**	**Range**	**Comment**
SG	This is the *stallGuard2 result*. A higher reading indicates less mechanical load. A lower reading indicates a higher load and thus a higher load angle. Tune the *SGT* setting to show a *SG* reading of roughly 0 to 100 at maximum load before motor stall.	0... 1023	0: highest load low value: high load high value: less load

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

11.1 Tuning the 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*. Monitor *VACTUAL* to exceed the lower velocity threshold where stallGuard delivers a good result and enable *sg_stop* during this time only.

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 16.
For detail procedure see Application Note AN002 - *Parameterization of stallGuard2 & coolStep*

11.1.1 Variable Velocity Operation

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

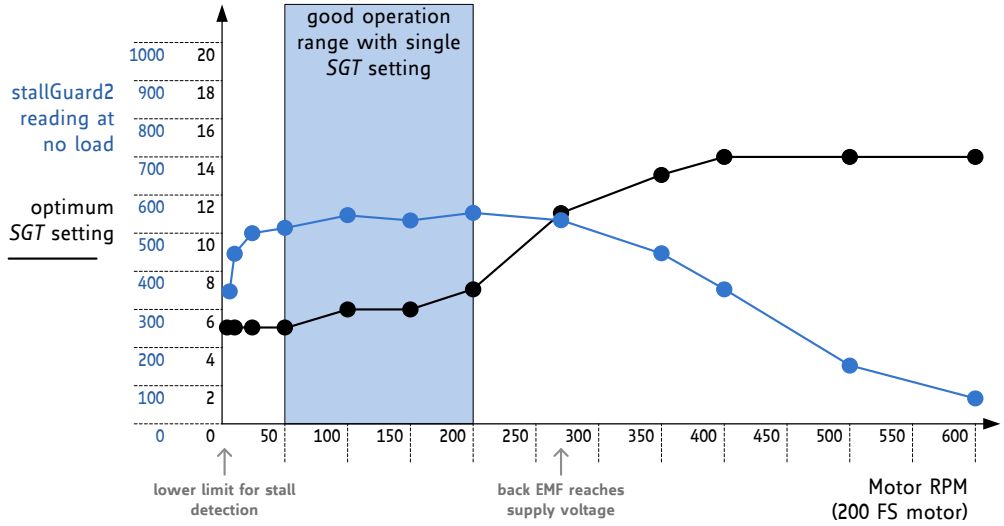


Figure 11.2 Example: Optimum SGT setting and stallGuard2 reading with an example motor

In many applications, operation at or near a single operation point is used most of the time and a single setting is sufficient. The ramp generator provides a lower and an upper velocity threshold to match this. The stall detection should be ignored and disabled by software outside the determined operation point, e.g. during acceleration phases preceding a sensorless homing procedure.

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

11.1.2 Small Motors with High Torque Ripple and Resonance

Motors with a high detent torque show an increased variation of the stallGuard2 measurement value SG with varying motor currents, especially at low currents. For these motors, the current dependency should be checked for best result.

11.1.3 Temperature Dependence of Motor Coil Resistance

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

11.1.4 Accuracy and Reproducibility of stallGuard2 Measurement

In a production environment, it may be desirable to use a fixed *SGT* value within an application for one motor type. Most of the unit-to-unit variation in stallGuard2 measurements results from manufacturing tolerances in motor construction. The measurement error of stallGuard2 – provided that all other parameters remain stable – can be as low as:

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

11.2 stallGuard2 Update Rate and Filter

The stallGuard2 measurement value *SG* is updated with each full step of the motor. This is enough to safely detect a stall, because a stall always means the loss of four full steps. In a practical application, especially when using coolStep, a more precise measurement might be more important than an update for each fullstep because the mechanical load never changes instantaneously from one step to the next. For these applications, the *sfilt* bit enables a filtering function over four load measurements. The filter should always be enabled when high-precision measurement is required. It compensates for variations in motor construction, for example due to misalignment of the phase A to phase B magnets. The filter should be disabled when rapid response to increasing load is required and for best results of sensorless homing using stallGuard.

11.3 Detecting a Motor Stall

For best stall detection, work without stallGuard filtering (*sfilt*=0). To safely detect a motor stall the stall threshold must be determined using a specific *SGT* setting. Therefore, the maximum load needs to be determined, which the motor can drive without stalling. At the same time, monitor the *SG* 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.

11.4 Homing with stallGuard

The homing of a linear drive requires moving the motor into the direction of a hard stop. As stallGuard needs a certain velocity to work, 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 as soon as the target velocity has been reached (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.

11.5 Limits of stallGuard2 Operation

stallGuard2 does not operate reliably at extreme motor velocities: Very low motor velocities (for many motors, less than one revolution per second) generate a low back EMF and make the measurement unstable and dependent on environment conditions (temperature, etc.). The automatic tuning procedure described above will compensate for this. Other conditions will also lead to extreme settings of *SGT* and poor response of the measurement value *SG* to the motor load.

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

12 coolStep Operation

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

12.1 User Benefits

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

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

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

12.2 Setting up for coolStep

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

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

Figure 12.1 shows the operating regions of coolStep:

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

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

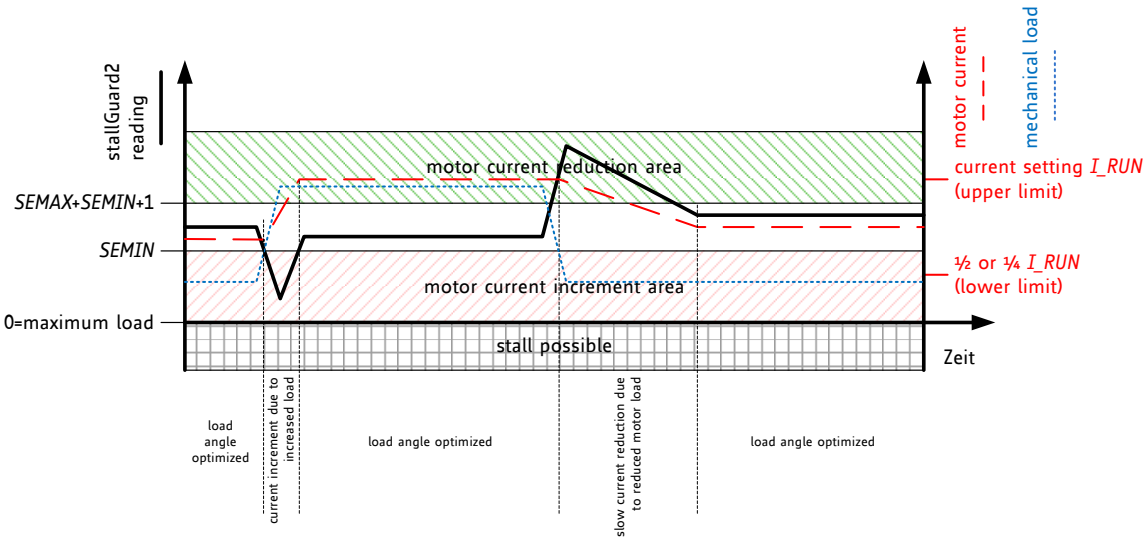


Figure 12.1 coolStep adapts motor current to the load

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

Parameter	Description	Range	Comment
SEUP	Sets the *current increment step*. The current becomes incremented for each measured stallGuard2 value below the lower threshold.	0…3	step width is 1, 2, 4, 8
SEDN	Sets the number of stallGuard2 readings above the upper threshold necessary for each *current decrement* of the motor current.	0…3	number of stallGuard2 measurements per decrement: 32, 8, 2, 1
SEIMIN	Sets the *lower motor current limit* for coolStep operation by scaling the *IRUN* current setting.	0	0: 1/2 of IRUN
		1	1: 1/4 of IRUN
VCOOL THRS	Lower ramp generator velocity threshold. Below this velocity coolStep becomes disabled. Adapt to the lower limit of the velocity range where stallGuard2 gives a stable result. *Hint:* May be adapted to disable coolStep during acceleration and deceleration phase by setting identical to *VMAX*.	1… 2^23	
VHIGH	Upper ramp generator velocity threshold value. Above this velocity coolStep becomes disabled. 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**
CSACTUAL	This status value provides the *actual motor current scale* as controlled by coolStep. The value goes up to the *IRUN* value and down to the portion of *IRUN* as specified by *SEIMIN*.	0…31	1/32, 2/32, … 32/32

12.3 Tuning coolStep

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

12.3.1 Response Time

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

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

12.3.2 Low Velocity and Standby Operation

Because coolStep is not able to measure the motor load in standstill and at very low RPM, a lower velocity threshold is provided 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.

13 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.

13.1 User Benefits

dcStep™

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

13.2 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.

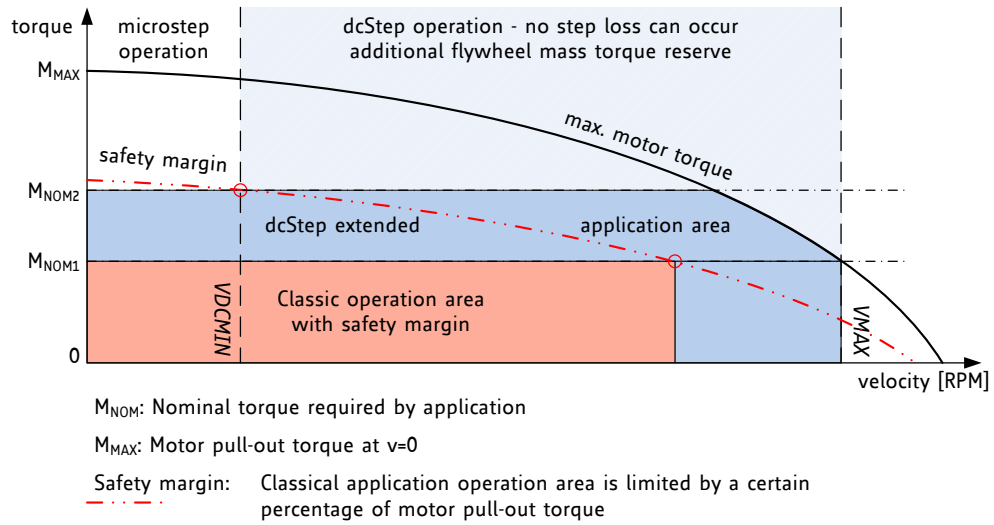


Figure 13.1 dcStep extended application operation area

Quick Start
For a quick start, see the Quick Configuration Guide in chapter 16.
For detail configuration procedure see Application Note AN003 - *dcStep*

13.3 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 *VACTUAL* or at reduced velocity if the motor becomes overloaded. It requires setting the minimum operation velocity *VDCMIN*. *VDCMIN* shall be set to the lowest operating velocity where dcStep gives a reliable detection of motor operation. The motor never stalls unless it becomes braked to a velocity below *VDCMIN*. In case the velocity should fall below this value, the motor would restart once its load is released, unless the stall detection becomes enabled (set *sg_stop*). Stall detection is covered by stallGuard2.

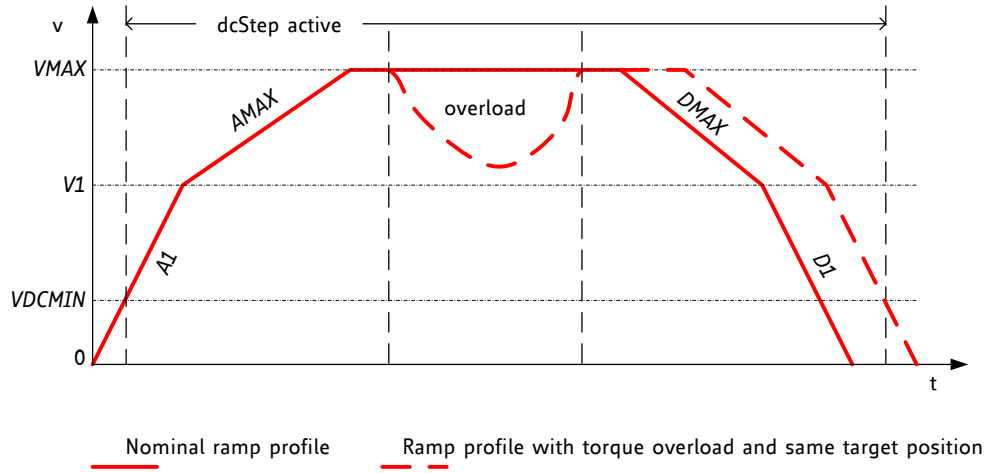


Figure 13.2 Velocity profile with impact by overload situation

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 14.2 for an initialization with the default table.

13.4 Stall detection in dcStep mode

While dcStep is able to decelerate the motor upon overload, it cannot avoid a stall in every operation situation. Once the motor is blocked, or it becomes decelerated below a motor dependent minimum velocity where the motor operation cannot safely be detected any more, the motor may stall and loose steps. In order to safely detect a step loss and avoid restarting of the motor, the stop on stall can be enabled (set flag *sg_stop*). In this case *VACTUAL* becomes set to zero once the motor is stalled. It remains stopped until reading the *RAMP_STAT* status flags. The flag *event_stop_sg* shows the active stop condition. A stallGuard2 load value is not available during dcStep operation. Before enabling stallGuard, make sure that the ramp generator has accelerated the motor to a velocity value *VACTUAL* slightly above *VDCMIN* or up to *VMAX*.

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

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

13.5 Measuring Actual Motor Velocity in dcStep Operation

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

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

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

Example:

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

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

14 Sine-Wave Look-up Table

Each of the TMC5062 drivers provides a programmable look-up table for storing the microstep current wave. As a default, the tables are 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.

14.1 User Benefits

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

14.2 Microstep Table

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

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

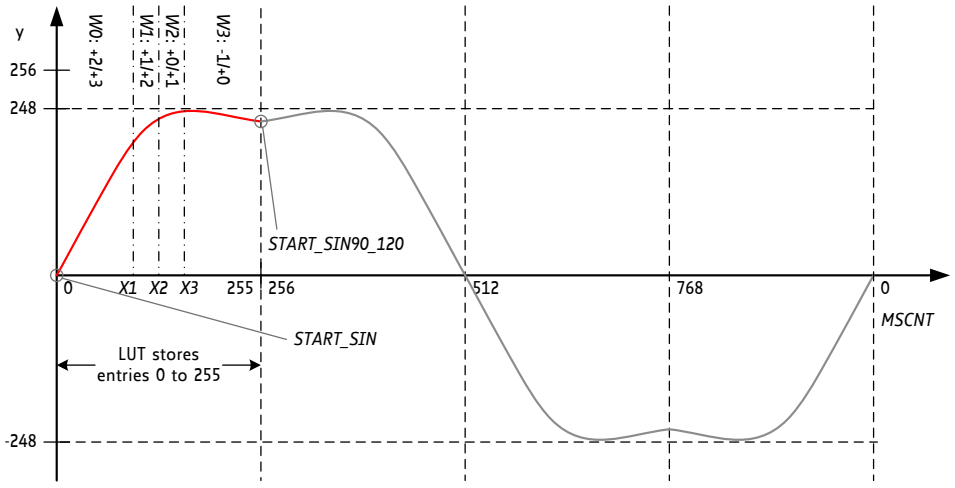


Figure 14.1 LUT programming example

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_120*. This register stores the resulting table entry for a phase shift of 90° for 2-phase stepper motors.

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 TMC5062-EVAL comes with a calculation tool for own waves.

Initialization example for the default microstep table:

MSLUTx[0]= %10101010101010101011010101010100 = 0xAAAAB554
MSLUTx[1]= %01001010100101010101010010101010 = 0x4A9554AA
MSLUTx[2]= %00100100010010010010100100101001 = 0x24492929
MSLUTx[3]= %00010000000100000100001000100010 = 0x10104222
MSLUTx[4]= %11111011111111111111111111111111 = 0xFBFFFFFF
MSLUTx[5]= %10110101101110110110111011101101 = 0xB5BB777D
MSLUTx[6]= %01001001001010010101010101010110 = 0x49295556
MSLUTx[7]= %00000000010000000100001000100010 = 0x00404222

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

MSLUTSTARTx= 0x00F70000:
START_SIN_0= 0, *START_SIN90_120*= 247

15 ABN Incremental Encoder Interface

The TMC5062 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.

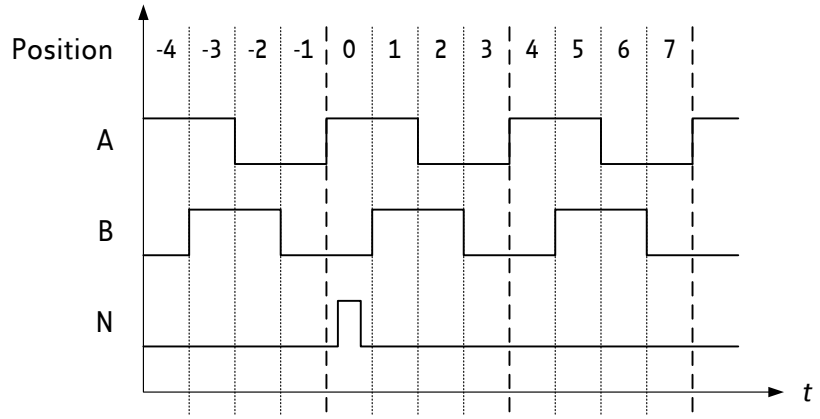


Figure 15.1 Outline of ABN signals of an incremental encoder

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 an signed fixed point number (16.16) to facilitate the generic adaption between motors and encoders. In decimal mode, the lower 16 bits represent a number between 0 and 9999. For stepper motors equipped with incremental encoders the fixed number representation allows very comfortable parameterization. Additionally, mechanical gearing can easily be taken into account. Negating the sign of *ENC_CONST* allows inversion of the counting direction to match motor and encoder direction.

Examples:

- Encoder factor of 1.0: *ENC_CONST* = 0x0001.0x0000 = FACTOR.FRACTION
- Encoder factor of -1.0: *ENC_CONST* = 0xFFFF.0x0000. This is the two's complement of 0x00010000. It equals (2^16-(FACTOR+1)).(2^16-FRACTION)
- Decimal mode encoder factor 25.6: 00025.6000 = 0x0019.0x1770 = FACTOR.DECIMALS
- Decimal mode encoder factor -25.6: 0xFFE6.4000 = 0xFFE6.0x0FAO. This equals (2^16-(FACTOR+1)).(10000-DECIMALS)

THE ENCODER COUNTER *X_ENC*

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

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.

15.1 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}		

15.2 Setting the Encoder to Match Motor Resolution

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

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

Example:

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

15.3 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.

16 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 SETTING UP SPREADCYCLE

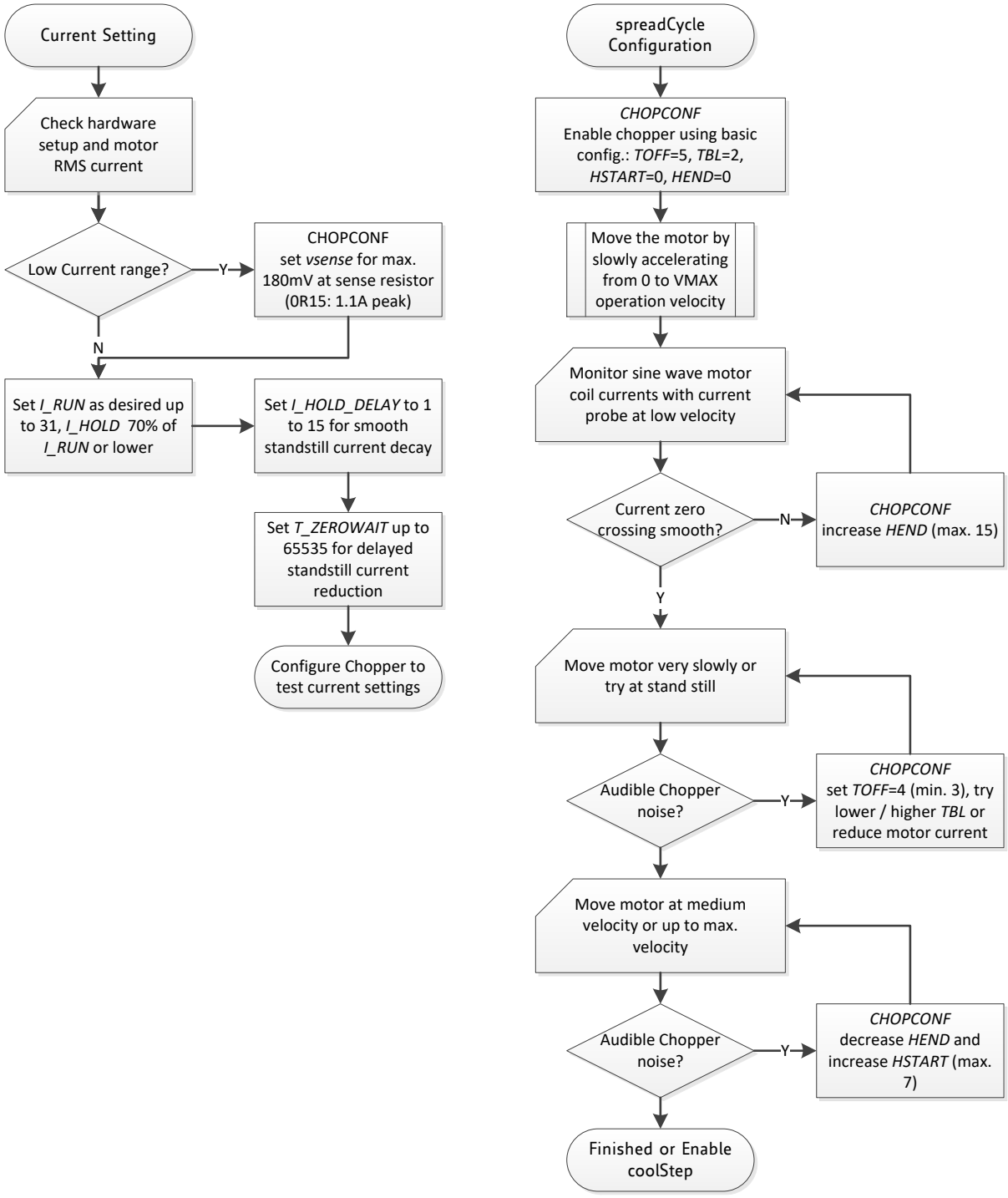


Figure 16.1 Current setting and setting up spreadCycle

MOVING THE MOTOR USING THE MOTION CONTROLLER

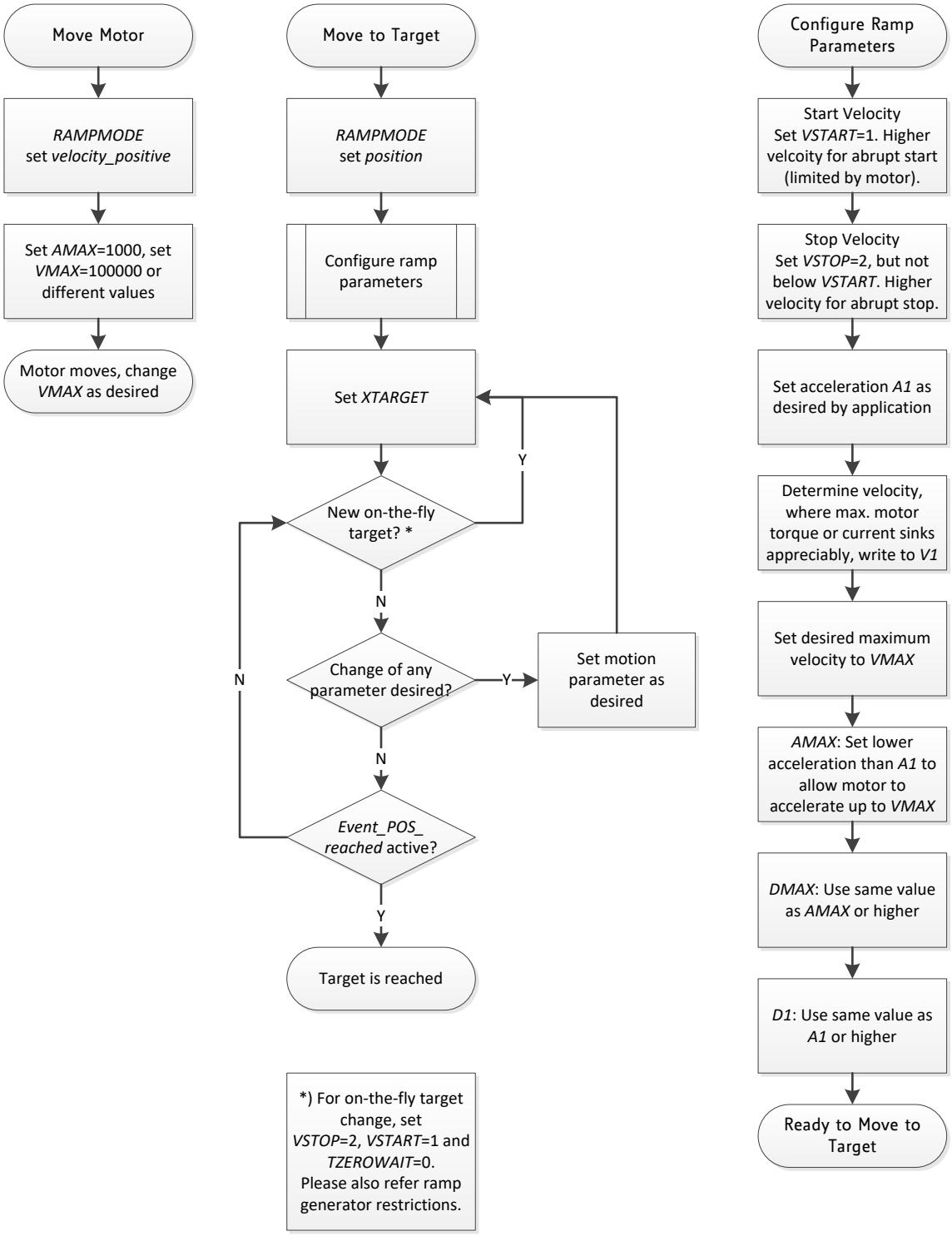


Figure 16.2 Moving the motor using the motion controller

ENABLING COOLSTEP (IN COMBINATION WITH SPREADCYCLE)

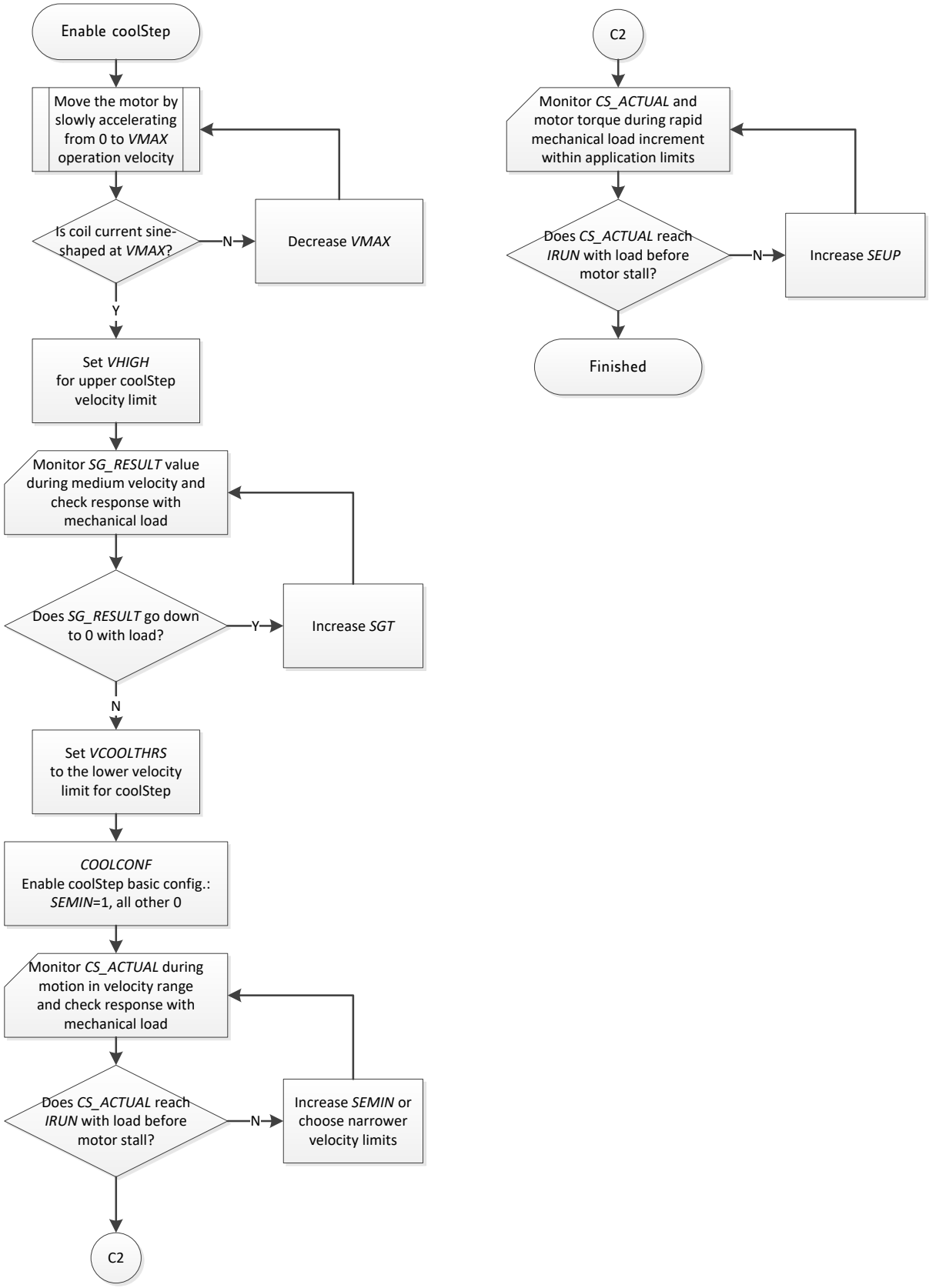


Figure 16.3 Enabling coolStep (in combination with spreadCycle)

SETTING UP DCSTEP

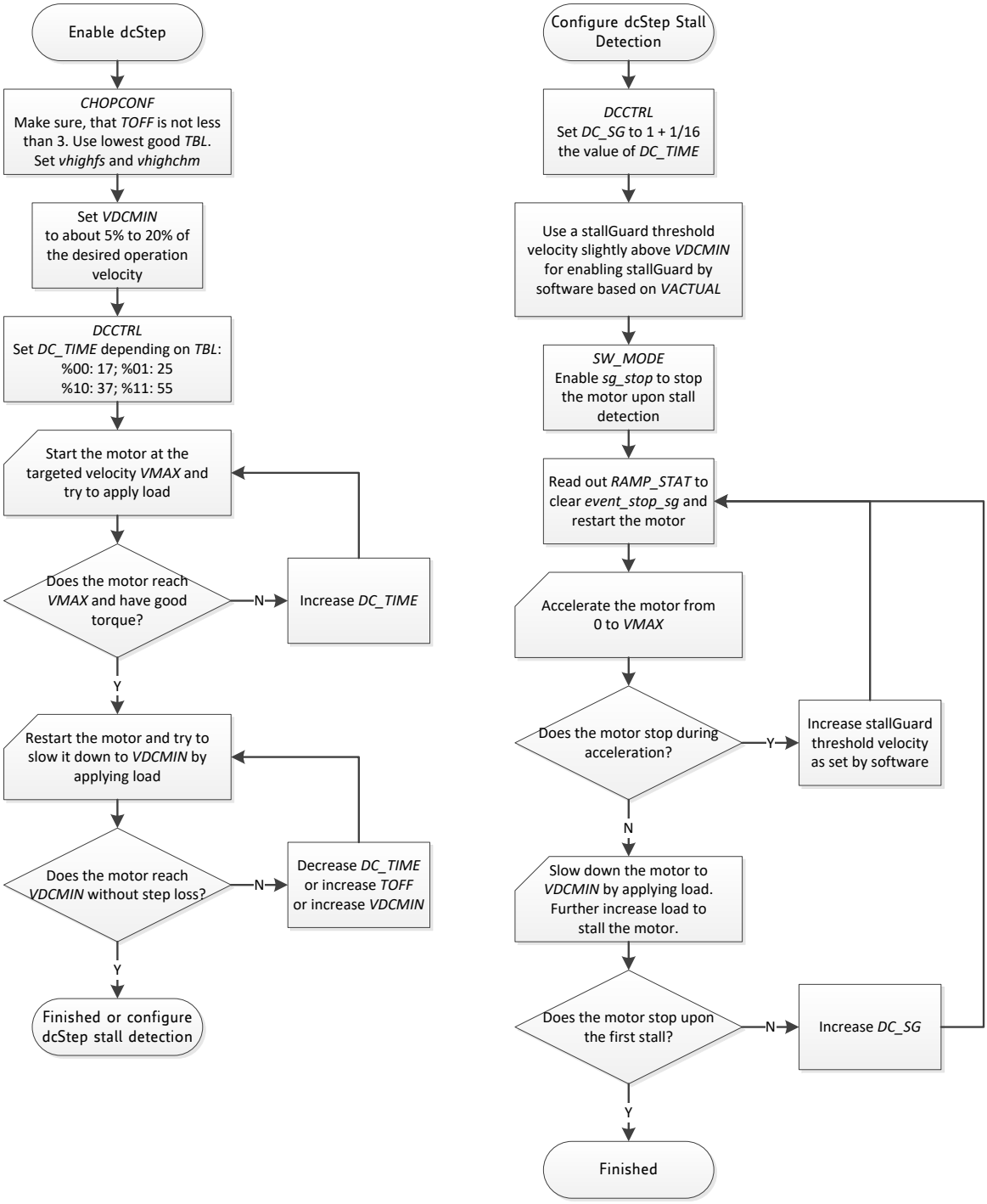


Figure 16.4 Setting up dcStep

17 Getting Started

Please refer to the TMC5062 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 16 will guide you through the process of correctly setting up all registers.

17.1 Initialization Examples

Initialization SPI datagram example sequence to enable and initialize driver 1 and ramp generator 1 to move the motor in velocity mode and read access 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: 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 TMC5062. It programs driver 1 and ramp generator 1 to move the motor in velocity mode and read accesses the position and actual velocity registers:

```
UART write: 0x05 0xEC 0x00 0x71 0x03 0x06 0x45;   // TOFF=6, HEND=6, SYNC=7, HSTR=0,
                                                  // TBL=2, MRES=0, CHM=0
UART write: 0x05 0xB0 0x00 0x01 0x14 0x05 0x47;   // IHOLD=5, IRUN=20, IHOLDDELAY=1
UART write: 0x05 0xA6 0x00 0x00 0x13 0x88 0xA4;   // AMAX=5000
UART write: 0x05 0xA7 0x00 0x00 0x4E 0x20 0x95;   // VMAX=20000
UART write: 0x05 0xA0 0x00 0x00 0x00 0x01 0xB3;   // RAMPMODE=1 (positive velocity)

// Now motor 1 should start rotating

UART write: 0x05 0x21 0x8D;                       // Query XACTUAL
UART read 7 bytes;
UART write: 0x05 0x22 0xC3;                       // Query VACTUAL
UART read 7 bytes;
```

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

18 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.

18.1 Using the Internal Clock

Directly tie the CLK input to GND near to the TMC5062 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 13.5. 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 *VMAX*=10000 and *AMAX*=60000 (for quick acceleration). The acceleration phase is ended after a few ms.
3. Read out *XACTUAL* twice, at time point t1 and time point t2, 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 *VMAX*=0.
5. The number of steps done in between of t1 and t2 now can be used to calculate the factor
$$f = \frac{VMAX * dt}{XACTUAL(t2) - XACTUAL(t1)} = \frac{1000}{XACTUAL(t2) - XACTUAL(t1)}$$
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.

18.2 Using an External Clock

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

18.3 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 TMC5062 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.

19 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}	-0.5	22	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

20 Electrical Characteristics

20.1 Operational Range

Parameter	Symbol	Min	Max	Unit
Junction temperature	T_J	-40	125	°C
Supply voltage (using internal +5V regulator)	V_{VS}	5.5	20	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}	4.75	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 $T_J \leq 105°C$, e.g. for 100ms short time acceleration phase below 50% duty cycle.	I_{Ox}		1.5	A

20.2 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} = 16.0V					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Supply current, driver disabled	I_{VS}	f_{CLK}=16MHz		30	40	mA
Supply current, operating	I_{VS}	f_{CLK}=16MHz, 40kHz chopper		33		mA
Static supply current	I_{VS0}	f_{CLK}=0Hz		7		mA
Supply current, driver disabled, dependency on CLK frequency	I_{VSX}	f_{CLK} variable, additional to I_{VS0}		1.6		mA/MHz
Internal current consumption from 5V supply on VCC pin	I_{VCC}	f_{CLK}=16MHz, 40kHz chopper		30	40	mA
IO supply current	I_{VIO}	no load on outputs, inputs at V_{IO} or GND		10		µA

Motor driver section	**DC- and Timing-Characteristics** V_{VS} = 16.0V					
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		120	250	ns
slope, MOSFET turning off	t_{SLPOFF}	measured at 700mA load current		220	450	ns
Current sourcing, driver off	I_{OIDLE}	O_{XX} pulled to GND	120	180	250	µA

Charge pump	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Charge pump output voltage	V_{VCP}-V_{VS}	operating, typical f_{chop}<40kHz	4.0	V_{5VOUT} - 0.4	V_{5VOUT}	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_{5VOUT}	I_{5VOUT} = 0mA T_J = 25°C	4.75	5.0	5.25	V
Output resistance	R_{5VOUT}	Static load		3		Ω
Deviation of output voltage over the full temperature range	$V_{5VOUT(DEV)}$	I_{5VOUT} = 30mA T_J = 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°C	8.8	12.4	17.9	MHz
Clock oscillator frequency	f_{CLKOSC}	t_J=50°C	9.4	13.2	18.8	MHz
Clock oscillator frequency	f_{CLKOSC}	t_J=150°C	9.6	13.4	18.9	MHz
External clock frequency (operating)	f_{CLK}		8	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}	25			ns

Detector levels	**DC-Characteristics**					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
V_{VSA} undervoltage threshold for RESET	V_{UV_VSA}	V_{VSA} rising	3.8	4.2	4.6	V
V_{5VOUT} undervoltage threshold for RESET	V_{UV_5VOUT}	V_{5VOUT} rising		3.5		V
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	°C
Overtemperature shutdown	t_{OT}	Temperature rising	135	150	170	°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		mΩ

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

20.3 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) One motor 0.71A RMS 85°C (93°C) Surface temperature at package center (peak surface temperature), board 55mm x 85mm, 25°C environment stealthChop or spreadCycle, sinewave, 20kHz chopper, 20V, 16MHz, internal supply for VCC Motors: QSH4218-035-10-027	3.7 2.4	W W
Typical power dissipation Two motors active	P_D	Two motors 0.71A RMS 113°C (119°C) Two motors 0.35A RMS 64°C (68°C)	3.7 1.4	W 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 TMC5062 power dissipation is available on www.trinamic.com.

21 Layout Considerations

21.1 Exposed Die Pad

The TMC5062 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.

21.2 Wiring GND

All signals of the TMC5062 are referenced to their respective GND. Directly connect all GND pins under the TMC5062 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 TMC5062. Optimally place them close to the TMC5062, 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.

21.3 Supply Filtering

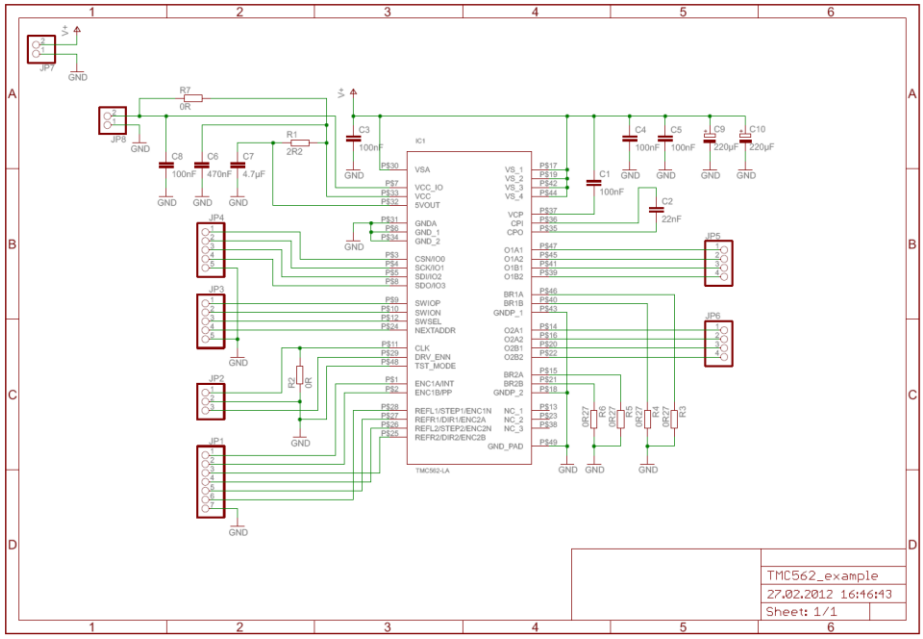
The 5VOUT output voltage ceramic filtering capacitor (4.7 µF recommended) should be placed as close as possible to the 5VOUT pin, with its GND return going directly to the GNDA pin. Use as short and as thick connections as possible. For best microstepping performance and lowest chopper noise an additional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

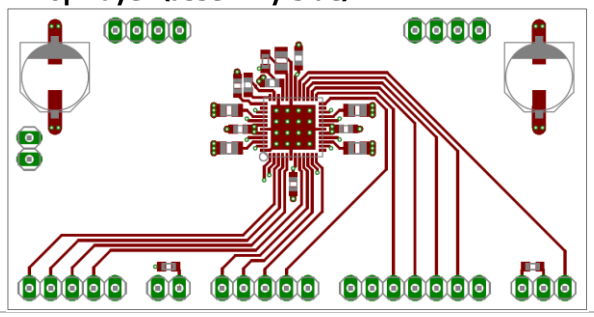
Take into account that the switching motor coil outputs have a high dV/dt. Thus capacitive stray into high resistive signals can occur, if the motor traces are near other traces over longer distances.

21.4 Layout Example

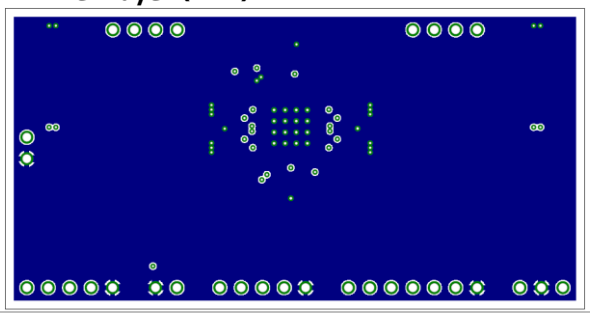
Schematic



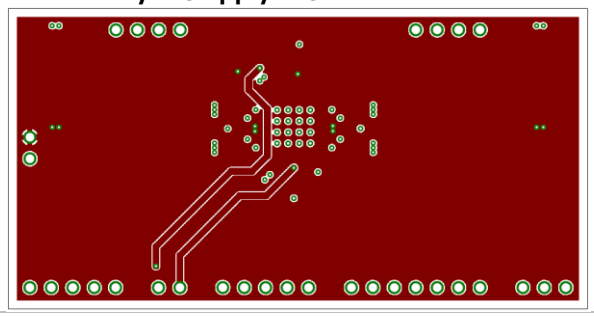
1- Top Layer (assembly side)



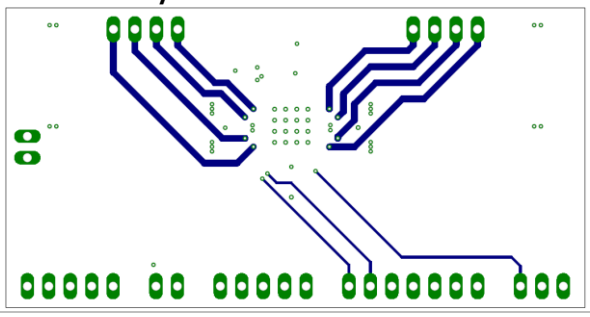
2- Inner Layer (GND)



3- Inner Layer (supply VS)



4- Bottom Layer



Components

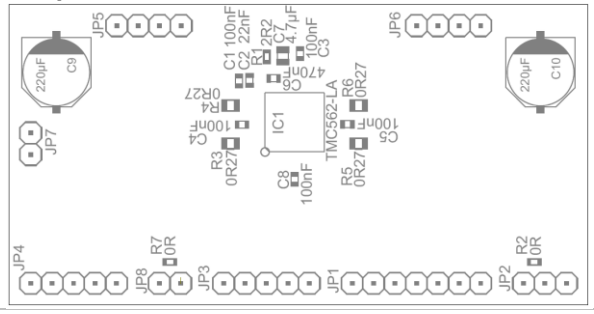


Figure 21.1 Layout example

22 Package Mechanical Data

22.1 Dimensional Drawings

Attention: Drawings not to scale.

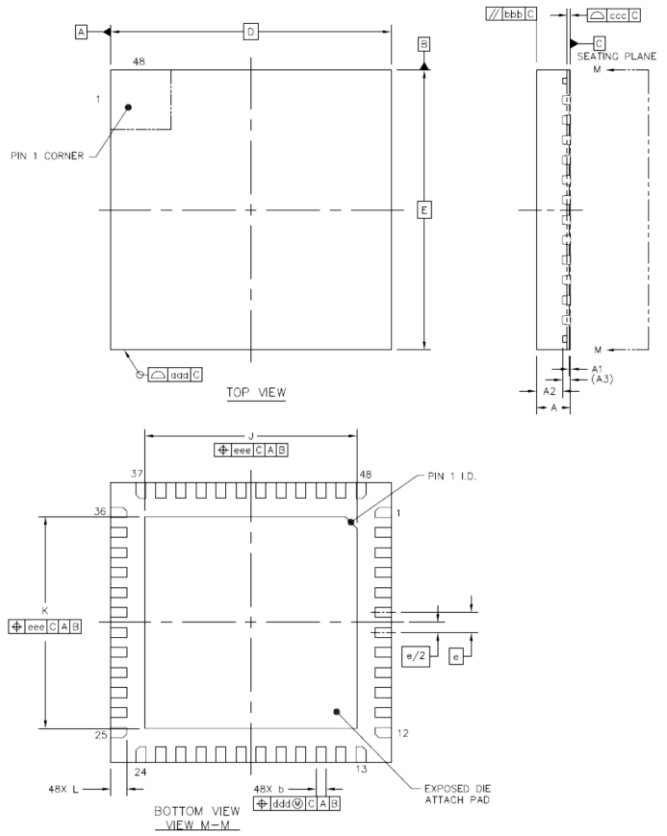


Figure 22.1 Dimensional drawings

Parameter [mm]	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

22.2 Package Codes

Type	Package	Temperature range	Code & marking
TMC5062-LA	QFN48 (RoHS)	-40°C ... +125°C	TMC5062-LA

23 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.

24 ESD Sensitive Device

The TMC5062 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.

25 Table of Figures

Figure 1.1 Basic application and block diagram 5
Figure 1.2 Energy efficiency with coolStep (example) 8
Figure 2.1 TMC5062 pin assignments 9
Figure 3.1 Standard application circuit 12
Figure 3.2 External supply of VCC_IO (showing optional filtering for VCC) 13
Figure 3.3 5V only operation 14
Figure 3.4 Using an external 5V supply to reduce linear regulator power dissipation 15
Figure 3.5 Using an external 5V supply to bypass internal regulator 15
Figure 3.6 Adding an 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 Connecting to a master via single wire UART interface 24
Figure 5.2 Connecting to a master via differential UART interface 24
Figure 8.1 Chopper phases 45
Figure 8.2 No ledges in current wave with sufficient hysteresis (magenta: current A, yellow & blue: sense resistor voltages A and B) 47
Figure 8.3 spreadCycle chopper scheme showing coil current during a chopper cycle 48
Figure 8.4 Classic const. off time chopper with offset showing coil current 49
Figure 8.5 Zero crossing with classic chopper and correction using sine wave offset 49
Figure 10.1 Ramp generator velocity trace showing consequent move in negative direction 54
Figure 10.2 Illustration of optimized motor torque usage with TMC5062 ramp generator 55
Figure 10.3 Ramp generator velocity dependent motor control 56
Figure 10.4 Using reference switches (example) 57
Figure 11.1 Function principle of stallGuard2 61
Figure 11.2 Example: Optimum SGT setting and stallGuard2 reading with an example motor 63
Figure 12.1 coolStep adapts motor current to the load 66
Figure 13.1 dcStep extended application operation area 68
Figure 13.2 Velocity profile with impact by overload situation 69
Figure 14.1 LUT programming example 71
Figure 15.1 Outline of ABN signals of an incremental encoder 73
Figure 16.1 Current setting and setting up spreadCycle 76
Figure 16.2 Moving the motor using the motion controller 77
Figure 16.3 Enabling coolStep (in combination with spreadCycle) 78
Figure 16.4 Setting up dcStep 79
Figure 21.1 Layout example 88
Figure 22.1 Dimensional drawings 89

26 Revision History

Version	Date	Author BD – Bernhard Dwersteg SD – Sonja Dwersteg	Description
1.04	2012_NOV-14	BD	First version of product TMC5062 datasheet based on TMC562 prototype datasheet. Features modified over previous prototype datasheet: - Adapted voltage rating to 16V operational, 18V max. - Added maximum ratings for short time current Feature set is application specific tailored compared to TMC562 engineering samples: no 3 phase support, dual motor operation only, single wire with single slave only (tie former address pin NEXTADDR to GND), use of internal sequencer only, 256 microsteps only *Former versions of the TMC562 datasheet are no longer valid for this product.*
1.05	2012-DEC-11	JP	Package Information updated.
1.06	2014-FEB-28	SD	- Chapter 20.3 (thermal characteristics) added. - Chapter 11.1 (tuning the stallGuard2 threshold) updated. - *CSACTUAL* in *DRV_STATUS* corrected (chapter 6.4.4). - Interrupt output remark in *RAMP_STAT* for *status_latch_l* and *status_latch_r* removed. Description *event_stop_l* and *event_stop_r* updated (chapter 6.2.2.2) - *SW_MODE* register updated (chapter 6.2.2.1). - Order codes updated. - New description of VCC_IO requirements (chapter 3.1.1). - *en_latch_encoder* updated. - Chapter 15 (ABN encoder information) updated. - Second SPI initialization example using ramp generator added. - Information about dcStep improved.
1.07	2014-MAY-12	SD	- Standard application circuit new (chapter 3.1): information about 3.3V operation added. - Motor current calculation updated
1.08	2014-JUL-01	BD	Integrated errata sheet V1.1 & workaround in 10.5
1.09	2015-MAR-23	BD	stallGuard Stop details: Improved homing algorithm, Added 11.4, Text for event_stop_sg, improved 13.4, Limits 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, added Quick configuration guide
1.10	2016-APR-28	BD	corrected TOFF calculation example, comments in GSTAT, comment on SPI_STATUS, 5V only +-5%, X1=128 in microstep table defaults, Setting negative encoder factors, Adaptation to internal fCLK, Interrupt handling, Wording V1 and VMAX register, ESD schematic w. varistors instead of snubber
1.11	2017-MAY-16	BD	Minor corrections

Table 26.1 Documentation revisions

27 References

[AN001] Trinamic Application Note 001 - Parameterization of spreadCycle™, www.trinamic.com
[AN002] Trinamic Application Note 002 - Parameterization of stallGuard2™ & coolStep™, www.trinamic.com
[AN027] Trinamic Application Note 027 - dcStep™ with TMC5062, www.trinamic.com
Calculation sheet TMC50XX_Calculations.xlsx