*Click **here** to ask an associate for production status of specific part numbers.*

TMC5240

36V $2A_{RMS}$+ Smart Integrated Stepper Driver and Controller

General Description

The TMC5240 is a smart high-performance stepper motor controller and driver IC with serial communication interfaces (SPI, UART) and extensive diagnostic capabilities. It combines a flexible, jerk-optimized ramp generator for automatic target positioning with industries' most advanced stepper motor driver based on the 256 microsteps, built-in indexer and fully integrated 36V, $3.0A_{MAX}$ H-Bridges plus non-dissipative integrated current sensing (ICS).

ADI-Trinamic's sophisticated StealthChop2 chopper ensures absolutely noiseless operation combined with maximum efficiency and best motor torque.

High integration, high energy efficiency, and a small form factor enable miniaturized and scalable systems for cost-effective solutions. The complete solution reduces the learning curve to a minimum while giving best-in-class performance.

The H-Bridge field-effect transistors (FETs) have very low impedance resulting in high driving efficiency and minimal heat generated. The typical total R_{ON} (high side + low side) is 0.23Ω.

The maximum RMS current per H-Bridge is I_{RMS} = $2.1A_{RMS}$ at room temperature, assuming a four-layer PCB.

The maximum output current per H-Bridge is I_{MAX} = $5.0A_{MAX}$ limited by the overcurrent protection (OCP).

Since this current is limited by thermal considerations, the actual maximum RMS current depends on the thermal characteristics of the application (PCB ground planes, heatsinks, ventilation, etc).

The maximum full-scale current per H-Bridge is I_{FS} = 3.0A and can be set by an external resistor connected to IREF. This current is defined as the maximum current setting of the embedded current drive regulation circuit.

The non-dissipative ICS eliminates the bulky external power resistors, resulting in a dramatic space and power saving compared with mainstream applications based on external sense resistors while providing the same overall accuracy.

The TMC5240 features abundant diagnostics and protections such as short protection/OCP, thermal shutdown, and undervoltage lockout (UVLO).

During thermal shutdown and UVLO events, the driver is disabled.

Furthermore, the TMC5240 provides functions to measure the driver temperature, estimate the motor temperature, and measure one external analog input.

The TMC5240 is available in a small TQFN32 5mm x 5mm package and a thermally optimized TSSOP38 9.7mm x 4.4mm with an exposed pad.

Applications

- Textile, Sewing Machines, Knitting Machines
- Lab and Factory Automation
- 3D Printers, ID Printers/Card Printers
- Liquid Handling, Medical Applications
- Office Automation and Paper Handling
- Point-of-Sale (POS), Massage Chairs
- Automated Teller Machine (ATM), Cash Recycler, Bill Validators, Cash Machines
- Closed-Circuit Television (CCTV), Security
- Pumps and Valve Control
- Heliostat and Antenna Positioning

Benefits and Features

- Voltage Range 4.5V to 36V DC
- Low $R_{DS(ON)}$ (HS + LS): 230 mΩ Typical (T_A = 25°C)
- Current Ratings per H-Bridge (Typical at 25°C):
  - I_{MAX} = 5.0A (Bridge Peak Current)
    - I_{RMS} = $2.1A_{RMS}$ (3A Sine Wave Peak)
- Fully Integrated Lossless Current Sensing
- Eight-Point Motion Controller for Minimum Jerk
- SPI and Single Wire UART
- Encoder Interface and 2x Reference Switch Input
- Highest Resolution 256 Microsteps per Full Step
- Flexible Wave Table and Phase Shift to Match Motor
- StealthChop2 Silent Motor Operation
- SpreadCycle Highly Dynamic Motor Control Chopper
- Jerk-Free Combination of StealthChop2 and SpreadCycle
- StallGuard2 and StallGuard4 Sensorless Motor Load Detection
- CoolStep Current Control for Energy Savings up to 75%
- Passive Braking and Freewheeling Mode
- Motor Phase Temperature Estimation
- Chip Temperature Measurement
- General Purpose Analog Input
- Full Protection and Diagnostics
- Overvoltage Protection Output
- Compact 5mm x 5mm TQFN32 Package or 9.7mm x 4.4mm TSSOP38

Ordering Information appears at end of data sheet.

19-101625C; Rev 1; 4/24

Simplified Block Diagram

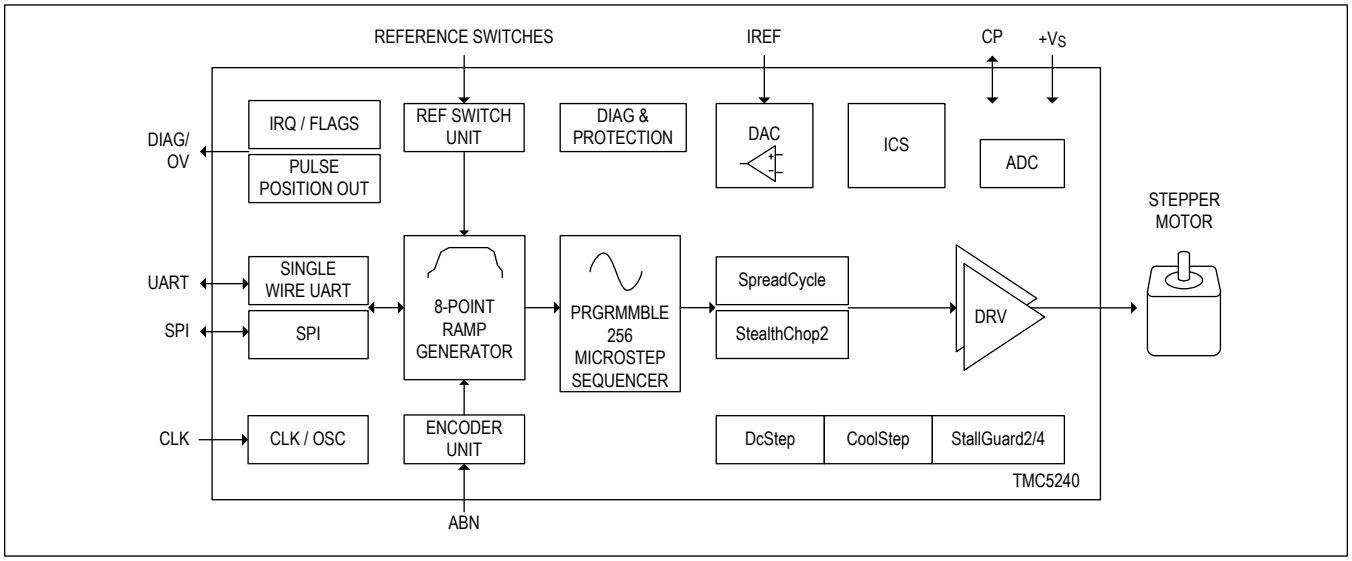


TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

LIST OF TABLES

Absolute Maximum Ratings

V_S to GND ..... -0.3V to 41V
V_{DD1V8} to GND ..... -0.3V to min (2.2, V_S + 0.3)V
AGND to GND ..... -0.3V to +0.3V
OUT1A, OUT2A, OUT1B, OUT2B ..... -0.3V to V_S + 0.3V
V_{CP} to GND ..... V_S - 0.3V to min (44, V_S + 6)V
CPO to GND ..... V_S - 0.3V to min (44, V_S + 6)V
CPI to GND ..... -0.3V to min (41, V_S + 0.3)V
SLEEPN to GND ..... -0.3V to V_S + 0.3V
IREF, AIN to GND ..... -0.3V to min (2.2, V_{DD1V8} + 0.3)V
V_{CC_IO} to GND ..... -0.3V to 5.5V
Logic Input/Output Voltage to GND ..... -0.3V to V_{CC_IO} + 0.3V
OV to GND ..... -0.3V to 6V
Operating Temperature Range ..... -40°C to 125°C
Junction Temperature ..... +165°C
Storage Temperature Range ..... -65°C to +150°C
Soldering Temperature (reflow) ..... +260°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Package Information

TQFN32 5mm x 5mm

Package Code	T3255+5C
Outline Number	21-0140
Land Pattern Number	90-0013
Thermal Resistance, Single-Layer Board:	
Junction to Ambient (θ_{JA})	47°C/W
Junction to Case (θ_{JC})	1.7°C/W
Thermal Resistance, Four-Layer Board:	
Junction to Ambient (θ_{JA})	29°C/W
Junction to Case (θ_{JC})	1.7°C/W

TSSOP38 9.7mm x 4.4mm EP

Package Code	U38E+3C
Outline Number	21-0714
Land Pattern Number	90-0435
Thermal Resistance, Four-Layer Board:	
Junction to Ambient (θ_{JA})	25°C/W
Junction to Case (θ_{JC})	1°C/W

For the latest package outline information and land patterns (footprints), go to ***www.maximintegrated.com/packages***. Note that a "+", "#", or "-" in the package code indicates RoHS status only. Package drawings may show a different suffix character, but the drawing pertains to the package regardless of RoHS status.

Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to ***www.maximintegrated.com/thermal-tutorial***.

Electrical Characteristics

(V_S = 4.5V to 36V, R_{REF} = from 12kΩ to 24kΩ , Typical values assume T_A = 25ºC and V_S = 24V, Limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
POWER SUPPLY						
Supply Voltage Range	V_S		4.5		36	V
Sleep Mode Current Consumption	I_{VS}	V(SLEEPN) = 0		4	18	µA

Electrical Characteristics (continued)

(V_S = 4.5V to 36V, R_{REF} = from 12kΩ to 24kΩ , Typical values assume T_A = 25ºC and V_S = 24V, Limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Quiescent Current Consumption	I_{VS}	V(SLEEPN) = 1, V(DRV_ENN) = 1		3.5	5	mA
1.8V Regulator Output Voltage	V_{VDD}	V_S = 4.5V		1.8		V
V_{DD} Current Limit	$IV18_{LIM}$		20			mA
Charge Pump Voltage	V_{CP}			V_S + 2.7		V
Logic I/O Supply Voltage Range	V_{CC_IO}		2.2		5.5	V
Sleep Mode Current Consumption	I_{VCC_IO}	V(SLEEPN) = 0		5	10	µA
Quiescent Current Consumption	I_{VCC_IO}	V(SLEEPN) = 1		35	60	µA
LOGIC LEVEL INPUTS-OUTPUTS						
Input Voltage Level - High	V_{IH}		0.7 x V_{CC_IO}			V
Input Voltage Level - Low	V_{IL}				0.3 x V_{CC_IO}	V
Input Hysteresis	V_{HYS}			0.15 x V_{CC_IO}		V
Internal Pullup/Pulldown Resistance	R_{PULL}	to GND or to V_{CC_IO}	60	100	140	kΩ
Input Leakage	In_{Leak}	Inputs without pullup/pulldown resistance	-1		+1	µA
Output Logic-Low Voltage	V_{OL}	I_{LOAD} = 5mA			0.4	V
Push-Pull Output Logic-High Voltage	V_{OH}	I_{LOAD} = 5mA	V_{CC_IO} - 400mV			
Open-Drain Output Logic High Leakage Current	I_{OH}	V(PIN) = 5.5V	-1		+1	µA
SLEEPN Voltage Level High	VIH_{SLEEPN}		0.9			V
SLEEPN Voltage Level Low	VIL_{SLEEPN}				0.6	V
SLEEPN Pulldown Input Resistance	RPD_{SLEEPN}		0.8	1.5		MΩ
OUTPUT SPECIFICATIONS						
Output ON-Resistance Low Side	RON_{LS}	Full-scale bits = 10		0.11	0.2	Ω
		Full-scale bits = 01		0.15	0.28	
Output ON-Resistance Low Side	RON_{LS}	Full-scale bits = 00		0.28	0.54	Ω
Output ON-Resistance High Side	RON_{HS}			0.12	0.22	Ω
Output Leakage	I_{LEAK}		-5		+5	µA

Electrical Characteristics (continued)

(V_S = 4.5V to 36V, R_{REF} = from 12kΩ to 24kΩ , Typical values assume T_A = 25ºC and V_S = 24V, Limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Output Slew Rate	SR	Slew-rate bits = 00		100		V/µs
		Slew-rate bits = 01		200		
		Slew-rate bits = 10		400		
		Slew-rate bits = 11		800		
PROTECTION CIRCUITS						
Overcurrent Protection Threshold	OCP	Full-scale bits = 10	5.0			A
		Full-scale bits = 01	3.33			
		Full-scale bits = 00	1.67			
Overcurrent Protection Blanking Time	T_{OCP}		0.9	1.5	2.3	µs
UVLO Threshold on V_S	UVLO	V_S falling	3.75	3.9	4.05	V
UVLO Threshold on V_S Hysteris	UVLOHYS			0.12		V
UVLO Threshold on V_{CC_IO}	UVLO	V_{CC_IO} falling	0.9	1.5	1.95	
V_{CC_IO} UVLO Hysteresis	UVLOVCCH			100		mV
Thermal Protection Threshold Temperature	TSD			165		°C
Thermal Protection Temperature Hysteresis				20		°C
CURRENT REGULATION						
IREF Pin Resistor Range	R_{REF}		12		60	kΩ
IREF Output Voltage	V_{REF}		0.882	0.9	0.918	V
Full-Scale Current Constant	KIFS	IFS = 1A		11.75		A x kΩ
Full-Scale Current Constant	KIFS	IFS = 2A		24		A x kΩ
Full-Scale Current Constant	KIFS	IFS = 3A		36		A x kΩ
Regulation Accuracy	DITRIP1	Output current from 7% to 100% FS, R_{REF} = 12kΩ	-5		+5	%
FUNCTIONAL TIMINGS						
SLEEP Time	t_{SLEEP}	SLEEPN = 0 to OUT_ three state			50	µs
Wake-Up Time from Sleep	TWAKE	SLEEPN = 1 to normal operation			2.5	ms
Enable Time	TEN	Time from DRV_ENN pin falling edge to driver on			1.5	µs
Disable Time	TEN	Time from DRV_ENN pin rising edge to driver off			6	µs

Electrical Characteristics (continued)

(V_S = 4.5V to 36V, R_{REF} = from 12kΩ to 24kΩ , Typical values assume T_A = 25ºC and V_S = 24V, Limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
CLOCK						
Internal Clock Frequency	f_{CLKOSC}		11.9	12.5	13.2	MHz
External Clock Frequency	f_{CLK}		8	16	20	MHz
External Clock Duty-Cycle	t_{CLKL}		40		60	%
External Clock Detection in Cycles			4		8	
External Clock Timeout Detection in Cycles of Internal f_{CLKOSC}			12		16	
External Clock Detection Lower Frequency Threshold	f_{CLKLO}		4			MHz
SPI TIMINGS						
SCK Valid Before or After Change of CSN	t_{CC}		T_{SCLK}			ns
CSN High Time	t_{CSH}		4 x T_{CLK}			ns
SCK Low Time	t_{CL}		20			ns
SCK High Time	t_{CH}		20			ns
SCK Frequency	f_{SCK}				10	MHz
SDI Setup Time Before SCK Rising Edge	t_{DU}		10			ns
SDI Hold Time After SCK Rising Edge	t_{DH}		10			ns
Data Out Valid Time After SCK Falling Edge	t_{DO}	V_{CC_IO} = 3.3V		27	40	ns
SDI, SCK, and CSN Filter Delay Time	t_{FILT}	Rising and falling edge		10		ns
ENCODER TIMING						
Encoder Counting Frequency	f_{CNT}			< 2/3 f_{CLK}	f_{CLK}	
A/B/N Input Low Time	t_{ABNL}		$3t_{CLK}$ + 20			ns
A/B/N Input High Time	t_{ABNH}		$3t_{CLK}$ + 20			ns
A/B/N Spike Filtering Time	$t_{FILTABN}$	Rising and falling edge		$3t_{CLK}$		
ADC/Analog Input/Temperature						
ADC Resolution		12 bit + sign		13		Bit
Analog Input Voltage Range	V_{AIN}		0		1.25	V
Analog Input Leakage	$I_{AIN,leak}$		-1		+1	uA

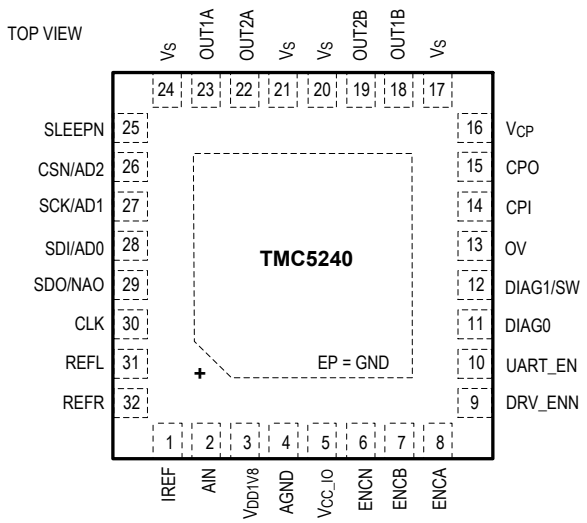
Electrical Characteristics (continued)

(V_S = 4.5V to 36V, R_{REF} = from 12kΩ to 24kΩ , Typical values assume T_A = 25ºC and V_S = 24V, Limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Analog Input Frequency	f_{AIN}	Assuming undersampling at AIN is accepted, the AIN input frequency needs to be lower than the given max value for a meaningful ADC conversion for a single ADC channel.			70	kHz
Driver Temperature Accuracy	T_{DRIVER}			±10		°C
Supply Voltage Measurement Accuracy			-5		+5	%
ADC Sample Rate	$f_{SAMPLE, ADC}$			$f_{CLK} / 2048$		

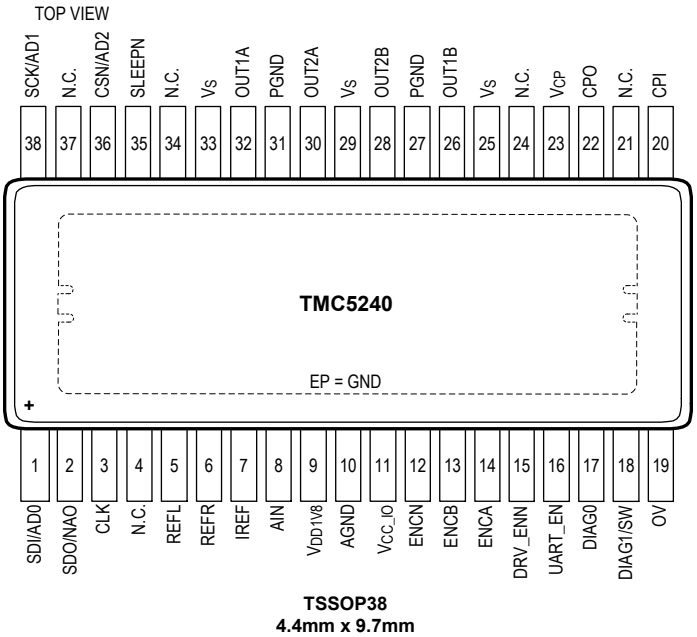
Pin Configurations

TMC5240 TQFN Pin Configuration



TQFN
5mm x 5mm

TMC5240 TSSOP Pin Configuration



TSSOP38
4.4mm x 9.7mm

Pin Description

PIN		NAME	FUNCTION	REF SUPPLY	TYPE
TQFN32	TSSOP38				
4	10	AGND	Analog Ground. Connect to ground plane.		GND
—	27, 31	PGND	Power ground. Connect to ground plane.		GND
17, 20, 21, 24	25, 29, 33	V_S	Motor supply voltage. Provide filtering capacity near pin with shortest loop to GND plane/exposed pad.		Supply
3	9	V_{DD1V8}	Output of internal 1.8V regulator. Attach 2.2µF or larger ceramic capacitor to AGND near to pin for best performance.		Supply
16	23	V_{CP}	Charge pump voltage. Tie to V_S using 1.0µF capacitor. Connect positive end of capacitor close to V_S pin to avoid inductive peaks.		Analog Output
5	11	V_{CC_IO}	Digital IO supply voltage provided from external source to define circuit IO level. Required for proper voltage level settings on output pins.	V_{CC_IO}	Analog Input
15	22	CPO	Charge pump capacitor output.		Analog Output
14	20	CPI	Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.		Analog Output
30	3	CLK	CLK input. Tie to GND using short wire for internal clock or supply external clock. Internal clock-fail over circuit protects against loss of external clock signal.	V_{CC_IO}	Digital Input
31	5	REFL	Left reference input for internal ramp generator	V_{CC_IO}	Digital Input
32	6	REFR	Right reference input for internal ramp generator	V_{CC_IO}	Digital Input
26	36	CSN/AD2	SPI chip select input (negative active) (UART_EN = 0) or Address input 2 (+4) in UART mode (UART_EN = 1).	V_{CC_IO}	Digital Input (pull up)
27	38	SCK/AD1	SPI serial clock input (UART_EN = 0) or address input 1 (+2) in UART mode (UART_EN = 1).	V_{CC_IO}	Digital Input (pull up)
28	1	SDI/AD0	SPI data input (UART_EN = 0) or address input 0 (+1) in UART mode (UART_EN = 1).	V_{CC_IO}	Digital Input (pull up)
29	2	SDO/NAO	SPI data output (three-state) (UART_EN = 0) or next address output (NAO) in UART mode (UART_EN = 1).	V_{CC_IO}	Digital Output
1	7	IREF	Analog reference current for current scaling. Provide external resistor to GND.	V_{CC_IO}	Analog Input
10	16	UART_EN	Interface selection pin. When tied low, the SPI interface is enabled. When tied high, the UART interface is enabled. Integrated pull-down resistor.	V_{CC_IO}	Digital Input (pull down)
7	13	ENCB	Encoder B-channel input.	V_{CC_IO}	Digital Input (pull up)

Pin Description (continued)

PIN		NAME	FUNCTION	REF SUPPLY	TYPE
TQFN32	TSSOP38				
8	14	ENCA	Encoder A-channel input.	V_{CC_IO}	Digital Input (pull up)
6	12	ENCN	Encoder N-channel input.	V_{CC_IO}	Digital Input (pull up)
9	15	DRV_ENN	Enable input. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level.	V_{CC_IO}	Digital Input (pull up)
11	17	DIAG0	Diagnostics output DIAG0. Interrupt or STEP output from internal motion controller for external driver. Use external pullup resistor in open drain mode. In system reset state, this pin is actively pulled low to indicate reset condition to external controller.	V_{CC_IO}	Digital Output
12	18	DIAG1/SW	Diagnostics output DIAG1. Position compare or DIR output from internal motion controller for external driver. Use external pullup resistor in open-drain mode. Single-wire I/O in UART mode.	V_{CC_IO}	Digital IO
25	35	SLEEPN	Low active power down input/reset input. Apply a continuous low level to bring the device to sleep mode. SLEEPN has an internal pull-down. If not used connect to V_S or V_{CC_IO} (this is a high voltage pin). Once the IC returns from sleep mode/reset, it must be reconfigured before being used again. Register content is not stored during sleep mode. While re-configuring the IC it is advised to still hold the bridge drivers disabled with DRV_ENN. Do not use while at high motor velocity!	V_S	Analog Input (pull down)
19	28	OUT2B	Motor coil B output 2	V_S	Analog Output
18	26	OUT1B	Motor coil B output 1	V_S	Analog Output
22	30	OUT2A	Motor coil A output 2	V_S	Analog Output
23	32	OUT1A	Motor coil A output 1	V_S	Analog Output

Pin Description (continued)

PIN		NAME	FUNCTION	REF SUPPLY	TYPE
TQFN32	TSSOP38				
EP	EP	GND	Exposed die pad. Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane. Serves as GND pin for power stage and internal circuitry.		GND
—	4, 21, 24, 34, 37	N.C.	No internal connection. Leave this pin open or tie it to GND for improved cooling.		N.C.
13	19	OV	Overvoltage indicator output (open-drain) with programmable threshold voltage. Attach external MOSFET with load resistor to limit supply voltage. External pullup resistor required. Updated by ADC with $f_{CLK} / 2048$.	V_{CC_IO}	Digital Output (open drain)
2	8	AIN	General-purpose analog input measured with internal ADC with $f_{CLK} / 2048$. Input range 0 to 1.25V. Value available through SPI/UART.	V_{CC_IO}	Analog Input

Functional Diagrams

TMC5240

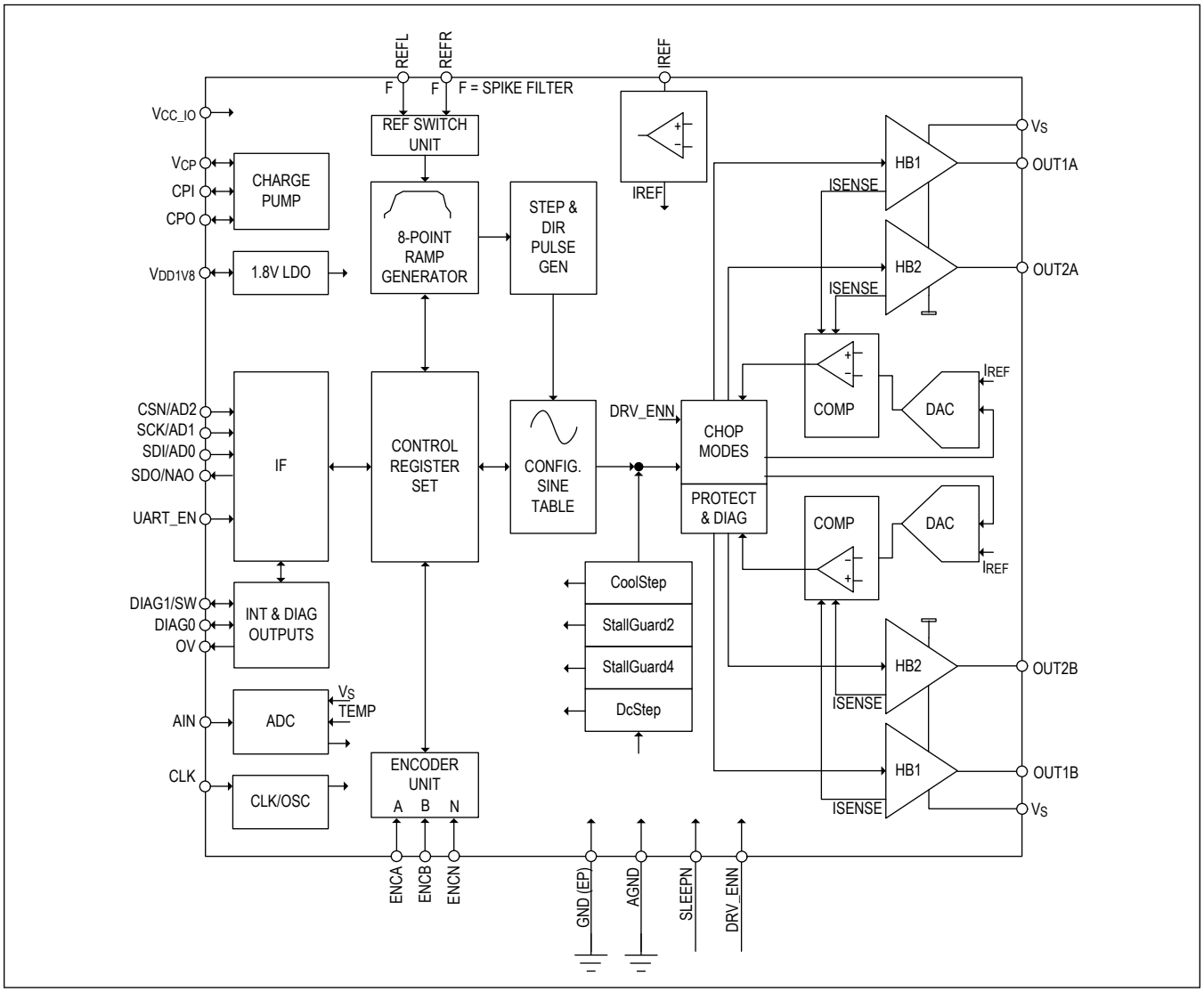


Figure 1. Block Diagram

Detailed Description

Principles of Operation

Full Featured Motion Controller and Driver

The TMC5240 motion controller and driver chip is an intelligent power component interfacing between CPU and stepper motor. All stepper motor logic is completely within the TMC5240. No software is required to control the motor — just provide target positions. The TMC5240 offers numerous unique enhancements, which are enabled by the system-on-chip integration of driver and controller. The eight-point ramp generator of the TMC5240 automatically uses StealthChop, CoolStep, StallGuard, DcStep, and SpreadCycle to optimize every motor movement.

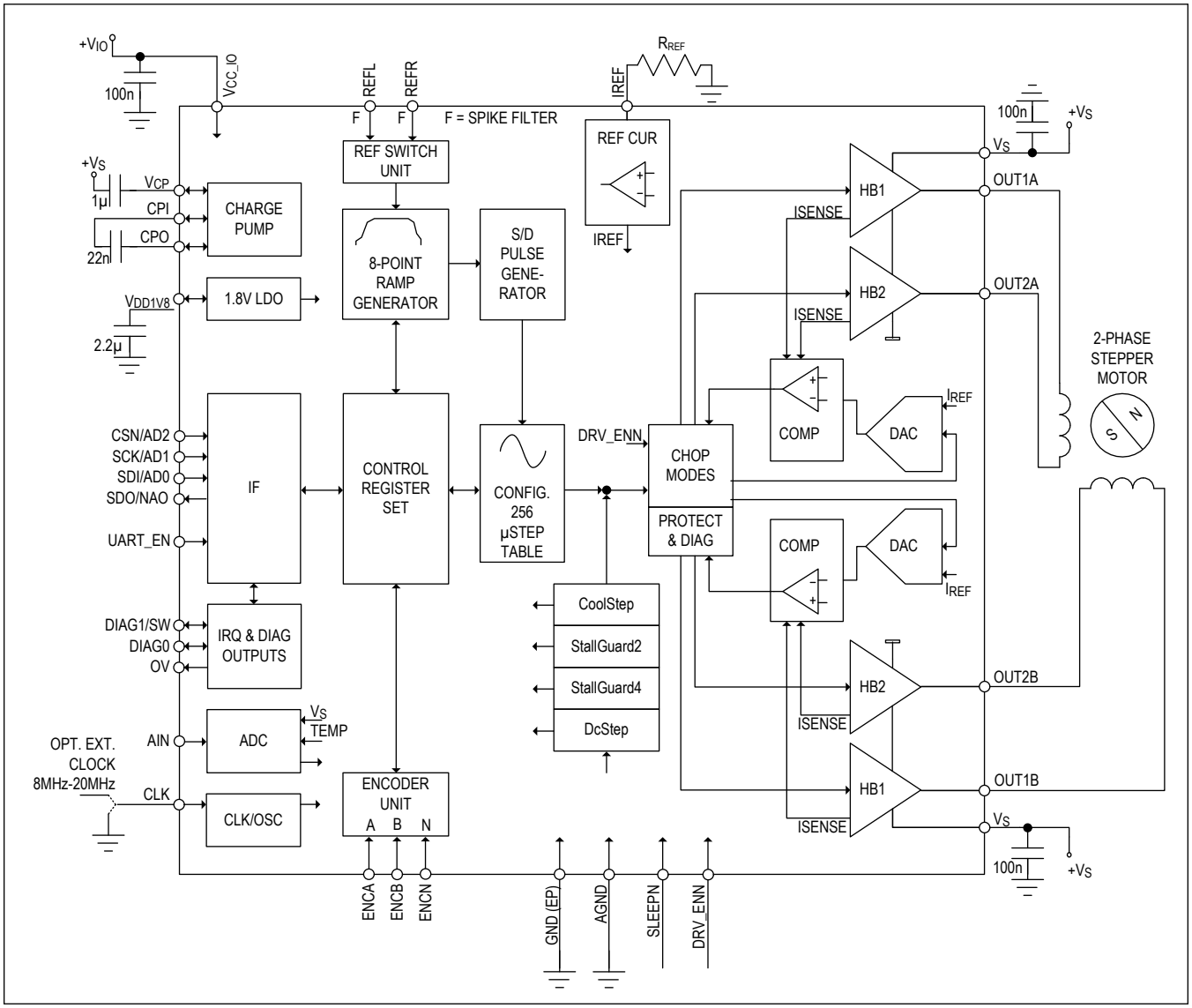


Figure 2. Block Diagram with Typical External Components

Key Concepts

The TMC5240 implements advanced features, which are exclusive to ADI-Trinamic products. These features contribute toward greater precision, greater energy efficiency, higher reliability, smoother motion, and cooler operation in many stepper motor applications.

StealthChop2	No-noise, high-precision chopper algorithm for inaudible motion and inaudible standstill of the motor. Allows faster motor acceleration and deceleration than StealthChop and extends StealthChop to low standstill motor currents.
SpreadCycle	High-precision cycle-by-cycle current control for highest dynamic movements.
StallGuard2	Sensorless stall detection and mechanical load measurement for SpreadCycle.
StallGuard4	Sensorless stall detection and mechanical load measurement for StealthChop.
CoolStep	Uses StallGuard measurement in order to adapt the motor current for best efficiency and lowest heat-up of motor and driver.

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

Control Interfaces

The TMC5240 supports both, an SPI and a UART-based single-wire interface with CRC checking. Selection of the actual interface combination is done through the UART_EN pin, which can be hardwired to GND or V_{CC_IO} depending on the desired interface selection.

The SPI is a bit-serial interface synchronous to a bus clock. For every bit sent from the bus controller to the bus peripheral, another bit is sent simultaneously from the peripheral back to the controller. Communication between an SPI controller (example, an MCU) and the peripheral always consists of sending one 40-bit command word and receiving one 40-bit status word.

The single-wire interface allows a bidirectional single-wire interfacing. It can be driven by any standard UART. No baud rate configuration is required.

Integrated Eight-Point Motion Controller

The integrated 32-bit motion controller automatically drives the motor to target positions or accelerates to target velocities. All motion parameters can be changed on the fly. The motion controller recalculates immediately. A minimum set of configuration data consists of acceleration and deceleration values and the maximum motion velocity. The start and stop velocities are supported as well as a second and third acceleration and deceleration setting selected by velocity thresholds resulting in an eight-point velocity profile. These settings allow adaptation of the motion profile to the motor torque profile as well as jerk reduction for near S-ramp performance. The integrated motion controller supports immediate reaction to mechanical reference switches and the sensorless stall detection StallGuard2 and StallGuard4.

Benefits

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

Automatic Standstill Power Down

An automatic current reduction drastically reduces application power dissipation and cooling requirements. A reduction to half of the run current reduces standstill power dissipation to roughly 25%. Standstill current, delay time, and decay parameters can be configured through the serial control interfaces.

Automatic freewheeling and passive motor braking are provided as an option for standstill. Passive braking reduces motor standstill power consumption to zero, while still providing effective dampening and braking!

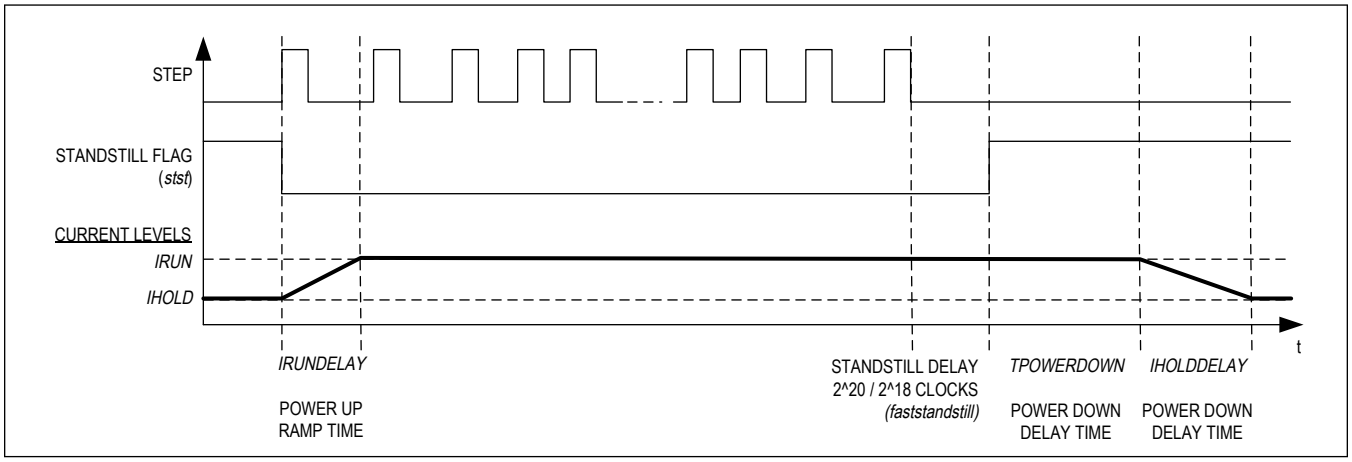


Figure 3. Automatic Motor Current Control at Standstill and Ramp-Up

StealthChop2 and SpreadCycle Driver

StealthChop2 is a voltage chopper-based principle. It especially guarantees that the motor is absolutely quiet in standstill and in slow motion, except for noise generated by ball bearings.

Unlike other voltage mode choppers, StealthChop2 does not require any configuration. It automatically learns the best settings during the first motion after power up and further optimizes the settings in subsequent motions.

An initial homing sequence is sufficient for learning. Optionally, initial learning parameters can be loaded to the register set. StealthChop2 allows high motor dynamics by reacting at once to a change of motor velocity.

For highest velocity applications, SpreadCycle is an alternative option to StealthChop2. StealthChop2 and SpreadCycle may even be used in a combined configuration for the best of both worlds: StealthChop2 for no-noise standstill, silent, and smooth performance, SpreadCycle at higher velocity for high dynamics and highest peak velocity at low vibration.

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

Benefits

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

StallGuard2/4 – Mechanical Load Sensing

StallGuard2 and StallGuard4 provide 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. While StallGuard2 combines with SpreadCycle chopper, StallGuard4 uses a different principle to combine with StealthChop2.

CoolStep – Load Adaptive Current Control

CoolStep drives the motor at the optimum current. It uses the StallGuard2 or StallGuard4 load measurement information to adjust the motor current to the minimum amount required in the actual load situation.

CoolStep results in energy savings and keeps the components cool. Due to driving the motor with the optimum current, CoolStep increases the motor efficiency compared to standard operation with approximately 50% torque reserve.

Benefits

- Highest energy efficiency, power consumption decreased by up to 75%
- Motor generates less heat
- Improved mechanical precision
- Less or no cooling
- Improved reliability
- Use of smaller motor is possible, less torque reserve required
- Less motor noise due to less energy exciting motor resonances

Encoder Interface

The TMC5240 provides an encoder interface for external incremental encoders. The encoder can be used for homing of the motion controller (alternatively to reference switches) and for consistency checks on-the-fly between encoder position and ramp generator position. A programmable prescaler allows the adaptation of the encoder resolution to the motor resolution. A 32-bit encoder counter is provided.

SPI

SPI Datagram Structure

The TMC5240 uses 40-bit SPI datagrams for communication with a microcontroller. Microcontrollers, which are equipped with hardware SPI, are typically able to communicate using integer multiples of eight bits. The CSN line of the device must stay active (= low) for the complete duration of the datagram transmission.

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

All registers are readable, most of them are read write, some read only, and some write 1 to clear (example, GSTAT registers).

Table 1. SPI Datagram Structure

MSB (TRANSMITTED **FIRST**)				**40 BIT**		**LSB (TRANSMITTED LAST)**			
39 ... 0									
write: 8 bit address read: 8 bit SPI status		read/write 32 bit data							
39 ... 32		31 ... 0							
write to RW + 7 bit address read from 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

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 always delivers data back to the controller, independent of the W bit. The data transferred back is the data read from the address, which is transmitted with the *previous* datagram, if the previous access is a read access. If the previous access is 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 controller 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 TMC5240, the MSBs delivered back contain the SPI status. The *SPI_STATUS* is a number of eight selected status bits.

Example:

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

Table 2. SPI Read/Write Example Flow

ACTION	DATA SENT TO TMC5240	DATA RECEIVED FROM TMC5240
read *XACTUAL*	0x2100000000	0xSS and unused data*
read *XACTUAL*	0x2100000000	0xSS and XACTUAL
write *VMAX* = 0x00ABCDEF	0xA700ABCDEF	0xSS and XACTUAL
write *VMAX* = 0x00123456	0xA700123456	0xSS00ABCDEF

* SS: is a placeholder for the status bits *SPI_STATUS*.

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.

Table 3. SPI_STATUS – Status Flags Transmitted with Each SPI Access in Bits 39 to 32

BIT	NAME	COMMENT
7	*status_stop_r*	*RAMP_STAT*[1] – 1: Signals stop right switch status (motion controller only)
6	*status_stop_l*	*RAMP_STAT*[0] – 1: Signals stop left switch status (motion controller only)
5	*position_reached*	*RAMP_STAT*[9] – 1: Signals target position reached (motion controller only)
4	*velocity_reached*	*RAMP_STAT*[8] – 1: Signals target velocity reached (motion controller only)
3	*standstill*	*DRV_STATUS*[31] – 1: Signals motor stand still
2	*sg2*	*DRV_STATUS*[24] – 1: Signals StallGuard flag active
1	*driver_error*	*GSTAT*[1] – 1: Signals driver driver error (clear by reading *GSTAT*)
0	*reset_flag*	*GSTAT*[0] – 1: Signals, that a reset has occurred (clear by reading *GSTAT*)

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.

SPI Signals

The SPI bus on the TMC5240 has four signals:

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

The SPI peripheral 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 peripheral 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 TMC5240.

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.

The 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 SPI controller to the SPI peripheral. If more than 40 bits are sent, only the last 40 bits received before the rising edge of CSN are recognized as the command.

SPI Timing

The SPI max frequency is at 10MHz. SCK is independent from the clock frequency of the system while the only parameter depending on the clock frequency is the minimum CSN high time. All SPI inputs are internally filtered to avoid triggering on pulses shorter than 10ns. The following figure shows the timing parameters of an SPI bus transaction. Timing values are given in the EC table.

The SPI uses SPI MODE 3.

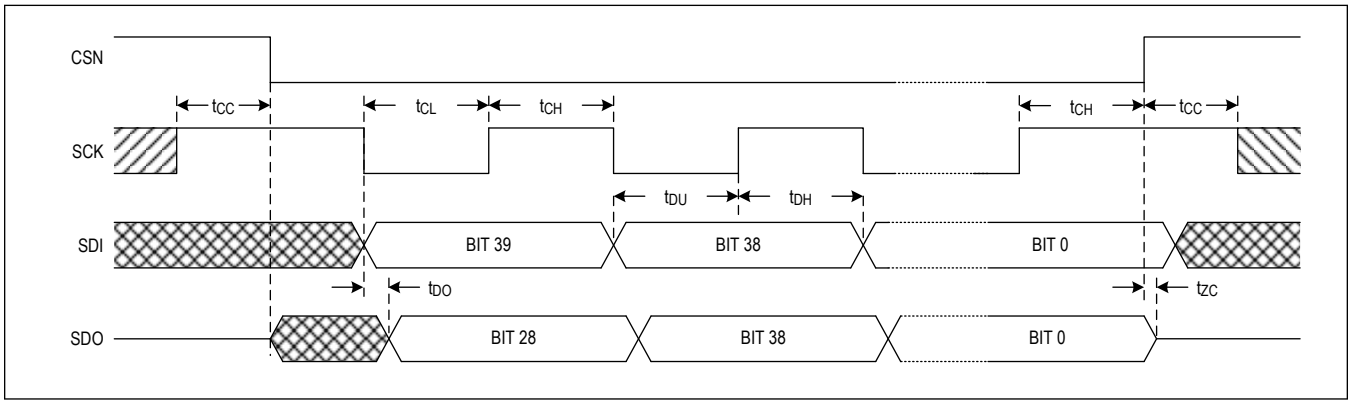


Figure 4. SPI Timing Diagram

UART Single-Wire Interface

The UART single-wire interface allows control of the TMC5240 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 (example, over cables between two PCBs) can be bridged without danger of wrong or missed commands even in the event of electromagnetic disturbance. The automatic baud rate detection makes this interface easy to use.

Datagram Structure

Write Access

Table 4. UART Write Access Datagram Structure

EACH BYTE IS LSB…MSB, HIGHEST BYTE TRANSMITTED FIRST																			
0 … 63																			
sync + reserved								8 bit node address			RW + 7 bit register addr.			32 bit data			CRC		
0…7								8…15			16…23			24…55			56…63		
1	0	1	0	Reserved (don't cares but included in CRC)				*NODEADDR*			register address		1	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	55	56	…	63

A sync nibble precedes each transmission to and from the TMC5240 and is embedded into the first transmitted byte, followed by an addressing byte. Each transmission allows a synchronization of the internal baud rate divider to the UART host 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 DIAG1/SW) and ends with a stop bit (logic 1, high level on DIAG1/SW). 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 20MHz clock (worst case for low baud rate). Maximum baud rate is $f_{CLK}/16$ due to the required stability of the baud clock.

The initial peripheral address *NODEADDR* is selected by CSN_AD2, SCK_AD1, SDI_AD0 in the range 0 to 7.

The peripheral address is determined by the sum of the register *NODEADDR* and the pin selection given above. This means that a high level on SDI (with CSN low and SCK low) increments the *NODEADDR* setting by one.

Bit 7 of the register address identifies a read (0) or a write (1) access. Example: Address 0x10 is changed to 0x90 for a write access.

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 UART host to reset communication in case of transmission errors. Any pulse on an idle data line below 16 clock cycles is 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 resynchronization of the transmission after any error conditions. Consider that due to this mechanism an abrupt reduction of the baud rate to less than 15% 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 UART host to check the success of an initialization sequence or single write accesses. Read accesses do not modify the counter.

Read Access

Table 5. UART Read Access Request Datagram Structure

EACH BYTE IS LSB…MSB, HIGHEST BYTE TRANSMITTED FIRST																	
sync + reserved								8 bit node address			RW + 7 bit register address			CRC		
0...7								8…15			16…23			24…31		
1	0	1	0	Reserved (don't cares but included in CRC)				*NODEADDR*			register address		0	CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	31

The read access request datagram structure is identical to the write access datagram structure, but uses a lower number of user bits. Its function is the addressing of the UART node and the transmission of the desired register address for the read access. The TMC5240 responds with the same baud rate as the UART host uses for the read request.

To ensure a clean bus transition from the host to the node, the TMC5240 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 UART host. In a multi-node system, set *SENDDELAY* to min. 2 for all nodes. Otherwise, a non-addressed node might detect a transmission error upon read access to a different node.

Table 6. UART Read Access Reply Datagram Structure

EACH BYTE IS LSB…MSB, HIGHEST BYTE TRANSMITTED FIRST																			
0 63																			
sync + reserved								8 bit node address			RW + 7 bit register addr.			32 bit data			CRC		
0…7								8…15			16…23			24…55			56…63		
1	0	1	0	reserved (0)				0xFF			register address		0	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	55	56	…	63

The read response is sent to the UART host using address code %11111111. The transmitter becomes switched inactive four bit times after the last bit is sent.

Address %11111111 is reserved for read accesses going to the UART host. A node cannot use this address.

CRC Calculation

An 8-bit CRC polynomial is used for checking both read and write access. It allows detection of up to eight single-bit errors. The CRC8-ATM polynomial with an initial value of zero is applied LSB to MSB, including the sync and addressing byte. The sync nibble is assumed to be always correct. The TMC5240 responds only to correctly transmitted datagrams containing its own node address. It increases its datagram counter for each correctly received write access datagram.

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

Serial calculation example

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

C-Code Example for CRC Calculation

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

  *crc = 0;

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

UART Signals

The UART interface on the TMC5240 comprises five signals. In UART mode, each node checks the single-wire pin DIAG1/SW for correctly received datagrams with its own address continuously. The pin is switched as input during this time. It adapts to the baud rate based on the sync nibble, as described earlier. In case of a read access, it switches on its output driver on DIAG1/SW and sends its response using the same baud rate.

Table 7. TMC5240 UART Interface Signals

SIGNAL	DESCRIPTION
DIAG1/SW	Data input and output
CSN/AD2	Bit 2 of UART address increment (+4)
SCK/AD1	Bit 1 of UART address increment (+2)

Table 7. TMC5240 UART Interface Signals (continued)

SDI/AD0	Bit 0 of UART address increment (+1), tie to NAO of previous IC in chain
SDO/NAO	NAO pin for chained sequential addressing scheme (reset default = high)

Addressing Multiple Nodes

If only one or up to eight TMC5240 are addressed by a host using a single UART bus interface, a simple hardware address selection can be used. The individual UART node addresses are set by connecting the UART address pins (SDI, SCK, CSN) to V_{CC_IO} and GND.

If more than eight nodes need to be connected to the same UART bus, then a different approach must be used. This approach can address up to 255 nodes by using the output NAO (SDO) as a selection pin for the bit 0 address pin of the next device. Proceed as follows:

- Tie all address pins as well as SDI/AD0 of the first TMC5240 to GND.
- Connect SDO/NAO output of the first TMC5240 to the next node's address[0] pin (SDI/AD0). Connect further nodes in the same fashion.
- Now, the first node responds to address 0. Following nodes are set to address 1.
- Program the first TMC5240 to its specific node address. **Note**: Once a node is initialized with its node address, its SDO/NAO output, which is tied to the next node's address[0] pin (SDI/AD0), has to be programmed to logic 0 to differentiate the next node from all following nodes.
- Now, the second node is accessible and can get its specific node address. Further nodes can be programmed to their specific node addresses sequentially.

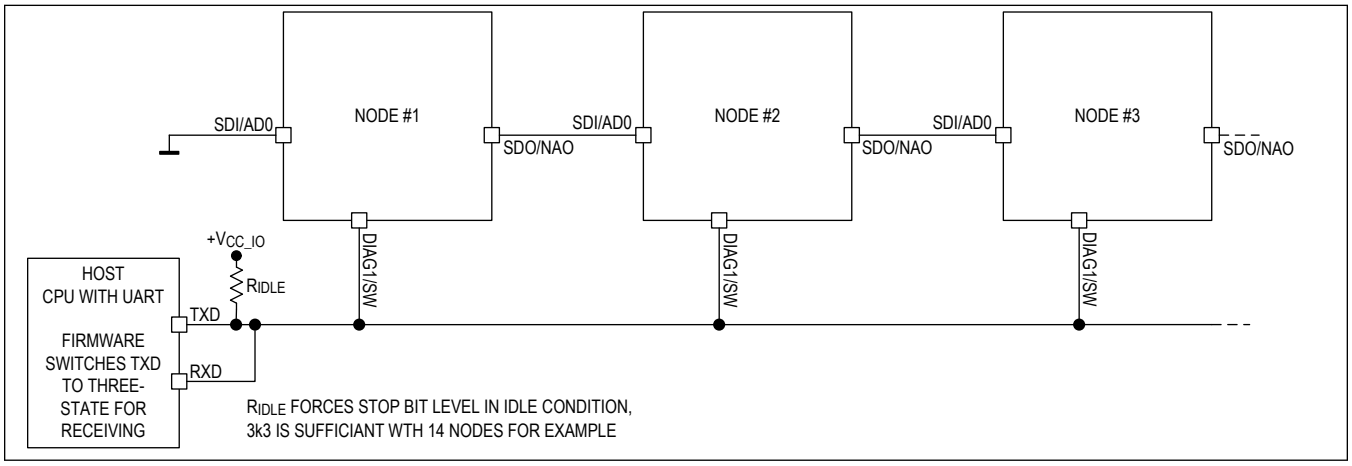


Figure 5. UART Daisy-Chaining Example

Table 8. UART Example for Addressing up to 255 Nodes

PHASE	NODE #1	NODE #2	NODE #3
Addressing phase 1	Address 0, NAO is high	Address 1	Address 1
Addressing phase 2	Program to address 254 & set NAO low	Address 0, NAO is high	Address 1
Addressing phase 3	Address 254	Program to address 253 and set NAO low	Address 0
Addressing phase 4	Address 254	Address 253	Program to address 252 and set NAO low
Addressing phase x	Continue procedure		

StealthChop2

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

Operate the motor within the application when exploring StealthChop2. Motor performance often is better with a mechanical load because it prevents the motor from stalling due to mechanical oscillations, which can occur without load.

Automatic Tuning

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

Table 9. Constraints and Requirements for StealthChop2 Autotuning AT#1 and AT#2

STEP	PARAMETER	CONDITIONS	REQUIRED DURATION
AT#1	*PWM_OFS_AUTO*	• Motor in standstill and actual current scale (*CS*) is identical to run current (*IRUN*). • If standstill reduction is enabled, an initial step pulse switches the drive back to run current, or set *IHOLD* to *IRUN*. • Pin V_S at operating level.	≤ 2^{20} + 2 x 2^{18} t_{CLK}, ≤ 130ms (with internal clock)
AT#2	*PWM_GRAD_AUTO*	• Move motor at a velocity, where a significant amount of back EMF is generated and where the full run current can be reached. Conditions: • 1.5 x *PWM_OFS_AUTO* x (*IRUN*+1)/32 < *PWM_SCALE_SUM* < 4 x *PWM_OFS_AUTO x* (*IRUN*+1)/32 • *PWM_SCALE_SUM* < 255 **Hint**: A typical range is 60RPM to 300RPM.	8 fullsteps are required for a change of ±1. For a typical motor with *PWM_GRAD_AUTO* optimum at 50 or less, up to 400 fullsteps are required when starting from default value 0.

Hint:

Determine best conditions for automatic tuning with the evaluation board.

Use application-specific parameters for *PWM_GRAD* and *PWM_OFS* for initialization in firmware to provide initial tuning parameters.

Monitor *PWM_SCALE_AUTO* going down to zero during the constant velocity phase in AT#2 tuning. This indicates a successful tuning.

Attention:

Operating in StealthChop2 without proper tuning can lead to high motor currents during a deceleration ramp, especially with low resistive motors and fast deceleration settings. Follow the automatic tuning process and check optimum tuning conditions using the evaluation board. It is recommended to use an initial value for settings *PWM_OFS* and *PWM_GRAD* determined per motor type.

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

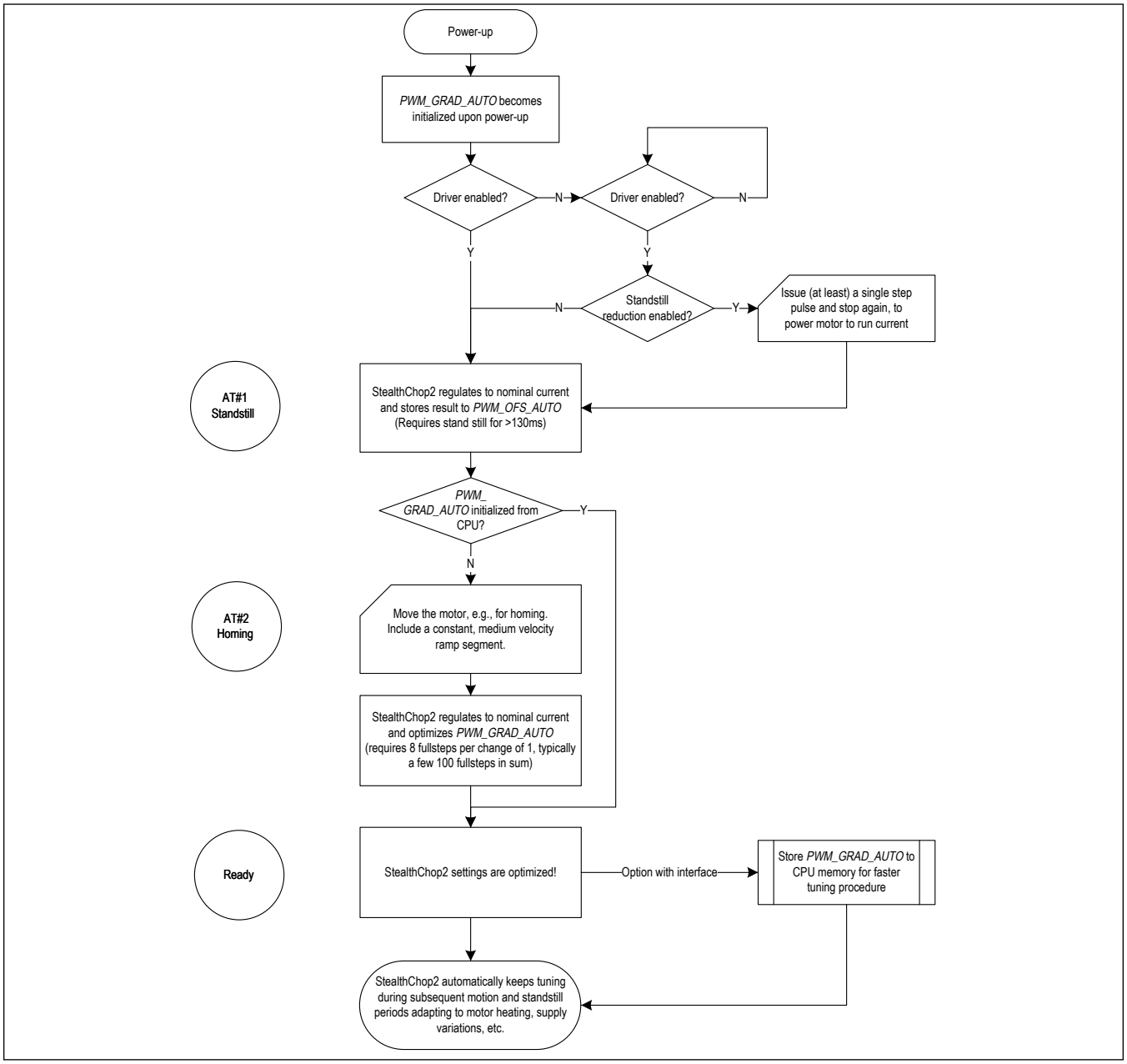


Figure 6. StealthChop2 Automatic Tuning Procedure

StealthChop2 Options

To match the motor current to a certain level, the effective PWM voltage becomes scaled depending on the actual motor velocity. Several additional factors influence the required voltage level to drive the motor at the target current: the motor resistance, its back EMF (example, directly proportional to its velocity), as well as the actual level of the supply voltage. Two modes of PWM regulation are provided: the automatic tuning mode (AT) using current feedback (*pwm_autoscale* = 1, *pwm_autograd* = 1) and a feed forward velocity-controlled mode (*pwm_autoscale* = 0). The feed forward velocity-controlled mode does not react to a change of the supply voltage or to events like a motor stall, but it provides very stable amplitude. It does not use or require any means of current measurement. This is perfect when motor type and supply

voltage are well known. Therefore, the automatic mode is recommended, unless current regulation is not satisfying in the given operating conditions.

It is recommended to use application-specific initial tuning parameters, fitting the motor type and supply voltage. Additionally, operate in automatic tuning mode to respond to parameter change, example, due to motor heat-up or change of supply voltage.

Non-automatic mode (*pwm_autoscale = 0*) should be taken into account only with well-known motor and operating conditions. In this case, careful programming through the interface is required. The operating parameters *PWM_GRAD* and *PWM_OFS* can be determined in automatic tuning mode initially.

The StealthChop2 PWM frequency can be chosen in four steps to adapt the frequency divider to the frequency of the clock source. A setting in the range of 20kHz to 50kHz is good for most applications. It balances low current ripple and good higher velocity performance vs. dynamic power dissipation.

Table 10. Choice of PWM Frequency for StealthChop2 (Bold Font = Recommended)

CLOCK FREQUENCY f_{CLK}	**PWM_FREQ = %00 f_{PWM} = 2/1024 f_{CLK}**	**PWM_FREQ = %01 f_{PWM} = 2/683 f_{CLK}**	**PWM_FREQ = %10 f_{PWM} = 2/512 f_{CLK}**	**PWM_FREQ = %11 f_{PWM} = 2/410 f_{CLK}**
20MHz	**39.1kHz**	58.1kHz	78.1kHz	97.6kHz
18MHz	**35.2kHz**	52.7kHz	70.3kHz	87.8kHz
16MHz	**31.3kHz**	**46.9kHz**	62.5kHz	78.0kHz
12.5MHz (internal)	**24.4kHz**	**36.6kHz**	**48.8kHz**	61.0kHz
10MHz	19.5kHz	**29.3kHz**	**39.1kHz**	**48.8kHz**
8MHz	15.6kHz	**23.4kHz**	**31.2kHz**	**39.0kHz**

StealthChop2 Current Regulator

In StealthChop2 voltage PWM mode, the autoscaling function (*pwm_autoscale* = 1, *pwm_auto_grad* = 1) regulates the motor current to the desired current setting. Automatic scaling is used as part of the AT process, and for subsequent tracking of changes within the motor parameters. The driver measures the motor current during the chopper on time and uses a proportional regulator to regulate *PWM_SCALE_AUTO* to match the motor current to the target current. *PWM_REG* is the proportionality coefficient for this regulator. Basically, the proportionality coefficient should be as small as possible to get a stable and soft regulation behavior, but it must be large enough to allow the driver to quickly react to changes caused by variation of the motor target current (example, change of V_{REF}). During initial tuning step, AT#2, *PWM_REG* also compensates for the change of motor velocity. Therefore, a high acceleration during AT#2 requires a higher setting of *PWM_REG*. With careful selection of homing velocity and acceleration, a minimum setting of the regulation gradient often is sufficient (*PWM_REG* = 1). *PWM_REG* setting should be optimized for the fastest required acceleration and deceleration ramp (compare the following two figures).

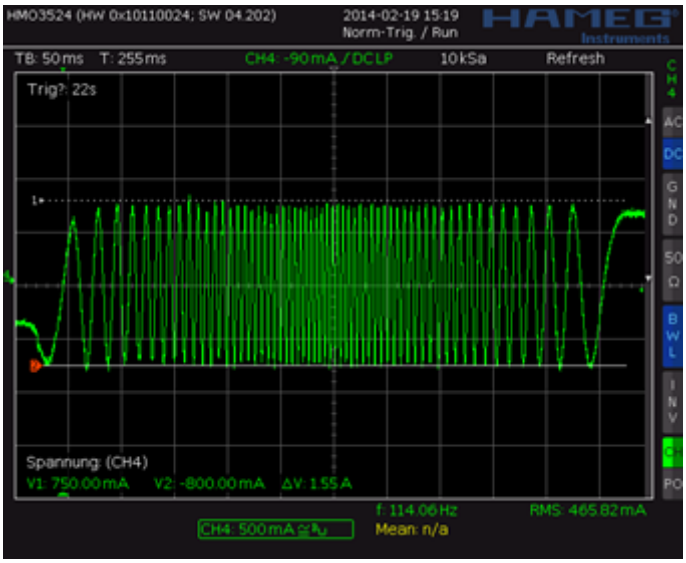


Figure 7. StealthChop2: Good Setting for PWM_REG

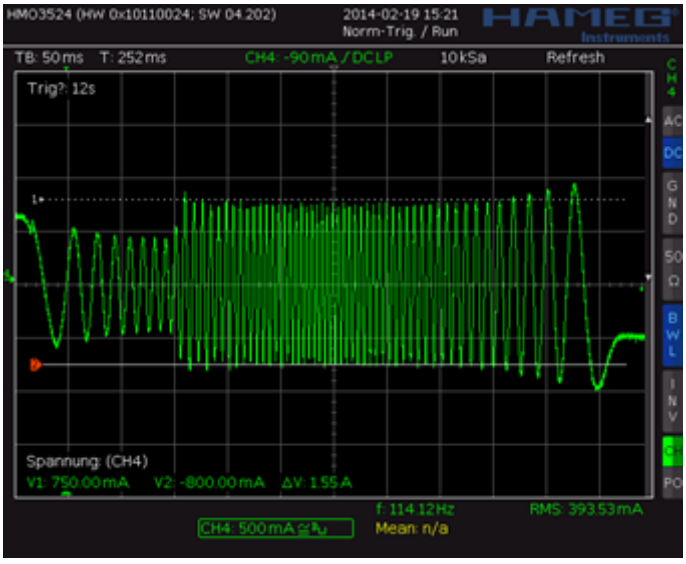


Figure 8. StealthChop2: Too Small Setting for PWM_REG during AT#2

The quality of the setting *PWM_REG* in phase AT#2 and the finished automatic tuning procedure (or non-automatic settings for *PWM_OFS* and *PWM_GRAD*) can be examined when monitoring motor current during an acceleration phase, as shown in the next figure.

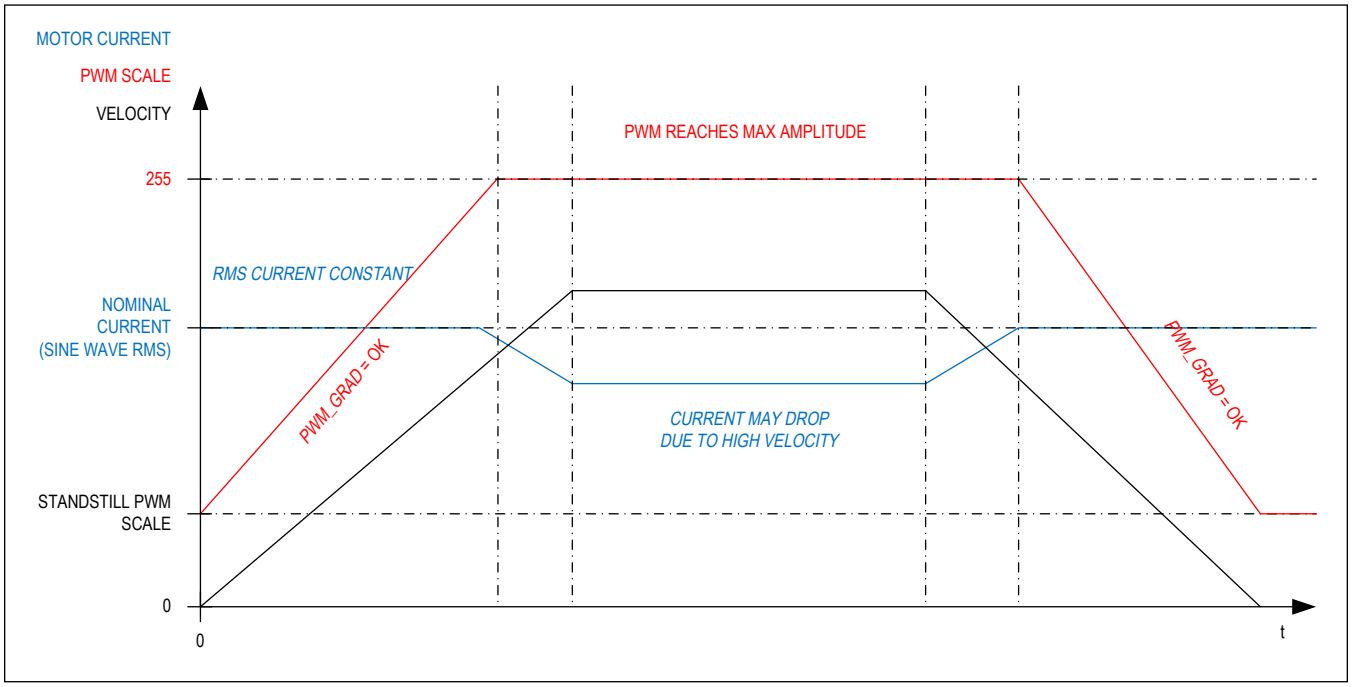


Figure 9. Successfully Determined PWM_GRAD(_AUTO) and PWM_OFS(_AUTO)

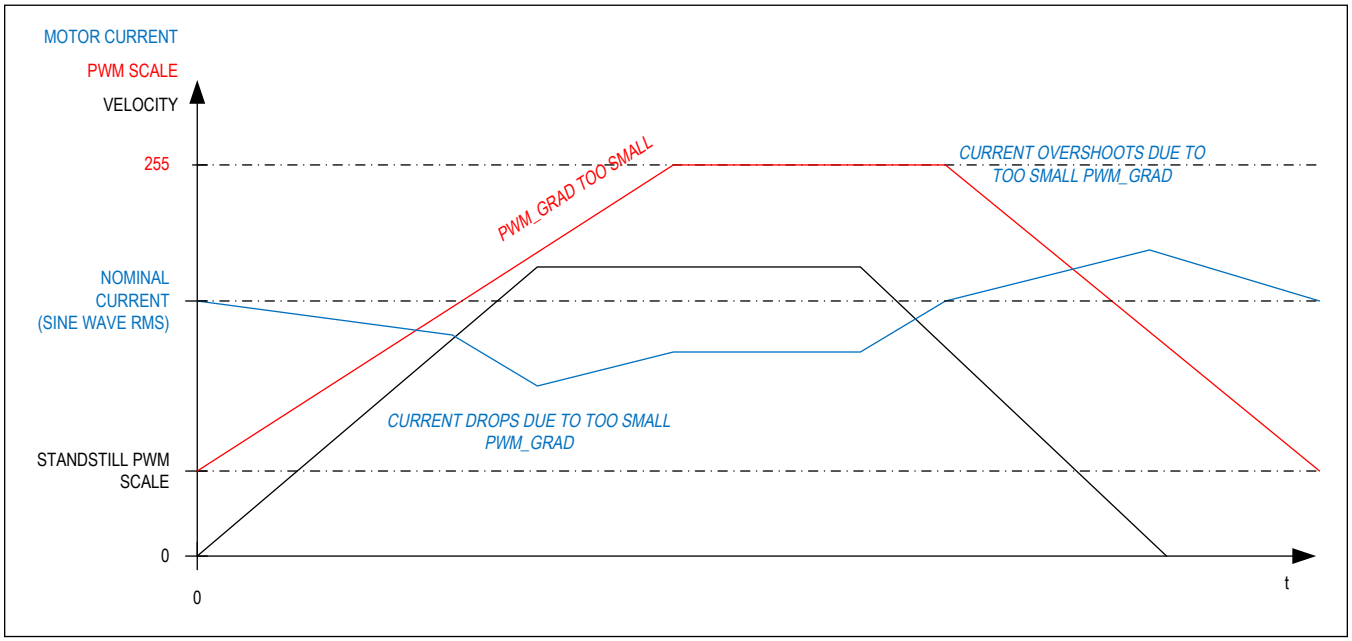


Figure 10. Example for Too Small PWM_GRAD Setting

Lower Current Limit

Depending on the setting of *pwm_meas_sd_enable,* the StealthChop2 current regulator principle imposes a lower limit

for motor current regulation. As the coil current is measured during chopper on phase only (*pwm_meas_sd_enable* = 0), a minimum chopper duty cycle allowing coil current regulation is given by the blank time as set by *TBL* and by the chopper frequency setting. Therefore, the motor-specific minimum coil current in StealthChop2 autoscaling mode rises with the supply voltage and with the chopper frequency. A lower blanking time allows a lower current limit. It is important for the correct determination of *PWM_OFS_AUTO*, that in AT#1 the run current, *GLOBALSCALER,* and *IRUN* is well within the regulation range. Lower currents (example, for standstill power down) are automatically realized based on *PWM_OFS_AUTO* and *PWM_GRAD_AUTO,* respectively, based on *PWM_OFS* and *PWM_GRAD* with non-automatic current scaling. The freewheeling option allows going to zero motor current.

Lower motor coil current limit for StealthChop2 automatic tuning (*pwm_meas_sd_enable = 0)* :

$$I_{\mathrm{LowerLimit}} = t_{\mathrm{BLANK}} \times f_{\mathrm{PWM}} \times \frac{V_S}{R_{\mathrm{COIL}}}$$

V_S being the motor supply voltage and R_{COIL} the motor coil resistance.

$I_{LowerLimit}$ can be treated as a rule-of-thumb value for the minimum nominal *IRUN* motor current setting. I the lower limit is not sufficient to reach the desired setting be sure to set *pwm_meas_sd_enable = 1*.

f_{PWM} is the chopper frequency as determined by setting *PWM_FREQ*.

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

$$I_{\mathrm{LowerLimit}} = 24 t_{\mathrm{CLK}} \times \frac{2}{1024 t_{\mathrm{CLK}}} \times \frac{24V}{5\Omega} = \frac{24}{512} \times \frac{24V}{5\Omega} = 225\mathrm{mA}$$

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

Attention*:*

For automatic tuning, a lower coil current limit applies.

IRUN ≥ 8: Current settings for *IRUN* below 8 do not work with automatic tuning.

$I_{LOWERLIMIT}$: Depending on the setting of bit *pwm_meas_sd_enable (in register PWM_CONF[22])* for automatic tuning, a lower coil current limit applies.The motor current in automatic tuning phase AT#1 must exceed this lower limit. Calculate $I_{LOWERLIMIT}$ or measure it using a current probe. Setting the motor run-current or hold-current below the lower current limit during operation by modifying *IRUN* and *IHOLD* is possible after successful automatic tuning. The lower current limit also limits the capability of the driver to respond to changes of *GLOBALSCALER*.

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

To overcome the lower limit, set *pwm_meas_sd_enable* = 1. This allows the IC to additionally measure coil current in the slow decay phase.

Velocity-Based Scaling

Velocity-based scaling scales the StealthChop2 amplitude based on the time between every two steps, example, based on *TSTEP*, measured in clock cycles. This concept basically does not require a current measurement, because no regulation loop is necessary. A pure velocity-based scaling is available through programming, only when setting *pwm_autoscale* = 0. The basic idea is to have a linear approximation of the voltage required to drive the target current into the motor. The stepper motor has a certain coil resistance and thus needs a certain voltage amplitude to yield a target current based on the basic formula I = U/R. R being the coil resistance, U the supply voltage is scaled by the PWM value, and the current I results. The initial value for *PWM_OFS* can be calculated:

$$\mathrm{PWM\_OFS} = \frac{374 \times R_{\mathrm{COIL}} \times I_{\mathrm{COIL}}}{V_S}$$

V_S being the motor supply voltage and I_{COIL} the target RMS current.

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

$$U_{\mathrm{PWM}} = V_S \times \frac{\mathrm{PWM\_SCALE}}{256} \times \frac{248}{256} \times \frac{1}{\sqrt{2}} = V_S \times \frac{\mathrm{PWM\_SCALE}}{374}$$

With rising motor velocity, the motor generates an increasing back EMF voltage. The back EMF voltage is proportional to the motor velocity. It reduces the PWM voltage effective at the coil resistance and thus current decreases. The TMC5240 provides a second velocity dependent factor (*PWM_GRAD*) to compensate for this. The overall effective PWM amplitude (*PWM_SCALE_SUM*) in this mode automatically is calculated in dependence of the microstep frequency as:

$$\text{PWM\_SCALE\_SUM} = \text{PWM\_OFS} x\left(\frac{\text{CS\_ACTUAL} + 1}{32}\right) + \text{PWM\_GRAD} \times \frac{256}{\text{TSTEP}}$$

CS_ACTUAL takes into account the actual current scaling as defined by IHOLD and IRUN or respectively by CoolStep.

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

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

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

C_{BEMF} is the back EMF constant of the motor in Volts per radian/second.

MSPR is the number of microsteps per rotation related to 1/256 microstep resolution, example, 51200 = 256 microsteps multiplied by 200 fullsteps for a 1.8° motor.

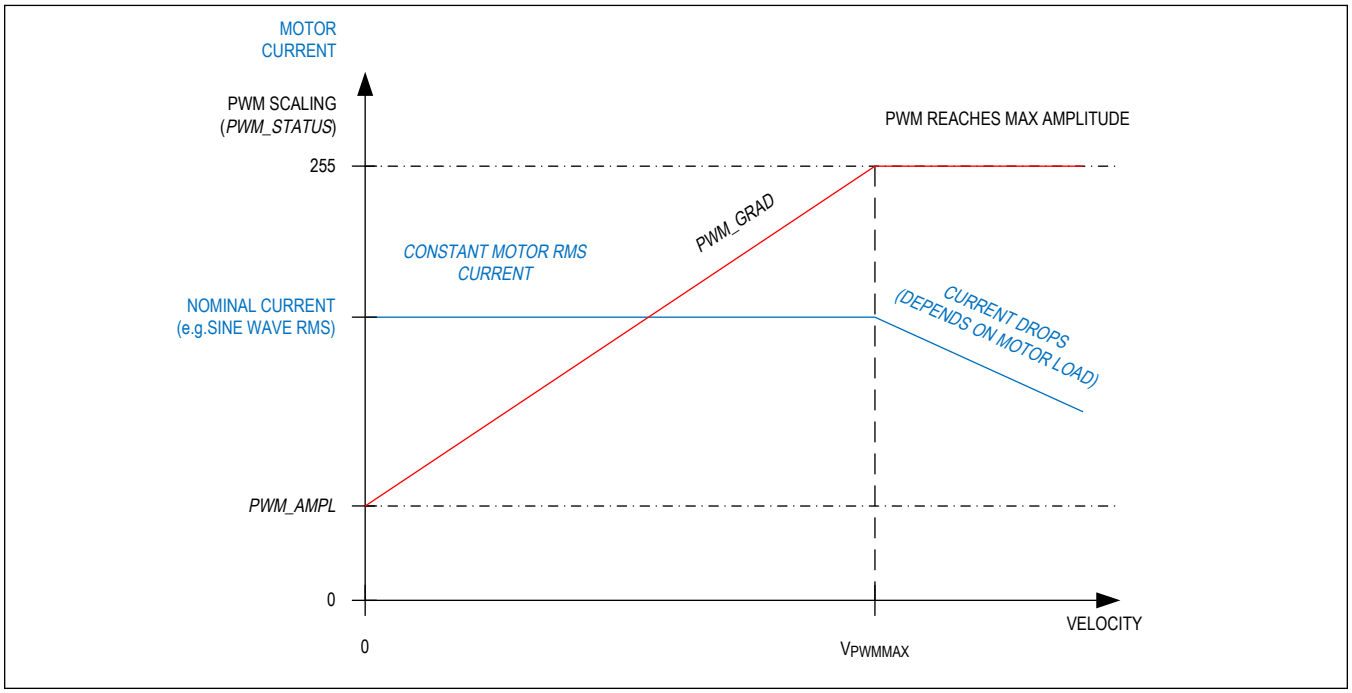


Figure 11. Velocity-Based PWM Scaling (pwm_autoscale = 0)

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

Understanding the back EMF constant of a motor:

The back EMF constant is the voltage a motor generates when turned with a certain velocity. Often motor data sheets do not specify this value, as it can be deducted from motor torque and coil current rating. Within SI units, the numeric value

of the back EMF constant C_{BEMF} has the same numeric value as the numeric value of the torque constant. For example, a motor with a torque constant of 1 Nm/A has a C_{BEMF} of 1V/rad/s. Turning such a motor with 1rps (1rps = 1 revolution per second = 6.28 rad/s) generates a back EMF voltage of 6.28V. Thus, the back EMF constant can be calculated as:

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

$I_{COILNOM}$ is the motor's rated RMS phase current for the specified holding torque.

HoldingTorque is the motor specific holding torque, example, the torque reached at $I_{COILNOM}$ on both coils. The torque unit is [Nm], where 1Nm = 100Ncm = 1000mNm.

The voltage is valid as RMS voltage per coil. Thus, the nominal current is multiplied by 2 in this formula, as the nominal current assumes a fullstep position, with two coils operating.

Combining StealthChop2 and SpreadCycle

For applications requiring high velocity motion, SpreadCycle may bring more stable operation in the upper velocity range. To combine no-noise operation with highest dynamic performance, the TMC5240 allows combining StealthChop2 and SpreadCycle based on a velocity threshold. With this, StealthChop2 is only active at low velocities.

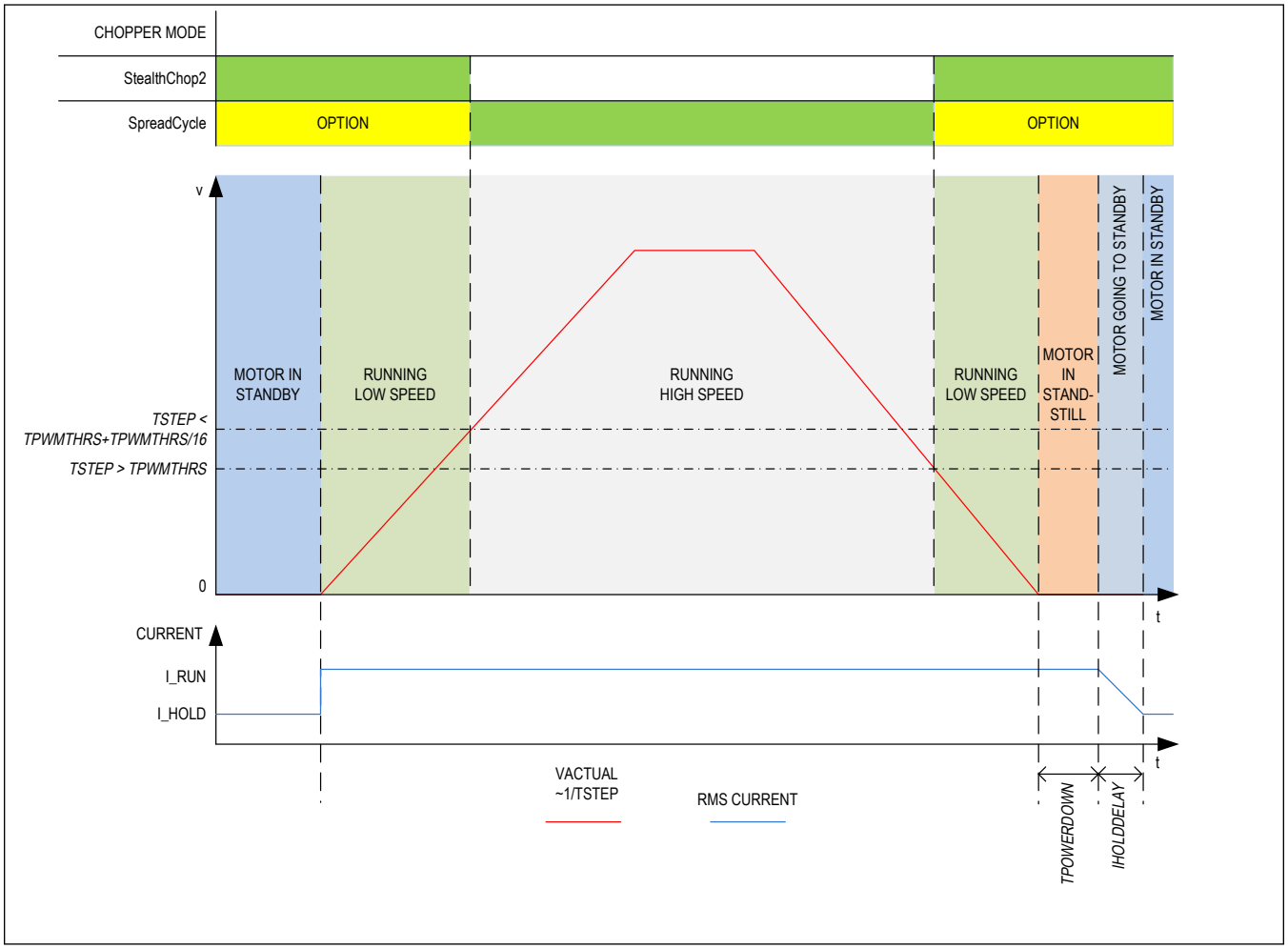


Figure 12. TPWMTHRS for Optional Switching to SpreadCycle

As a first step, both chopper principles should be parameterized and optimized individually.

In a next step, the switchover velocity has to be defined. For example, StealthChop2 operation is used for precise low speed positioning, while SpreadCycle shall be used for highly dynamic motion. *TPWMTHRS* determines this transition velocity. Read out *TSTEP* when moving at the desired velocity and program the resulting value to *TPWMTHRS*. Use a low transfer velocity to avoid a jerk at the switching point.

Jerkless Switching to SpreadCycle:

A jerk occurs when switching at higher velocities, because the back-EMF of the motor (which rises with the velocity) causes a phase shift of up to 90° between motor voltage and motor current. So, when switching at higher velocities between voltage PWM and current PWM mode, this jerk occurs with increased intensity. A high jerk may even produce a temporary overcurrent condition (depending on the motor coil resistance). At low velocities (example, 1RPM to a few 10RPM), it can be completely neglected for most motors. Therefore, consider the jerk when switching the driver between SpreadCycle and StealthChop2. With automatic switching controlled by *TPWMTHRS*, the driver can automatically eliminate the jerk by using StallGuard4 to determine the phase shift. It applies the same phase shift to SpreadCycle until the velocity falls back below the switching threshold. Set flag *SG4_THRS*.sg_angle_offset to enable this function.

Set *TPWMTHRS* zero to work with StealthChop2 only.

When enabling the StealthChop2 mode the first time using automatic current regulation, the motor must be at standstill to allow a proper current regulation. When the drive switches to SpreadCycle at a higher velocity, StealthChop2 logic stores the last current regulation setting until the motor returns to a lower velocity again. This way, the regulation has a known starting point when returning to a lower velocity, where StealthChop2 becomes re-enabled. Therefore, neither the velocity threshold nor the supply voltage must be considerably changed during the phase while the chopper is switched to a different mode because otherwise, the motor might lose steps or the instantaneous current might be too high or too low.

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

Start the motor from standstill when switching on StealthChop2 the first time and keep it stopped for at least 128 chopper periods to allow StealthChop2 to do initial standstill current control.

Flags in StealthChop2

As StealthChop2 uses voltage mode driving, status flags based on current measurement respond slower, respectively, the driver reacts delayed to sudden changes of back EMF, like on a motor stall.

A motor stall, or abrupt stop of the motion during operation in StealthChop2 can lead to an overcurrent condition. Depending on the previous motor velocity, and on the coil resistance of the motor, it significantly increases motor current for a time of several 10ms. With low velocities, where the back EMF is just a fraction of the supply voltage, there is no danger of triggering the short detection.

Switch the driver stage to the lowest current range (DRV_CONF.current_range) supporting the motor. This automatically adapts the overcurrent threshold in three steps and thus reduces peak currents in case of a sudden motor stall.

Open Load Flags

In StealthChop2 mode, the status information is different compared to the cycle-by-cycle regulated SpreadCycle mode for the flags OLA and OLB.

- If OLA and OLB are not set, this indicates that the current regulation is reaching the nominal current on both coils.
- If constant OLA and OLB flags, this indicates an interrupted motor coil.
- If flickering OLA and OLB, this indicates differences in motor coil resistance exceeding roughly 5%.
- One or both flags are active, if the current regulation did not succeed in scaling up to the full target current within the last few fullsteps (because no motor is attached or a high velocity exceeds the PWM limit).

When there is an open-load situation on one coil, the current regulation can exceed the target current on the other coil up to the overcurrent detection trip point. This is because the current regulation in certain situations, due to measurement restriction, only regulates the current on the coil with higher target current. In critical applications, check for open load in SpreadCycle first.

If desired, do an on-demand open load test using the SpreadCycle chopper as it delivers the safest result. With StealthChop2, *PWM_SCALE_SUM* can be checked to detect the correct coil resistance.

PWM_SCALE_SUM Informs about the Motor State

Information about the motor state is available with automatic scaling by reading out *PWM_SCALE_SUM*. As this parameter reflects the actual voltage required to drive the target current into the motor, it depends on several factors: motor load, coil resistance, supply voltage, and current setting. Therefore, an evaluation of the *PWM_SCALE_SUM* value allows checking the motor operation point. When reaching the limit (1023), the current regulator cannot sustain the full motor current, example, due to a permanent or temporary drop in supply voltage.

Freewheeling and Passive Braking

StealthChop2 provides different options for motor standstill. These options can be enabled by setting the standstill current *IHOLD* to zero and choosing the desired option using the *FREEWHEEL* setting. The desired option becomes enabled after a time period specified by *TPOWERDOWN* and *IHOLDDELAY*. Current regulation becomes frozen once the motor target current is at zero current to ensure a quick start-up. With the freewheeling options, both freewheeling and passive braking can be realized. Passive braking is an effective eddy current motor braking, which consumes a minimum amount of energy because no active current is driven into the coils. However, passive braking allows slow turning of the motor

when a continuous torque is applied.

Parameters Controlling StealthChop2

The following table contains all parameters related to the StealthChop2 chopper mode.

Table 11. Parameters Controlling StealthChop2

PARAMETER	DESCRIPTION	SETTING	COMMENT
en_pwm_ mode	General enable for use of StealthChop2 (register GCONF). Default = 0	0	StealthChop2 disabled. SpreadCycle active.
		1	StealthChop2 enabled (depending on velocity thresholds). Enable only while in stand-still and at IHOLD= nominal IRUN current.
pwm_meas_sd_enable	Control of current measurement during slow decay phase. Default = 0	0	Current measured during on-phases only. Lower current limit applies.
		1	Current measured during slow decay phases additionally to overcome lower current limit.
pwm_dis_reg_stst	This option eliminates any regulation noise during standstill. Default = 0	0	Current regulation always on.
		1	Disable current regulation when motor is in standstill and current is reduced (less than IRUN).
TPWMTHRS	Specifies the upper velocity for operation in StealthChop2. Enter the TSTEP reading (time between two microsteps) when operating at the desired threshold velocity. Default = 0	0 … 1048575	StealthChop2 is disabled if TSTEP falls under TPWMTHRS
PWM_LIM	Limiting value for limiting the current jerk when switching from SpreadCycle to StealthChop2. Reduce the value to yield a lower current jerk. Default = 12	0 … 15	Upper four bits of 8 bit amplitude limit
pwm_ autoscale	Enable automatic current scaling using current measurement. If off, use forward controlled velocity-based mode. Default = 1	0	Forward controlled mode
		1	Automatic scaling with current regulator
pwm_ autograd	Enable automatic tuning of PWM_GRAD_AUTO Default = 1	0	Disable, use PWM_GRAD from register instead
		1	Enable
PWM_FREQ	PWM frequency selection. Use the lowest setting giving good results. The frequency measured at each of the chopper outputs is half of the effective chopper frequency f_{PWM}. Default = 0	0	$f_{PWM} = 2/1024\ f_{CLK}$
		1	$f_{PWM} = 2/683\ f_{CLK}$
		2	$f_{PWM} = 2/512\ f_{CLK}$
		3	$f_{PWM} = 2/410\ f_{CLK}$

Table 11. Parameters Controlling StealthChop2 (continued)

PWM_REG	User defined PWM amplitude regulation loop P-coefficient. A higher value leads to a higher adaptation speed when pwm_autoscale = 1. Default = 4	1 … 15	Results in 0.5 to 7.5 steps for PWM_SCALE_AUTO regulator per fullstep
PWM_OFS	User defined PWM amplitude (offset) for velocity-based scaling and initialization value for automatic tuning of PWM_OFFS_AUTO. Default = 0x1D	0 … 255	PWM_OFS = 0 disables linear current scaling based on current setting
PWM_GRAD	User defined PWM amplitude (gradient) for velocity-based scaling and initialization value for automatic tuning of PWM_GRAD_AUTO. Default = 0	0 … 255	
PWM_SCALE_SUM	Actual PWM scaling as determined by the actual settings. This value is shown in higher precision (10 Bit) compared to 8 bit for PWM_GRAD/OFS_AUTO values. Default = 0	0 ... 1023	
FREEWHEEL	Standstill option when motor current setting is zero (I_HOLD = 0). Only available with StealthChop2 enabled. The freewheeling option makes the motor easily movable, while both coil short options realize a passive brake. Default = 0	0	Normal operation
		1	Freewheeling
		2	Coil short via LS drivers
		3	Coil short via HS drivers
PWM_SCALE _AUTO	Read back of the actual StealthChop2 voltage PWM scaling correction as determined by the current regulator. Shall regulate close to 0 during tuning. Default = 0	-255 … 255	(Read-only) Scaling value becomes frozen when operating in SpreadCycle
PWM_GRAD _AUTO PWM_OFS _AUTO	Allow monitoring of the automatic tuning and determination of initial values for PWM_OFS and PWM_GRAD. Default = 0	0 … 255	(Read-only)
TOFF	General enable for the motor driver, the actual value does not influence StealthChop2 Default = 0	0	Driver off
		1 … 15	Driver enabled
TBL	Comparator blank time. Choose a setting of 1 or 2 for typical applications. For higher capacitive loads, 3 may be required. Lower settings allow StealthChop2 to regulate down to lower coil current values. Default = 2	0	16 t_{CLK}
		1	24 t_{CLK}
		2	36 t_{CLK}
		3	54 t_{CLK}

SpreadCycle and Classic Chopper

While StealthChop2 is a voltage mode PWM controlled chopper, SpreadCycle is a cycle-by-cycle current control. Therefore, it can react extremely fast to changes in motor velocity or motor load. The currents through both motor coils are controlled using choppers. The choppers work independently of each other. In the following figure, the different chopper phases are shown.

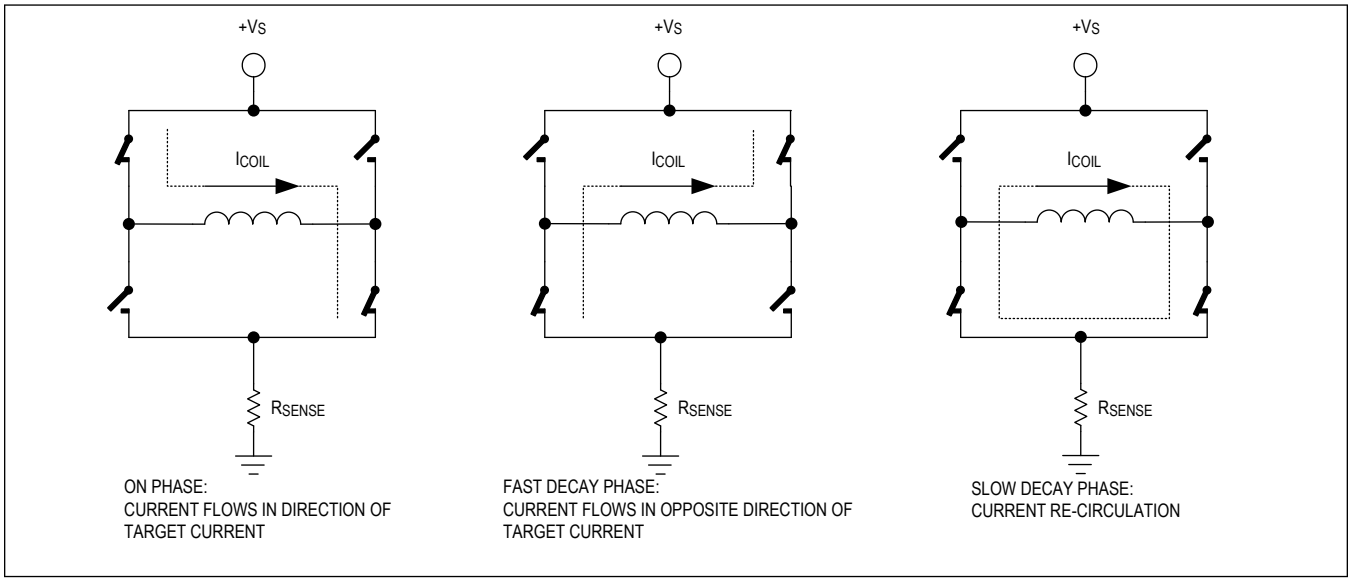


Figure 13. Typical Chopper Decay Phases

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 measures coil current during phases when the current flows through exactly one lowside transistor, but not during the slow decay phase. 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 in the $R_{DS(ON)}$-based current measurement occur due to charging and discharging parasitic capacitance. During this time, typically one or two microseconds, the current cannot be measured. Blanking is the time when the input to the comparator is masked to block these spikes.

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

The chopper frequency is an important parameter for a chopped motor driver. A too low frequency might generate audible noise. A higher frequency reduces current ripple in the motor, but with a too high frequency magnetic losses may rise. Also power dissipation in the driver rises with increasing frequency due to the increased influence of switching slopes causing dynamic dissipation. Therefore, a compromise needs to be found. Most motors are optimally working in a frequency range of 25kHz to 40kHz. 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 25kHz to 40kHz gives a good result for most motors when using SpreadCycle. A higher frequency leads to increased switching losses.

Table 12. Parameters Controlling SpreadCycle and Classic Constant Off Time Chopper

PARAMETER	DESCRIPTION	SETTING	COMMENT

Table 12. Parameters Controlling SpreadCycle and Classic Constant Off Time Chopper (continued)

TOFF	Sets the slow decay time (*off time*). This setting also limits the maximum chopper frequency. For operation with StealthChop2, this parameter is not used, but it is required to enable the motor. In case of operation with StealthChop2 only, any setting is OK. Setting this parameter to zero completely disables all driver transistors and the motor can free-wheel. Default = 0	0	Chopper off
		1…15	Off time setting N_{CLK} = 24 + 32 x *TOFF* (1 works 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. For most applications, a setting of 1 or 2 is good. For highly capacitive loads, example, when filter networks are used, a setting of 2 or 3 is required. Default = 2	0	16 t_{CLK} Restriction: Use this setting only in combination with external clock oscillator <=8MHz
		1	24 t_{CLK} Restriction: May be used with internal clock, or if external clock frequency <=13MHz is applied.
		2	36 t_{CLK}
		3	54 t_{CLK}
chm	Selection of the *chopper mode* Default = 0	0	SpreadCycle
		1	Classic const. off time

SpreadCycle Chopper

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

Each chopper cycle comprises an on phase, a slow decay phase, a fast decay phase, and a second slow decay phase. 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% to 70% of the chopper cycle in standstill and are important for low motor and driver power dissipation.

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

- Target Chopper frequency: 25kHz
  - t_{OFF} = 1 / 25kHz × 50 / 100 × 1 / 2 = 10 µ *s*
  - Assumption: Two slow decay cycles make up for 50% of overall chopper cycle time.
- For the *TOFF* setting this means: TOFF = $\left(t_{OFF} \times f_{CLK} - 12\right) / 32$
- With 12MHz clock this results in TOFF = 3.4, which requires a setting of TOFF = 3 or 4,
- With 16MHz clock this results in TOFF = 4.6, which requires a setting of TOFF = 4 or 5.

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

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 to give best microstepping results. This allows the chopper to precisely regulate the current for both rising and 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 leads 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 (example, *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 with a current probe. Checking the sine wave shape near the zero transition shows a small ledge between both half waves in case the hysteresis setting is too small. At medium velocities (example,100 fullsteps to 400 fullsteps per second), a too low hysteresis setting leads to increased humming and vibration of the motor. A too high hysteresis setting leads to reduced chopper frequency and increased chopper noise but does not yield any benefit for the wave shape.

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 results in reduced microstep accuracy, while a too high setting leads to more chopper noise and motor power dissipation. When the fast decay time becomes slightly longer than the blanking time, the setting is optimum. 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, example, 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 from reaching the audible range.

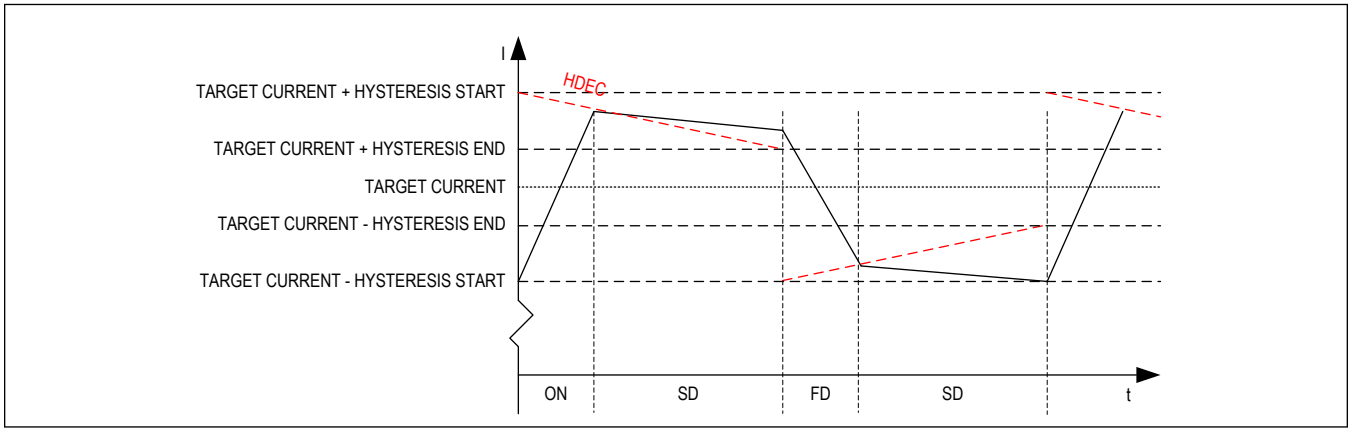


Figure 14. SpreadCycle Chopper Scheme Showing Coil Current During a Chopper Cycle

Table 13. SpreadCycle Mode Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
HSTRT	*Hysteresis start* setting. This value is an offset from the hysteresis end value *HEND*. Default = 5	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. Default = 2	0…2	-3…-1: negative HEND
		3	0: zero HEND

Table 13. SpreadCycle Mode Parameters (continued)

		4…15	1…12: positive HEND

Even at HSTRT = 0 and HEND = 0, the TMC5240 sets a minimum hysteresis through analog circuitry.

Example:

A hysteresis of 4 is chosen. There is the option 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, example, 1: 3 + 1 = 4)

To take advantage of the variable hysteresis, set most of the value to the HSTRT, example, 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)

Classic Constant Off Time Chopper

The classic constant off time chopper is an alternative to SpreadCycle. The 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.

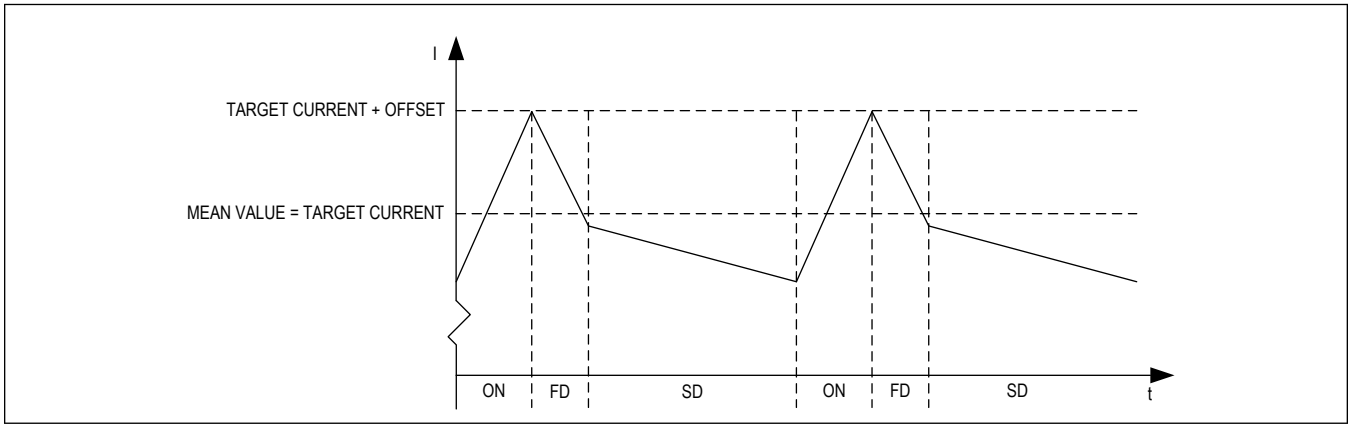


Figure 15. Classic Constant Off-Time Chopper with Offset Showing Coil Current

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 following figures). 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.

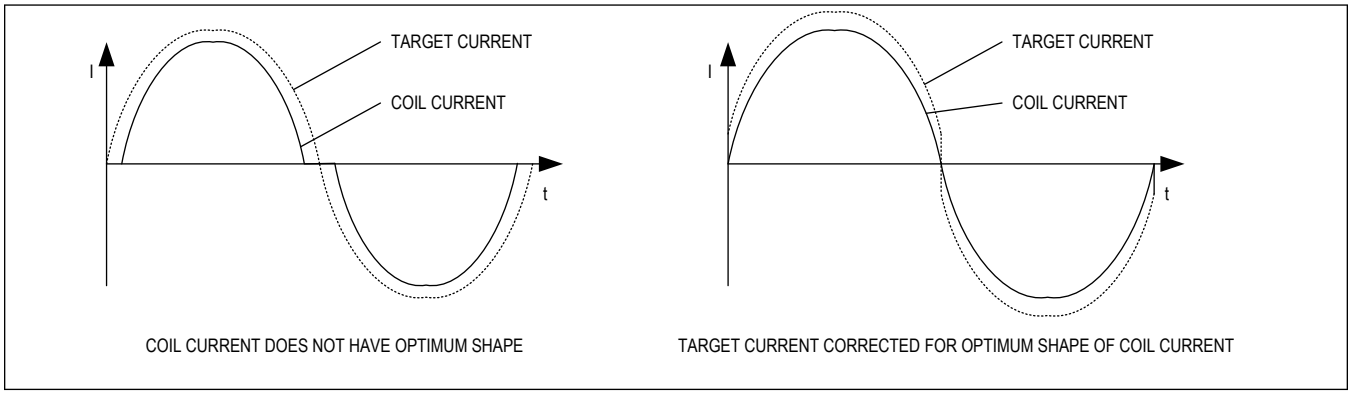


Figure 16. Zero Crossing with Classic Chopper and Correction Using Sine Wave Offset

Table 14. Parameters Controlling Constant Off-Time Chopper 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. Default = 5	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. Default = 2	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. Default = 0	0	Enable comparator termination of fast decay cycle
		1	End by time only

Integrated Current Sense

Non-dissipative current sensing is integrated in the TMC5240 (ICS). This feature eliminates the bulky external power resistors, which are normally required with external current sensing. The ICS results in a dramatic space and power saving compared with mainstream applications based on the external sense resistor. For optimum performance, the ICS individually measures $R_{DS(ON)}$ for each of the power MOSFETs taking into account individual MOSFET temperature to yield the best results.

Setting the Motor Current

The TMC5240 allows to set the motor phase current. The parameters given in the following table allow to adapt the current scaling as well as the current ramp up and ramp down.

Table 15. Parameters Controlling the Motor Current

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. Default = 31	0…31	Scaling factor 1/32, 2/32, … 32/32
IHOLD	Identical to IRUN, but for motor in standstill. Lower values <16 are OK with IHOLD in comparison to IRUN. Default = 8		
IHOLDDELAY	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 x 15 x 2^{18} clock cycles, example, roughly one second at 16MHz. Default = 1	0	Instant power down to IHOLD
		1…15	1 x 2^{18} … 15 x 2^{18} clocks per current decrement
IRUNDELAY	Controls the number of clock cycles for motor power up after start is detected. Allows smooth current increment upon start of a motion from hold current (IHOLD) to run current (IRUN). While a quick power-up is important to establish full motor torque, a small delay time helps to reduce acoustic noise and avoids a jump on the power supply current. Default = 4	0	instant power up to IRUN
		1...15	Delay per current increment step in multiple of IRUNDELAY x 512 clocks

Setting the Full-Scale Current Range

The full scale current I_{FS} is a peak current setting.

The full-scale current is selected with an external reference resistor and 2 bits in the DRV_CONF register.

A standard low-power resistor with 1% accuracy is sufficient.

Three different full-scale current ranges can be configured to adapt to different motor sizes and applications.

This is needed to benefit from a best possible current control resolution.

Therefore, connect a resistor from I_{REF} to GND to set the full-scale chopping current I_{FS}.

Bits 1..0 in DRV_CONF register define the typical ON resistance of the driver stage and further control the full-scale range based on the external resistor.

The following equation shows the full-scale current I_{FS} as a function of the R_{REF} resistor connected to pin I_{REF} and the DRV_CONF register bit setting.

The proportionality constant K_{IFS} depends on the selected full-scale range setting (DRV_CONF register bits 1..0). The external resistor R_{REF} can range between 12kΩ and 60kΩ.

$I_{FS} = K_{IFS}(KV) / R_{REF}(k\Omega)$

Table 16. I_{FS} Full-Scale Peak Range Settings (Example for R_{REF} = 12kΩ)

REGISTER CONFIG DRV_CONF bits 1..0	K_{IFS} (A x kΩ)	MAX. FS SETTING (PEAK)	TYPICAL $R_{DS(ON)}$ (HS + LS)	NOTES
11	36	3A	0.23Ω	Optimized efficiency and extended operating range up to 3A$_{FS}$.
10	36	3A	0.23Ω	Optimized efficiency and extended operating range up to 3A$_{FS}$.
01	24	2A	0.27Ω	Reduced operating range up to 2A$_{FS}$. When high accuracy at lower current is required.
00 (default)	11.75	1A	0.40Ω	Reduced operating range up to 1A$_{FS}$. When high accuracy at low current is required.

The following table is a matrix of different reference resistor values (at pin I_{REF}) versus the different pin configurations for the full-scale current. The resulting maximum RMS current is given in each cell.

Table 17. I_{FS} Full-Scale RMS Current in Ampere (A RMS) Based on DRV_CONF Bits 1..0 Setting and Different R_{REF}

R_{REF} (kΩ)	MAX FULL SCALE CURRENT (A RMS) BASED ON DRV_CONF BITS 1..0 SETTING AND K_{IFS} (A x kΩ)			
	DRV_CONF BITS 1..0 = 11	DRV_CONF BITS 1..0 = 10	DRV_CONF BITS 1..0 = 01	DRV_CONF BITS 1..0 = 00
	K_{IFS} = 36	K_{IFS} = 36	K_{IFS} = 24	K_{IFS} = 11.75
12	2,12	2,12	1,41	0,69
15	1,70	1,70	1,13	0,55
16	1,59	1,59	1,06	0,52
18	1,41	1,41	0,94	0,46
22	1,16	1,16	0,77	0,38
24	1,06	1,06	0,71	0,35
27	0,94	0,94	0,63	0,31
33	0,77	0,77	0,51	0,25
39	0,65	0,65	0,44	0,21
47	0,54	0,54	0,36	0,18
48	0,53	0,53	0,35	0,17
56	0,45	0,45	0,30	0,15

Velocity-Based Mode Control

The TMC5240 allows the configuration of different chopper modes and modes of operation for optimum motor control. Depending on the motor load, the different modes can be optimized for lowest noise and high precision, highest dynamics, or maximum torque at highest velocity. Some of the features like CoolStep or StallGuard2 are useful in a limited velocity range. A number of velocity thresholds allow combining the different modes of operation within an application requiring a wide velocity range.

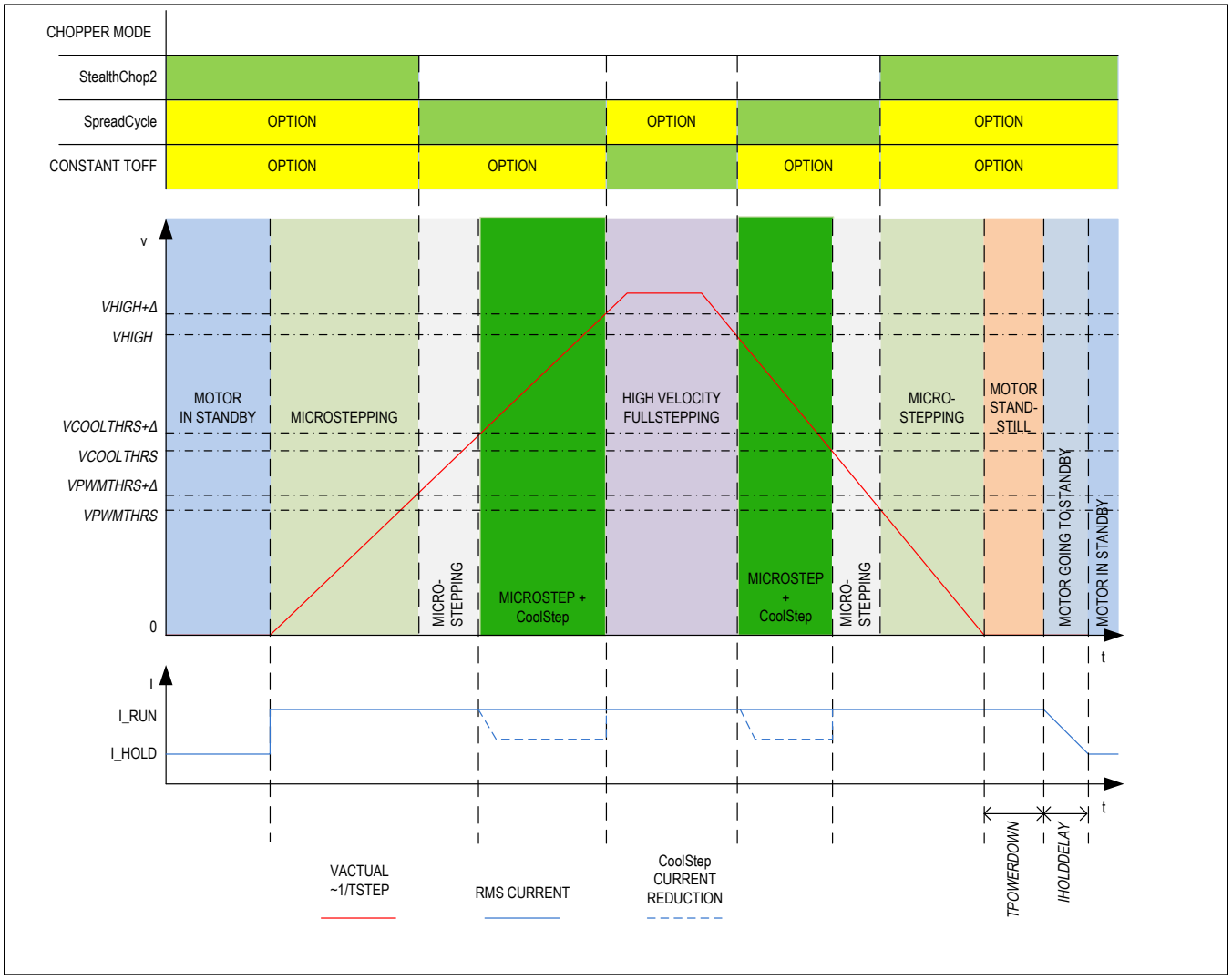


Figure 17. Choice of Velocity-Dependent Modes

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

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

Table 18. Velocity-Based Mode Control Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
stst	Indicates motor stand still in each operation mode. Time is 2^{20} clocks after the last step pulse. Default / reset: 0	0/1	Status bit, read-only
TPOWER DOWN	This is the delay time after stand still (stst) of the motor to motor current power down. Time range is about 0 to 4 seconds (with f_{CLK} = 16MHz). Setting 0 is no delay, 1 is one clock cycle delay. Further increment is in discrete steps of 2^{18} clock cycles. Default: 0xA	0…255	Time in multiples of 2^{18} * t_{CLK}
TSTEP	Actual measured time between two 1/256 microsteps derived from the step input frequency in units of 1/fCLK. Measured value is (2^{20}) - 1 in case of overflow or stand still. Default / reset: 0	0… 1048575	Status register, read-only. Actual measured step time in multiple of t_{CLK}
TPWMTHRS	TSTEP ≥ TPWMTHRS ● StealthChop2 PWM mode is enabled, if configured ● DcStep is disabled Default: 0	0… 1048575	Setting to control the upper velocity threshold for operation in StealthChop2
TCOOLTHRS	TCOOLTHRS ≥ TSTEP ≥ THIGH: ● StallGuard2 and CoolStep are enabled, if configured ● StealthChop2 voltage PWM mode is disabled TCOOLTHRS ≥ TSTEP ● StallGuard2 stall output signal is enabled (if configured) for use with external controller Default: 0	0… 1048575	Setting to control the lower velocity threshold for operation with CoolStep and StallGuard2
THIGH	TSTEP ≤ THIGH: ● CoolStep is disabled (motor runs with normal current scale) ● StealthChop2 voltage PWM mode is disabled ● If vhighchm is set, the chopper switches to chm = 1 with TFD = 0 (constant off time with slow decay, only). ● Chopper sync is switched off (SYNC = 0) ● If vhighfs is set, the motor operates in fullstep mode and the stall detection becomes switched over to DcStep stall detection. Default: 0	0… 1048575	Setting to control the upper threshold for operation with CoolStep and StallGuard2 as well as optional high velocity step mode
small_ hysteresis	Hysteresis for step frequency comparison based on TSTEP (lower velocity threshold) and (TSTEP x 15/16) - 1 respectively (TSTEP x31/ 32) - 1 (upper velocity threshold) Default: 0	0 1	Hysteresis is 1/16 Hysteresis is 1/32
vhighfs	This bit enables switching to fullstep, when VHIGH is exceeded. Switching takes place only at 45° position. The fullstep target current uses the current value from the microstep table at the 45° position. Default: 0	0 1	No switch to fullstep Fullstep at high velocities

Table 18. Velocity-Based Mode Control Parameters (continued)

vhighchm	This bit enables switching to chm = 1 and fd = 0, when VHIGH is exceeded. This way, a higher velocity can be achieved. Can be combined with vhighfs = 1. If set, the TOFF setting automatically becomes doubled during high velocity operation to avoid doubling of the chopper frequency. Default: 0	0	No change of chopper mode
		1	Classic const. Toff chopper at high velocities
en_pwm_mode	StealthChop2 voltage PWM enable flag (depending on velocity thresholds). Switch from off to on state while in standstill, only. Default: 0	0	No StealthChop2
		1	StealthChop2 active if configured and TSTEP > TPWMTHRS

Ramp Generator

The ramp generator allows motion based on target position or target velocity. It automatically calculates the motion profile, taking into account acceleration and velocity settings. The TMC5240 integrates a new type of ramp generator, which offers faster machine operation compared to the classical linear acceleration ramps. The EightPoint ramp generator allows adapting the acceleration ramps to the torque curves of a stepper motor. It uses three different acceleration settings each for the acceleration phase and deceleration phase to allow for jerk minimized ramps.

Real Word Unit Conversion

The TMC5240 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.

v[TMC5240] and a[TMC5240] are internal units of TMC5240. These values must be written to the velocity/acceleration registers of TMC5240.

Calculator tools are available from the product website and evaluation tools.

Table 19. Ramp Generator Parameters vs. Units

PARAMETER / SYMBOL	UNIT	DESCRIPTION
f_{CLK}	[Hz]	Clock frequency of the TMC5240
s	[s]	Second
US	microstep	
FS	fullstep	
USC - microstep count	-	Microstep resolution in number of microsteps (that is, the number of microsteps between two fullsteps – normally 256)
FSC - fullstep count	-	Motor fullsteps per rotation, example, 200
µstep velocity v[Hz]	microsteps / s	v[Hz] = v[TMC5240] * (f_{CLK}[Hz] / 2^{24})
µstep acceleration a[Hz/s]	microsteps / s^2	a[Hz/s] = a[TMC5240] * f_{CLK}[Hz]^2 / 2^{42}
Rotations per second v[rps]	rotations / s	v[rps] = v[microsteps/s] / USC / FSC
RPS acceleration a[rps/s^2]	rotations / s^2	a[rps/s^2] = a[microsteps/s^2] / USC / FSC
Ramp steps[microsteps] = rs	microsteps	rs = (v[TMC5240])^2 / a[TMC5240] / 2^8 microsteps during linear acceleration ramp (assuming acceleration from 0 to v)

Table 19. Ramp Generator Parameters vs. Units (continued)

TSTEP, Txxx_THRS	-	TSTEP = f_{CLK} / $f_{256STEP}$ = f_{CLK} / (f_{STEP}*256/USC) = 2^{24} / (VACTUAL*256/USC) The time reference for velocity thresholds is referred to the actual 1/256 microstep frequency ($f_{256STEP}$) of the step input, respectively, velocity v[Hz].
Ramp generator update rate	[Hz]	f_{UPDATE} = f_{CLK} / 512 VACTUAL updates with this frequency.

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

Motion Profiles

Ramp Mode

The ramp generator delivers three phase acceleration and three phase deceleration ramps with additional programmable start and stop velocities.

Three different sets of acceleration and deceleration can be combined freely. The transition velocities *V1* and *V2* allow for velocity dependent switching between three acceleration and deceleration settings. A typical high velocity application uses 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 deceleration capability of the system is quicker than acceleration capability. Thus, deceleration values can be set higher in many applications. This way, operation speed of the motor in time critical applications is maximized.

As target positions and ramp parameters may be changed at any time during the motion, the motion controller always uses the optimum (fastest) way to reach the target, while sticking to the acceleration constraints set by the user. This way, the motion may automatically stop, cross zero, and drive back again. For example, when during the final deceleration phase, the target position is again changed and "pulled-in" to a closer position, and the configured deceleration value does not allow to reach the new target position directly. This case is flagged by the special flag *second_move.*

The ramp generator further supports automatic jerk-reduction, by smoothening the transition from acceleration phase to deceleration phase, and from deceleration phase to acceleration phase, by enforcing a constant velocity segment of a minimum duration (*TVMAX*), as required by mechanical jerk response. The following graphs give some examples on typical (corner) cases.

Note:

The start velocity can be set to zero, if not used.

The stop velocity can be set to a low value (1000 or down to 10), if not used. Background: if TSTOP = 0, the position might not precisely reach the configured microstep target position.

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.

Set TVMAX zero to disable jerk-reduction.

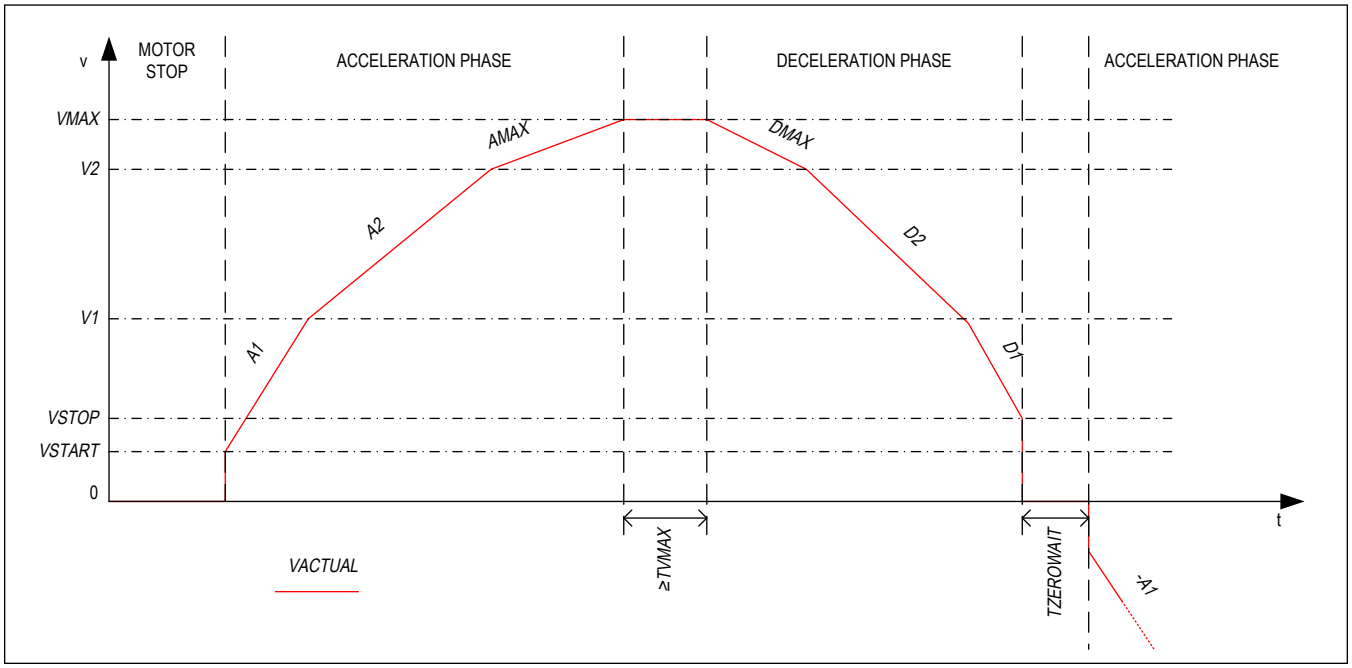


Figure 18. Ramp Generator Velocity Trace Showing Second Move into Negative Direction

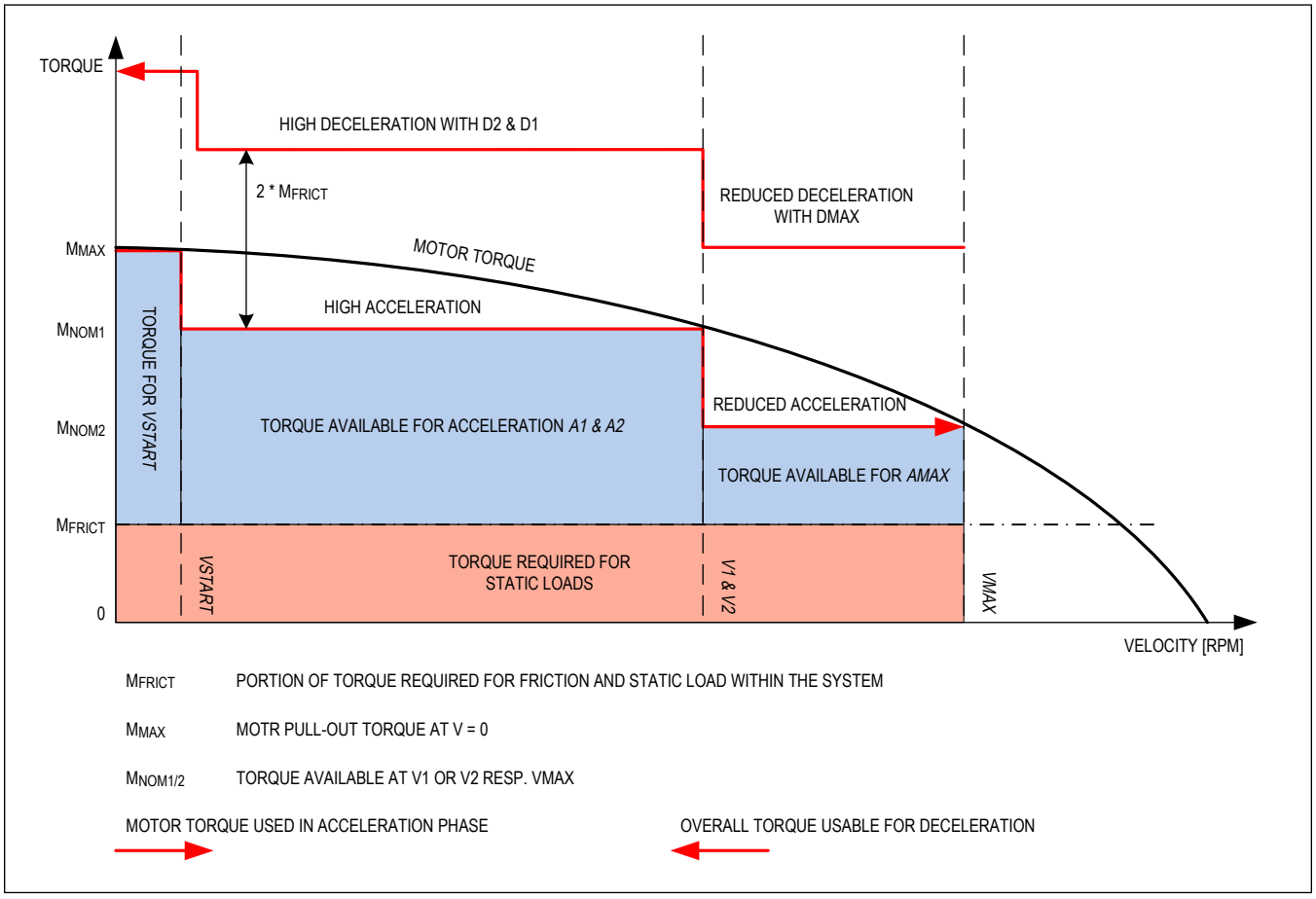


Figure 19. Illustration of Optimized Motor Torque Usage with the Ramp Generator

Eight Point Ramp

Start and Stop Velocity

When using increased levels of start and stop velocity, it becomes clear that a subsequent move into the opposite direction provides a jerk identical to *VSTART* + *VSTOP*, rather than only *VSTART*. As the motor probably is not able to follow this, 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.

The set of three acceleration and deceleration segments can be used in two ways: either for adaptation to the motor torque curve, by using higher acceleration values at lower velocity, or to reduce the jerk (change of acceleration) when transitioning from one acceleration segment to the next. For jerk optimized ramps, typically, *A1*, *D1*, *AMAX,* and *DMAX* are set to lower values than *A2* and *D2*. The most critical points with regards to jerk are the transition from acceleration to deceleration with no constant velocity segment, as well as the transition from deceleration to acceleration in case of on-the-fly change of target position.

To address both, the 8-point motion profile generator allows to enforce a constant velocity segment based on a minimum segment duration (*TVMAX*). In case this duration cannot be kept due to insufficient distance, a reduced *VMAX* (VMAX´) is calculated and is used for the constant velocity segment. Minimum VMAX´ is identical to *VSTOP*.

The following traces show the resulting velocity profiles based on different position distances for a pseudo-S-shaped configuration.

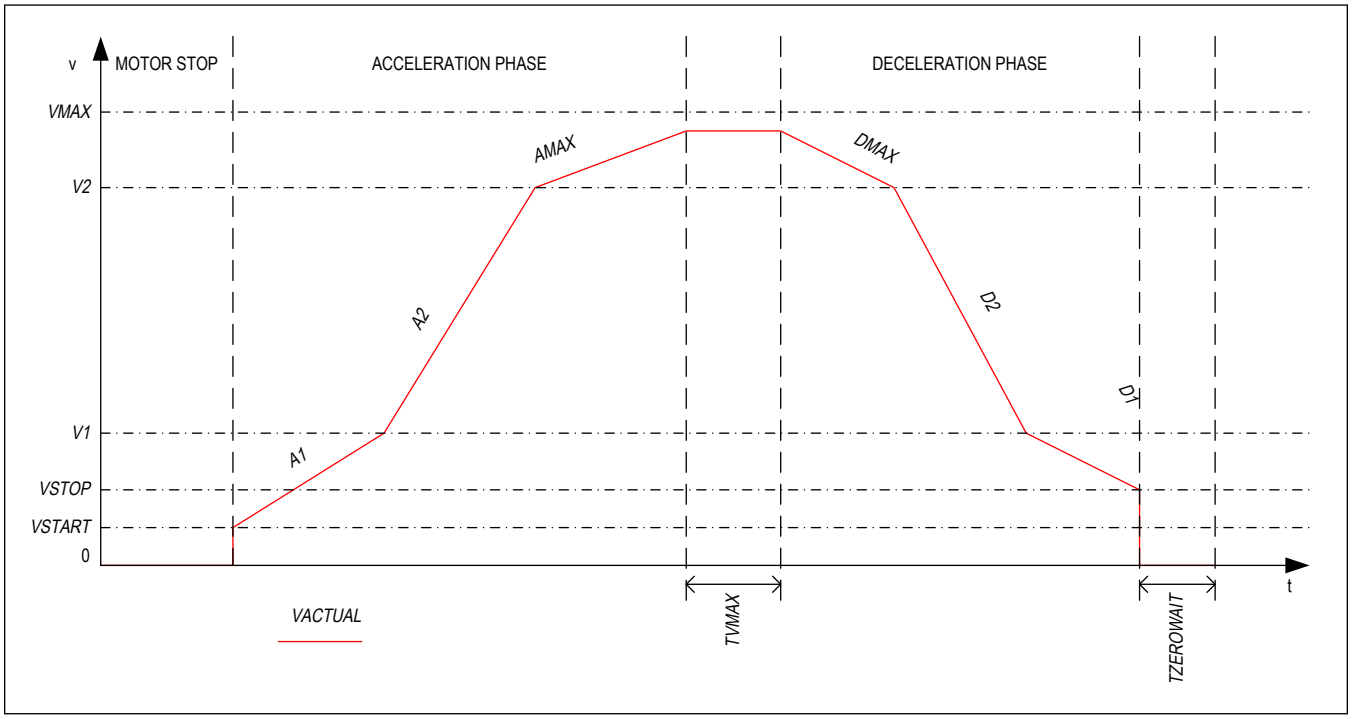


Figure 20. 8-Point Ramp with VMAX Not Reached Due to Too Low Distance

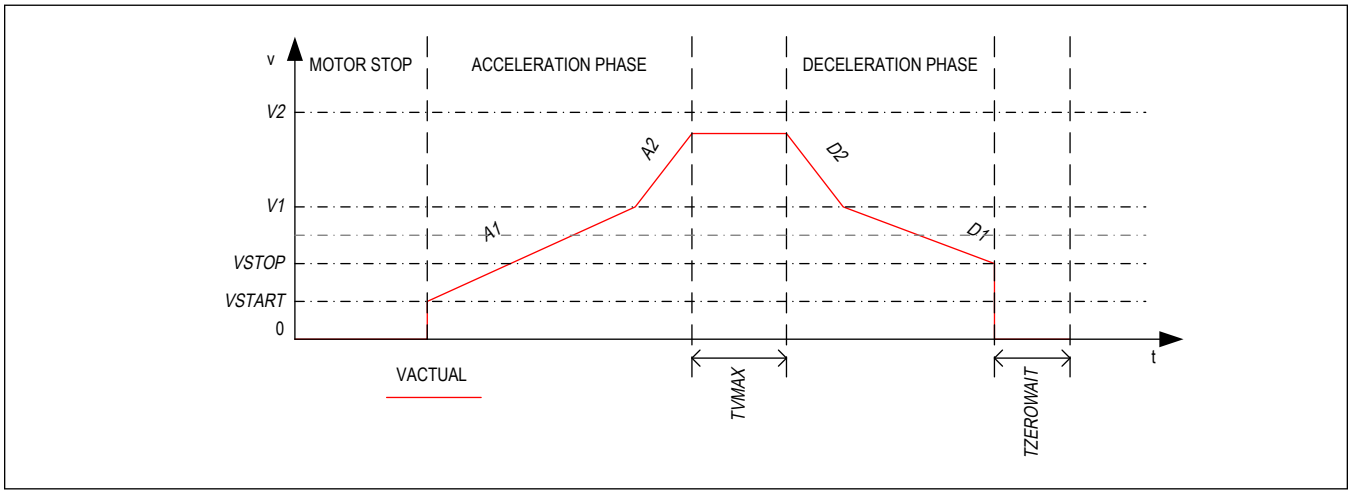


Figure 21. V2 Not Reached and No AMAX and DMAX Phase Due to Low Distance

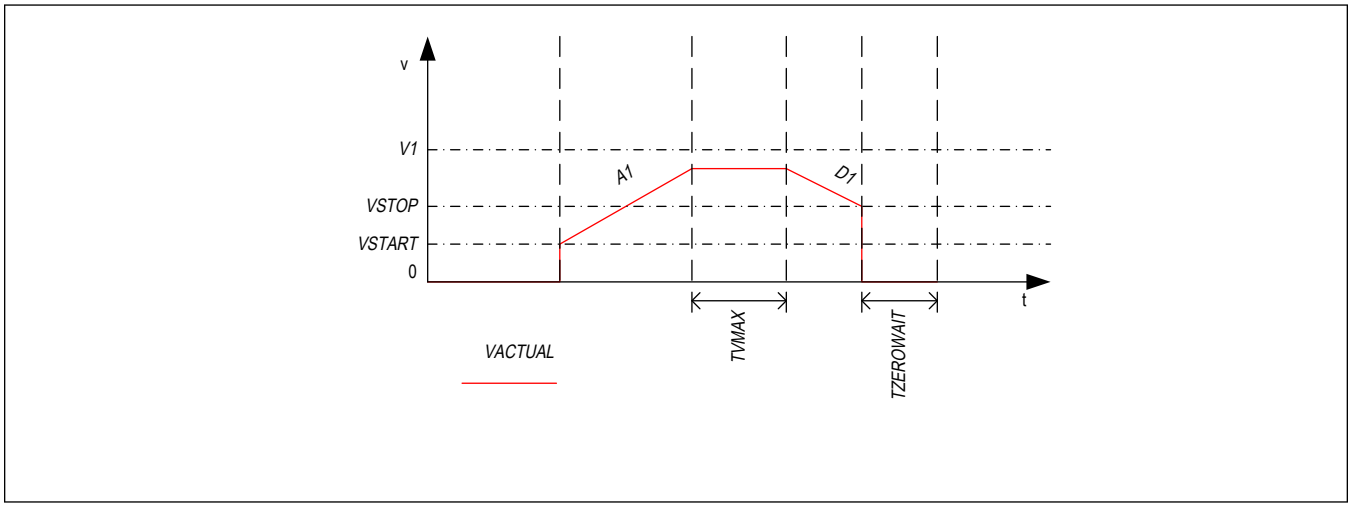


Figure 22. V1 Not Reached Due to Low Distance

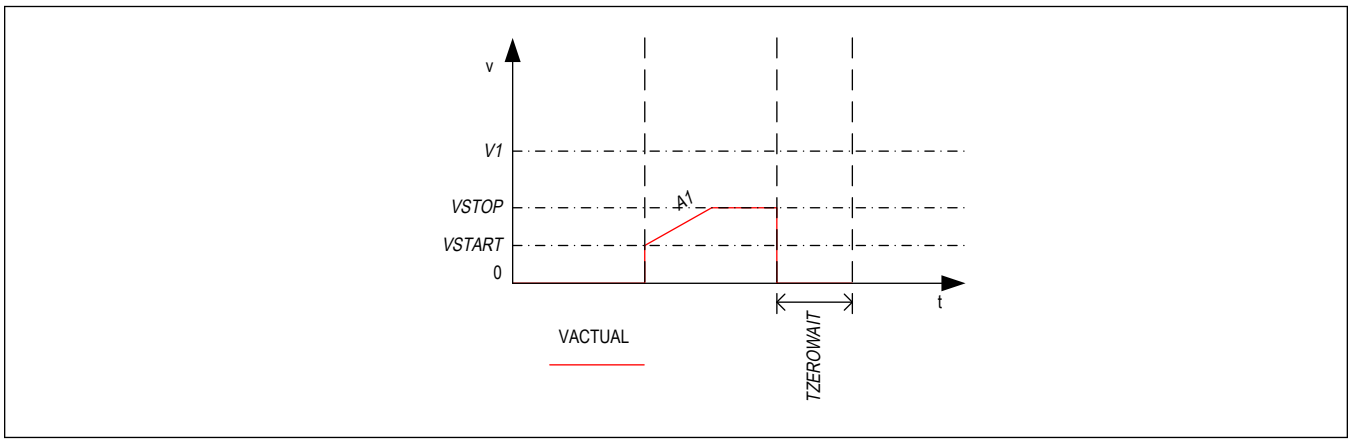


Figure 23. TVMAX Not Kept Due to Low Distance

If VSTOP is not reached due to a too short travel distance, there is only a very short linear acceleration using A1 and the ramp terminates immediately when XTARGET is reached.

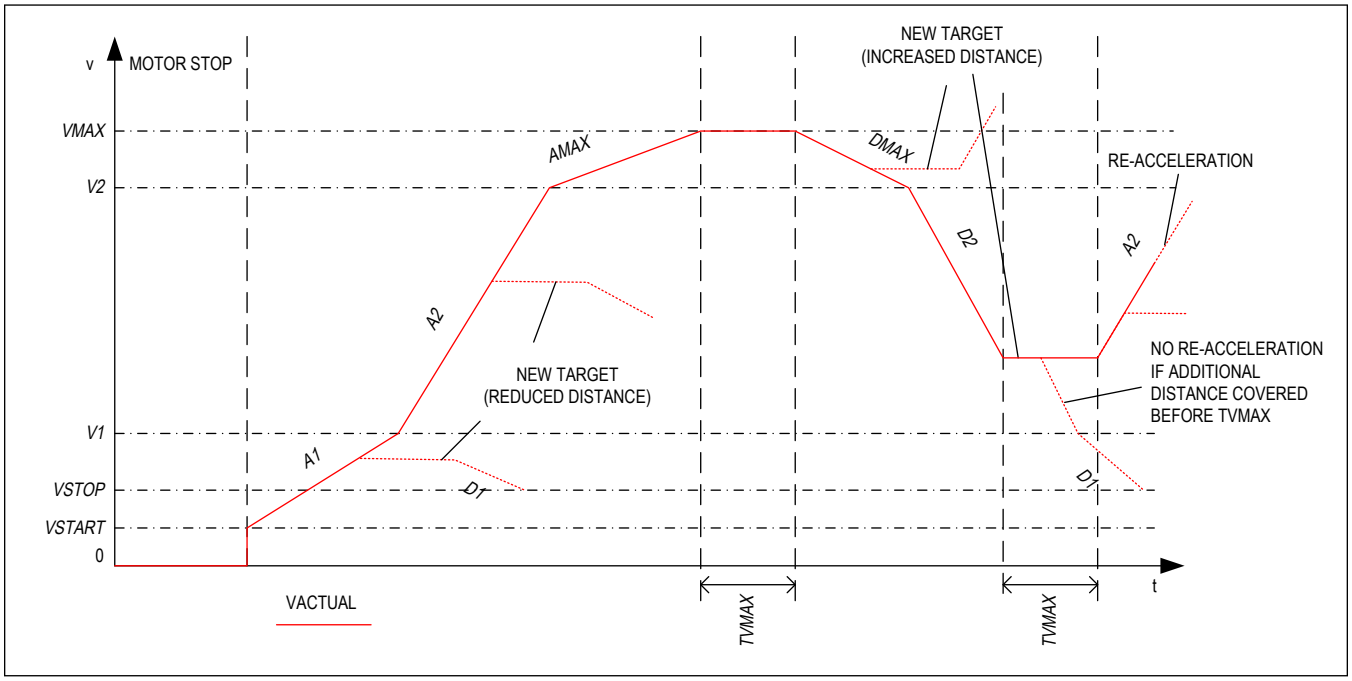


Figure 24. 8-Point Ramp Examples with On-the-Fly Target Position Change

Velocity Mode

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

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 is reached.

Early Ramp Termination

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

The options to terminate motion using acceleration settings:

1. Switch to velocity mode, set *VMAX* = 0 and *AMAX* to the desired deceleration value. This stops the motor using a linear ramp.
2. For a stop in positioning mode, set *VSTART* = 0 and *VMAX* = 0. *VSTOP* is not used in this case. The driver uses AMAX, A1, and A2 (as determined by V1 and V2) for decelerating to zero velocity.
3. For a stop using *DMAX*, *D1*, *D2,* and *VSTOP*, trigger the deceleration phase by copying *XACTUAL* to *XTARGET*. Set *TZEROWAIT* sufficiently to allow the CPU to interact during this time. The driver decelerates and eventually comes to a stop. Poll the actual velocity to terminate motion during *TZEROWAIT* time using option a) or b).
4. Activate a stop switch. This can be done with the hardware input, example, using a wired 'OR' to the stop switch input. If not using the hardware input and have tied the REFL and REFR to a fixed level, enable the stop function (*stop_l_enable*, *stop_r_enable*), and use the inverting function (*pol_stop_l*, *pol_stop_r*) to simulate the switch activation.
5. Utilize the virtual stop switches (VIRTUAL_STOP_L, VIRTUAL_STOP_R). The position comparison (X_ACTUAL vs. VIRTUAL_STOP_L/R) then triggers a stop accordingly.

Application Example: Joystick Control

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

Realize joystick control:

1. Use positioning mode to control the motion direction and set the motion limit(s). The limits might be used as the virtual stop switches. For example (*VIRTUAL_STOP_L* and *VIRTUAL_STOP_R*).
2. Modify *VMAX* depending on the joystick input at any time in the range *VSTART* to the maximum value. With *VSTART* = 0, also stop motion by setting CS_ACTUAL that takes into account the actual current scaling as defined by IHOLD and IRUN, respectively, by CoolStep*MAX*=0. The motion controller uses *A1, A2,* and *AMAX* as determined by *V1 and V2* to adapt velocity for ramping up and ramping down.
3. In case the acceleration settings are not modified, no need to rewrite *XTARGET*, just modify *VMAX*.
4. *DMAX, D1*, *D2,* and *VSTOP* can only be used when the ramp controller slows down due to reaching the target position, or when the target position is modified to point to the other direction.

Velocity Thresholds

The ramp generator provides a number of velocity thresholds coupled with the actual velocity *VACTUAL*. The different ranges allow programming the motor to the optimum step mode, coil current, and acceleration settings. Most applications do not require all the thresholds, but in principle all modes can be combined. VHIGH and VCOOLTHRS are determined by the settings *THIGH* and *TCOOLTHRS* to allow determination of the velocity when an external step source is used. *TSTEP* becomes compared to these threshold values. A hysteresis of 1/16 *TSTEP* resp. 1/32 *TSTEP* (see bit small_hysteresis in GCONF register) is applied to avoid continuous toggling of the comparison results when a jitter in the *TSTEP* measurement occurs. The upper switching velocity is higher by 1/16, resp. 1/32 of the value set as threshold. The StealthChop threshold *TPWMTHRS* is not shown. VCOOLTHRS can either be used in StealthChop2 velocity range, or in SpreadCycle velocity range.

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

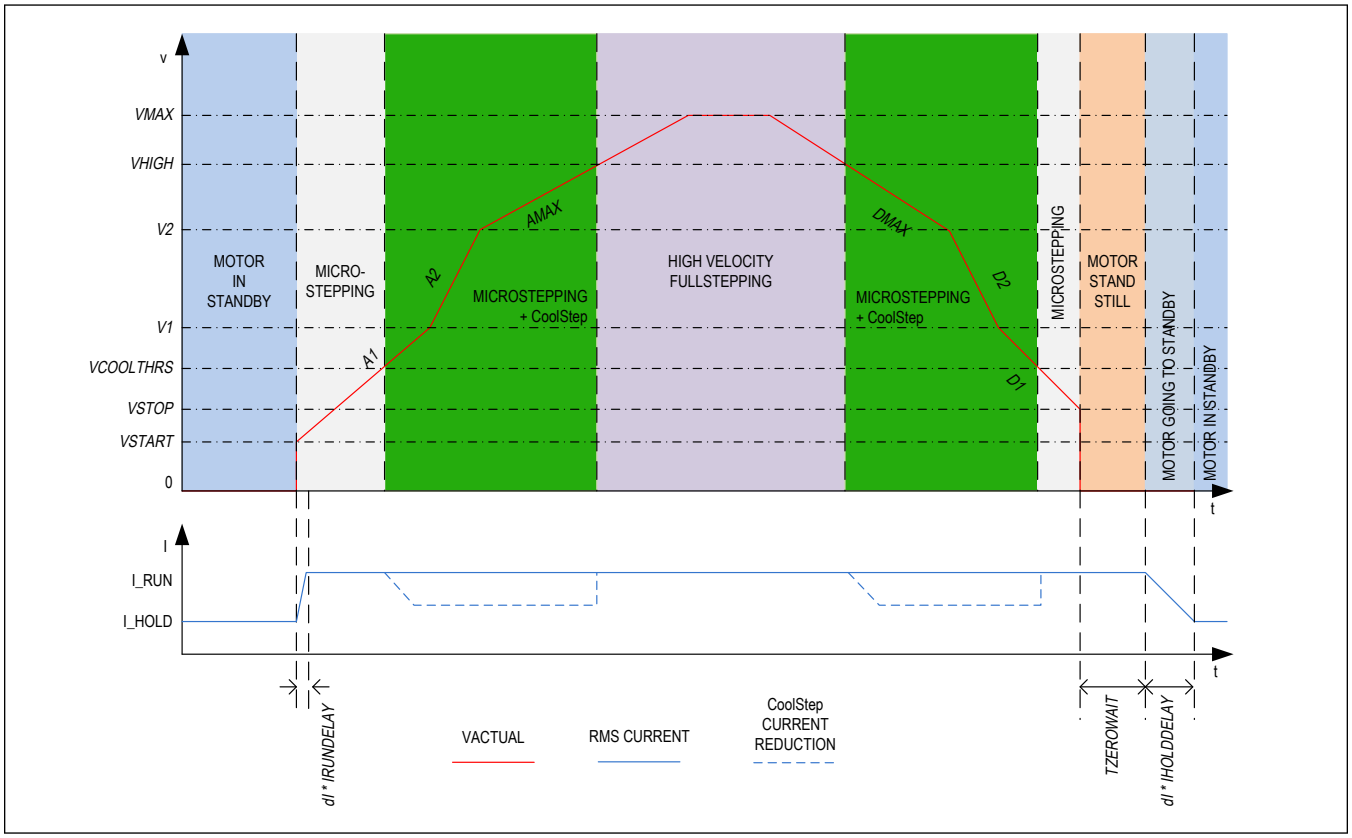


Figure 25. Ramp Generator Velocity Dependent Motor Control

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 StallGuard2, StallGuard4, or a reference switch.

In case of a linear drive, the mechanical motion range must not be exceeded. This can be ensured also for abnormal situations by enabling the stop switch functions for the left and right reference switches. 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 and for StallGuard2-based homing.
- Or, the motor can be softly decelerated to zero using deceleration settings (DMAX, V2, D2, V1, D1) using the soft stop function (bit *en_softstop = 1*).

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.

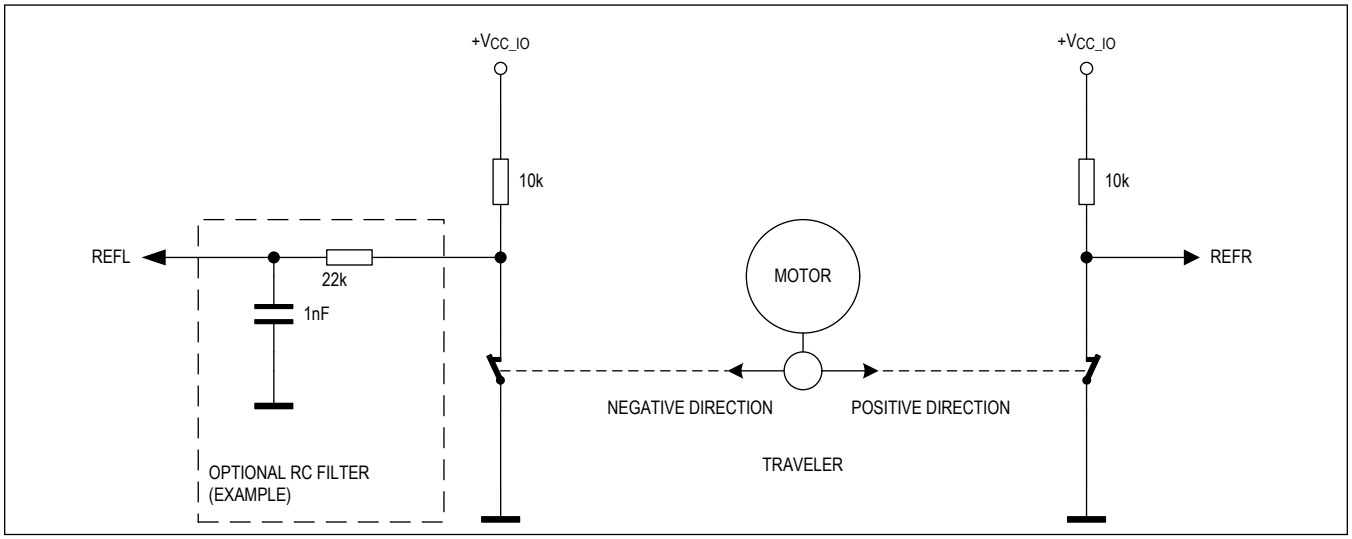


Figure 26. Using Reference Switches (Example)

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 that can be used are:

- Mechanical switches
- Photo interrupters
- Hall sensors

Be careful to select reference switch resistors matching the switch requirements!

In case of long cables, additional RC filtering might be required near the TMC5240 reference inputs. Adding an RC filter also reduces the danger of destroying the logic level inputs by wiring faults, but it adds 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, example, by moving away from the switch.
2. Activate position latching upon the desired switch event and activate motor (soft) stop upon active switch. StallGuard2-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 and to a more positive position for a right switch. Timeout this motion by using a position ramping command.
4. As soon as the switch is hit, the position becomes latched and the motor is stopped. Wait until the motor is in standstill again by polling the actual velocity *VACTUAL* or checking *vzero* or the *standstill* flag.
5. Switch the ramp generator to hold mode and calculate the difference between the latched position and actual position. For StallGuard2-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 brings back the motor exactly to the switching point. In case StallGuard2 is used for homing, a read access to *RAMP_STAT* clears the StallGuard2 stop event *event_stop_sg* and releases the motor from the stop condition.

Homing with a Third Switch:

Some applications use an additional home switch, which operates independently of the mechanical limit switches. The encoder functionality of the TMC5240 provides an additional source for position latching. It allows using the N channel input to snapshot *XACTUAL* with a rising or falling edge event, or both. This function also provides an interrupt output.

1. Activate the latching function (*ENCMODE*: Set *ignoreAB*, *clr_cont*, *neg_edge* or *pos_edge* and *latch_x_act*). The latching function can then trigger the interrupt output (check by reading *n_event* in *ENC_STATUS* when interrupt is

signaled at DIAG0).
2. Move to the direction, where the N channel switch should be. In case the motor hits a stop switch (REFL or REFR) before the home switch is detected, reverse the motion direction.
3. Read out *XLATCH* once the switch is triggered. It gives the position of the switch event.
4. After detection of the switch event, stop the motor, and subtract *XLATCH* from the actual position. A detailed description of the required steps is in the homing procedure above.

Virtual Reference Switches

The TMC5240 supports virtual reference switches to support applications that only have a single or no reference switch (StallGuard homing) to safely limit the physical motion range. The virtual stop switches become active when the actual motor position (*XACTUAL*) exceeds *VIRTUAL_STOP_R* during a movement into positive direction or falls below *VIRTUAL_STOP_L* during a movement in negative direction. Enable virtual stop switches by setting *en_virtual_stop_l* resp. *en_virtual_stop_r*. Each virtual stop switch blocks only motion into the respective direction.

Optionally, the virtual stops can be switched to monitor the encoder position (*X_ENC*). To select, set *virtual_stop_enc*.

Set the values of the virtual stops (*VIRTUAL_STOP_R, VIRTUAL_STOP_L*) with sufficient distance to the overflow/ underflow ranges of the signed 32-bit motion range to allow motor deceleration, in case of soft deceleration used.

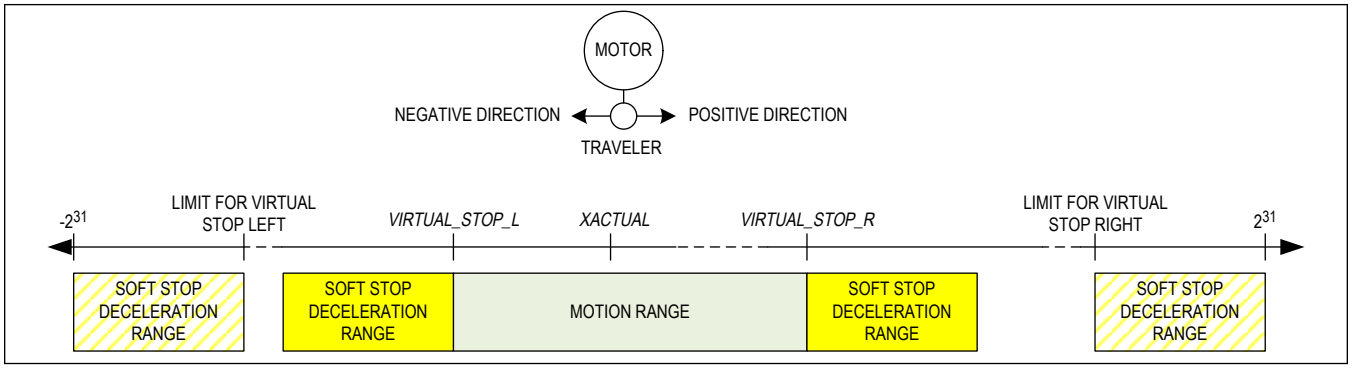


Figure 27. Virtual Stop Switches and Limit Visualization

Ramp Generator Response Time

The ramp generator is realized in hardware and executes commands in less than a microsecond, switching over to the desired mode and target values taking effect.

A velocity accumulator updates the velocities each 512 clock cycles, based on the actual acceleration setting, to give a smooth acceleration.

However, at low motion velocities and low acceleration settings, example, at the start of positioning ramp (VSTART) or its stop (VSTOP), the actual step pulse rate is comparatively low.

Therefore, a significant delay adds for execution of the first and last steps, as determined by the selected microstep velocity.

For example, at a microstep velocity of 10Hz, 100ms expires between each two steps. As (at least a part) the last microstep of a ramp is executed with a velocity equal to VSTOP, this can cause significant delay to reach the target position. Set VSTOP in a range of minimum 100 to 1000 for quick ramp termination (100 yields roughly <10ms, 1000 roughly <1ms).

External STEP/DIR Driver

The TMC5240 allows using the internal ramp generator to control an external STEP/DIR driver like the ADI-Trinamic TMC262C, TMC2160, or TMC2240 for powerful stepper applications. In this configuration, the internal driver is normally not used, but it may be used in addition to the external driver, example, when two motors shall move synchronously.

The DIAG0 and DIAG1_SW outputs are enabled for STEP and DIR output by setting *GCONF* flags *diag0_nint_step* and

diag1_nposcomp_dir. Additional internal driver features like DcStep and automatic motor current control are not available in this mode because there is no feedback from the external driver to the TMC5240.

A low level on DIAG1_SW corresponds to a step in positive direction (*XACTUAL* increasing), a high level to a step into negative direction (*XACTUAL* decreasing).

Two options for the external STEP signal are selectable:

1. Double edge (*GCONF.length_step_pulse* = 0). Each toggle of the STEP output corresponds to a step. DIR is valid at least one CLK cycle in advance.Enable the *dedge* function on the external driver to match.
2. Single edge (*GCONF.length_step_pulse* > 0). Each positive pulse on the STEP output corresponds to a step. The pulse length is controlled by *length_step_pulse* in CLK cycles. The DIR signal only changes in between of step pulses and keeps a minimum setup time equal to the STEP pulse length in advance to the next step pulse.

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

Position Compare Functions

The position compare function allows triggering of external events synchronously to the motor motion. The function outputs an active high level, whenever XACTUAL = X_COMPARE. The duration of the pulse thus corresponds to the time that XACTUAL matches X_COMPARE, that is, the duration of one microstep at the actual velocity.

A repetition function allows triggering a periodic compare pulse. Use X_COMPARE_REPEAT to program the desired period (microstep distance) with up to 2^{24} - 1 steps.

Table 20. X_COMPARE_REPEAT Options and Periodic Pulse Behavior

VALUE	DESCRIPTION
0	No repetition.
>1	Results in a continuous pulse train, once the first compare position has been passed until the motion direction is changed. Distance between compare pulses: 2 to 2^{24} - 1 microsteps

Whenever the position compare condition XACTUAL = X_COMPARE goes from a true to a false state, X_COMPARE becomes automatically incremented, resp., decremented by the value programmed to X_COMPARE_REPEAT. The decision to increment or decrement X_COMPARE is taken based on the actual motion direction. This way, the first compare event in a motion is given by the content of X_COMPARE, and the distance of subsequent events is programmed by X_COMPARE_REPEAT. When changing motion direction, or whenever the next pulse position shall be altered by software, be sure to first disable the repetition mechanism by setting X_COMPARE_REPEAT = 0. In a next step, reprogram X_COMPARE with the next desired pulse position, because the previously automatically generated next position still lies in the previous motion direction and is not hit. Following the write access to X_COMPARE, the repetition mechanism can be enabled once again. The step to first disable the repetition mechanism is required, in case X_COMPARE is identical to X_ACTUAL during the write access to X_COMPARE, as it also triggers the repetition mechanism.

StallGuard2 Load Measurement

To fit different motor control schemes, the TMC5240 offers two types of StallGuard sensorless load detection schemes, covering the two basic chopper modes. StallGuard2 works in SpreadCycle operation, while StallGuard4 is optimized for StealthChop2 operation.

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 the load on the motor increases, the StallGuard value (SG_RESULT) decreases. Tuning is required to properly detect stalls. Set the StallGuard threshold (SGTHRS) such that SG_RESULT reaches 0 (or near to 0) when the motor becomes overloaded/stalls.

Hint: To use StallGuard2 and CoolStep, the StallGuard2 sensitivity should first be tuned using the SGT setting!

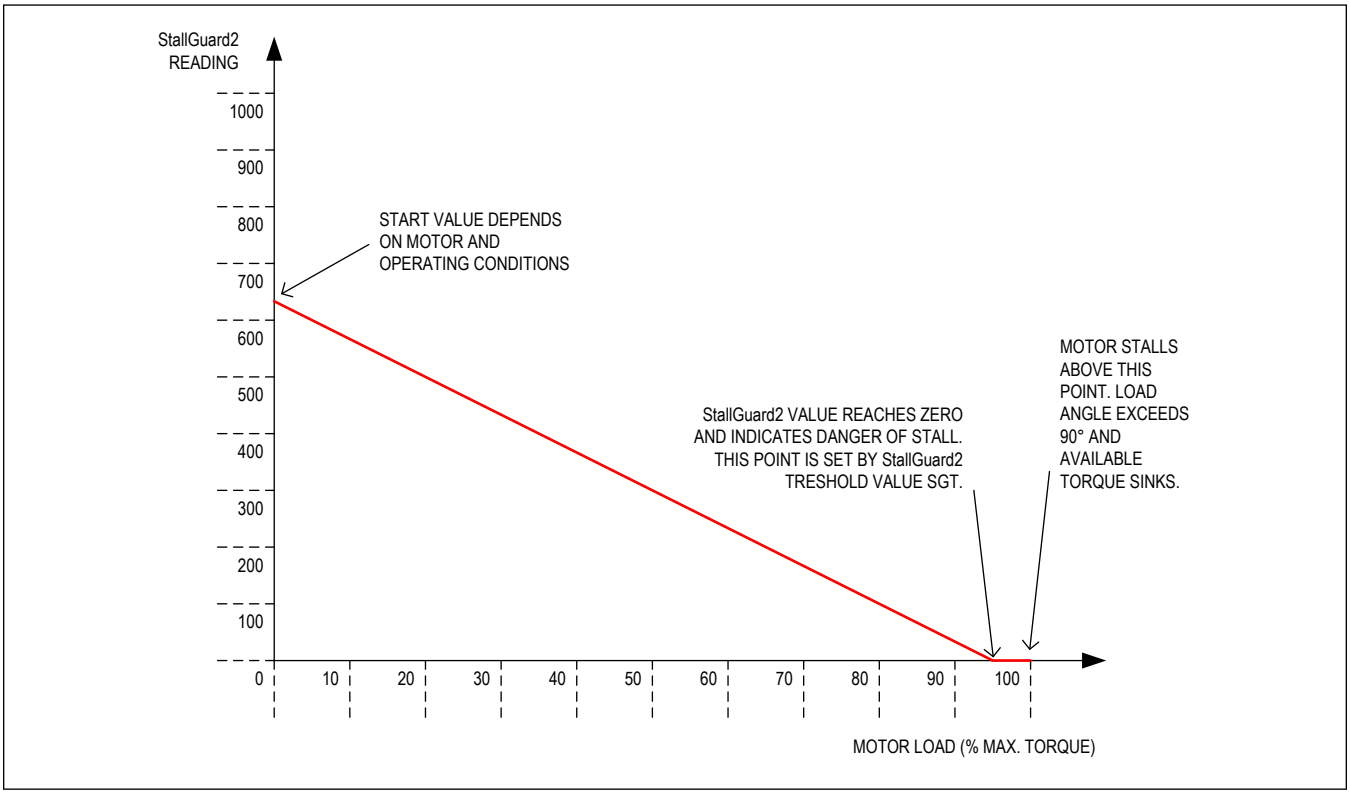


Figure 28. Function Principle of StallGuard2

Table 21. StallGuard2-Related Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
SGT	This signed value controls the StallGuard2 threshold level for stall detection and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value working with most motors. A higher value makes StallGuard2 less sensitive and requires more torque to indicate a stall.	0	Indifferent value
		+1…+63	Less sensitivity
		-1…-64	Higher sensitivity
sfilt	Enables the StallGuard2 filter for more precision of the measurement. If set, reduces the measurement frequency to one measurement per electrical period of the motor (4 fullsteps).	0	Standard mode
		1	Filtered mode
STATUS WORD	**DESCRIPTION**	**RANGE**	**COMMENT**
SG_RESULT	This is the StallGuard2 result. A higher reading indicates less mechanical load. A lower reading indicates a higher load and thus a higher load angle. Tune the SGT setting to show a SG_RESULT reading of roughly 0 to 100 at maximum load before motor stall.	0…1023	0: Highest load low value: high load high value: less load

Tuning StallGuard2 Threshold SGT

The StallGuard2 value *SG_RESULT* is affected by motor-specific characteristics and application-specific demands on

load, velocity, supply voltage, and current level. 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, supply voltage, and current setting for the application and monitor *SG_RESULT*.
2. Apply slowly increasing mechanical load to the motor. If the motor stalls before *SG_RESULT* reaches zero, decrease *SGT*. If *SG_RESULT* reaches zero before the motor stalls, increase *SGT*. A good *SGT* starting value is zero. *SGT* is signed. So, it can have negative or positive values.
3. Now enable *sg_stop* and make sure that the motor is safely stopped whenever it is stalled. Increase *SGT* if the motor stops before a stall occurs. Restart the motor by disabling *sg_stop* or by clearing *event_stop_sg* in the *RAMP_STAT* register (write to clear).
4. The optimum setting is reached when *SG_RESULT* is between 0 and roughly 100 at increasing load shortly before the motor stalls, and *SG_RESULT* increases by 100 or more without load. *SGT* in most cases can be tuned for a certain motion velocity or a velocity range. Make sure that the setting works reliable in a certain range (example, 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 factors 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 (that is, a few to a few fullsteps per second) and target operation current and supply voltage. In this velocity range, there is not much dependence of *SG_RESULT* on the motor load, because the motor does not generate significant back EMF. Therefore, mechanical load does not make a big difference on the result.
2. Switch on *sfilt*. Now increase *SGT* starting from 0 to a value where *SG_RESULT* starts rising. With a high *SGT*, *SG_RESULT* rises up to the maximum value. Reduce again to the highest value, where *SG_RESULT* stays at 0. Now the *SGT* value is set as sensibly as possible. When you see *SG_RESULT* increasing at higher velocities, there is useful stall detection.
3. *SG_RESULT* goes to zero when the motor stalls and the ramp generator can be programmed to stop the motor upon a stall event by enabling *sg_stop* in *SW_MODE*. Set *TCOOLTHRS* to match the lower velocity threshold, where StallGuard delivers a good result to use *sg_stop*.

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.

The system clock frequency affects *SG_RESULT*. An external crystal-stabilized clock should be used for applications that demand the highest performance. The power supply voltage also affects *SG_RESULT*. So, tighter regulation results in more accurate values. *SG_RESULT* measurement has a high resolution, and there are a few ways to enhance its accuracy, as described in the following sections.

Variable Velocity Limits TCOOLTHRS and THIGH

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

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

The velocity limits *VHIGH* and *VCOOLTHRS* are determined by the settings *THIGH* and *TCOOLTHRS*.

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

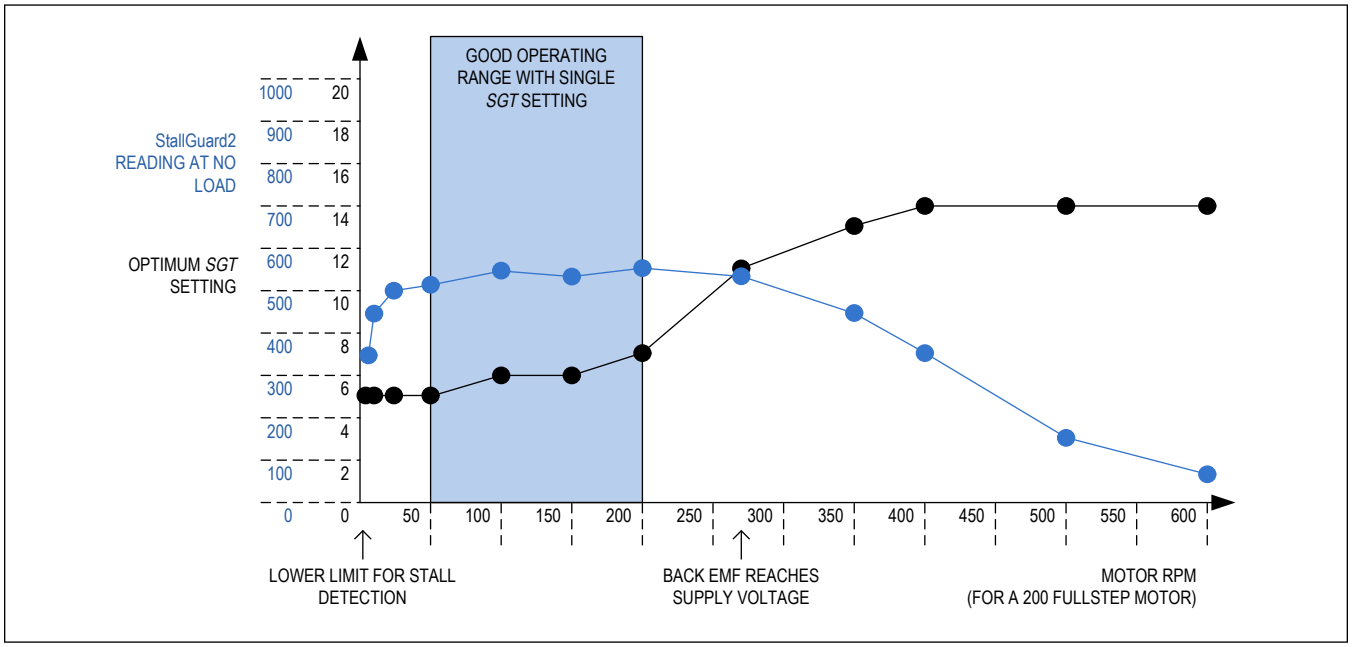


Figure 29. Example: Optimum SGT Setting and StallGuard2 Reading with an Example Motor

Small Motors with High Torque Ripple and Resonance

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

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_RESULT* at increasing temperature, as motor efficiency is reduced.

Accuracy and Reproducibility of StallGuard2 Measurement

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

$$\text{stallGuard2 measurement error} = +/- \max(1,|SGT|)$$

StallGuard2 Update Rate and Filter

The StallGuard2 measurement value *SG_RESULT* is updated with each fullstep of the motor. This is enough to safely detect a stall because a stall always means the loss of four fullsteps. 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.

Detecting a Motor Stall

For best stall detection, work without StallGuard2 filtering (*sfilt* = 0). To safely detect a motor stall, the stall threshold

must be determined using a specific *SGT* setting. Therefore, the maximum load needs to be determined, which the motor can drive without stalling. At the same time, monitor the *SG_RESULT* value at this load, example, 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_RESULT* value without load and with maximum load. They should show a difference of at least 100 or a few 100, which shall be largely compared to the offset. If the *SGT* value is set in a way that a reading of 0 occurs at maximum motor load, the stall can be automatically detected to issue a motor stop. In the moment of the step resulting in a step loss, the lowest reading is visible. After the step loss, the motor vibrates and shows a higher *SG_RESULT* reading.

Homing with StallGuard2

The homing of a linear drive requires moving the motor into the direction of a hard stop. As StallGuard2 needs a certain velocity to work (as set by *TCOOLTHRS*), make sure that the start point is far enough from the hard stop to provide the distance required for the acceleration phase. After setting up *SGT* and the ramp generator registers, start a motion into the direction of the hard stop and activate the stop on stall function (set *sg_stop* in *SW_MODE*). Once a stall is detected, the ramp generator stops motion and sets *VACTUAL* zero, stopping the motor. The stop condition also is indicated by the flag *StallGuard* in *DRV_STATUS*. After setting up new motion parameters to prevent the motor from restarting right away, StallGuard2 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* restarts the motor after expiration of *TZEROWAIT* in case the motion parameters are not modified.

Limits of StallGuard2 Operation

StallGuard2 does not operate reliably at extreme motor velocities: Very low motor velocities (for many motors, less than 1Rps) generate a low back EMF and make the measurement unstable and dependent on environment conditions (temperature, etc.). The automatic tuning procedure described earlier compensates for this. Other conditions also lead to extreme settings of *SGT* and poor response of the measurement value *SG_RESULT* to the motor load.

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

StallGuard4 Load Measurement

StallGuard4 is optimized for operation with StealthChop2, while its predecessor StallGuard2 works with SpreadCycle.

Anyway, the function is similar: Both deliver a load value, going from a high value at low load, to a low value at high load.

While StallGuard2 becomes tuned to show a “0” reading for stall detection, StallGuard4 uses a comparison-value to trigger stall detection, rather than shifting the measurement result by applying an offset.

StallGuard4 provides an accurate measurement of the load on the motor and can be used for stall detection, load estimation, as well as CoolStep load-adaptive current reduction. The StallGuard4 measurement value changes linearly over a wide range of load, velocity, and current settings, as shown in the next figure. When approaching maximum motor load, the value goes down to a motor-specific lower value. 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.

To use StallGuard4, check the sensitivity of the motor at border conditions.

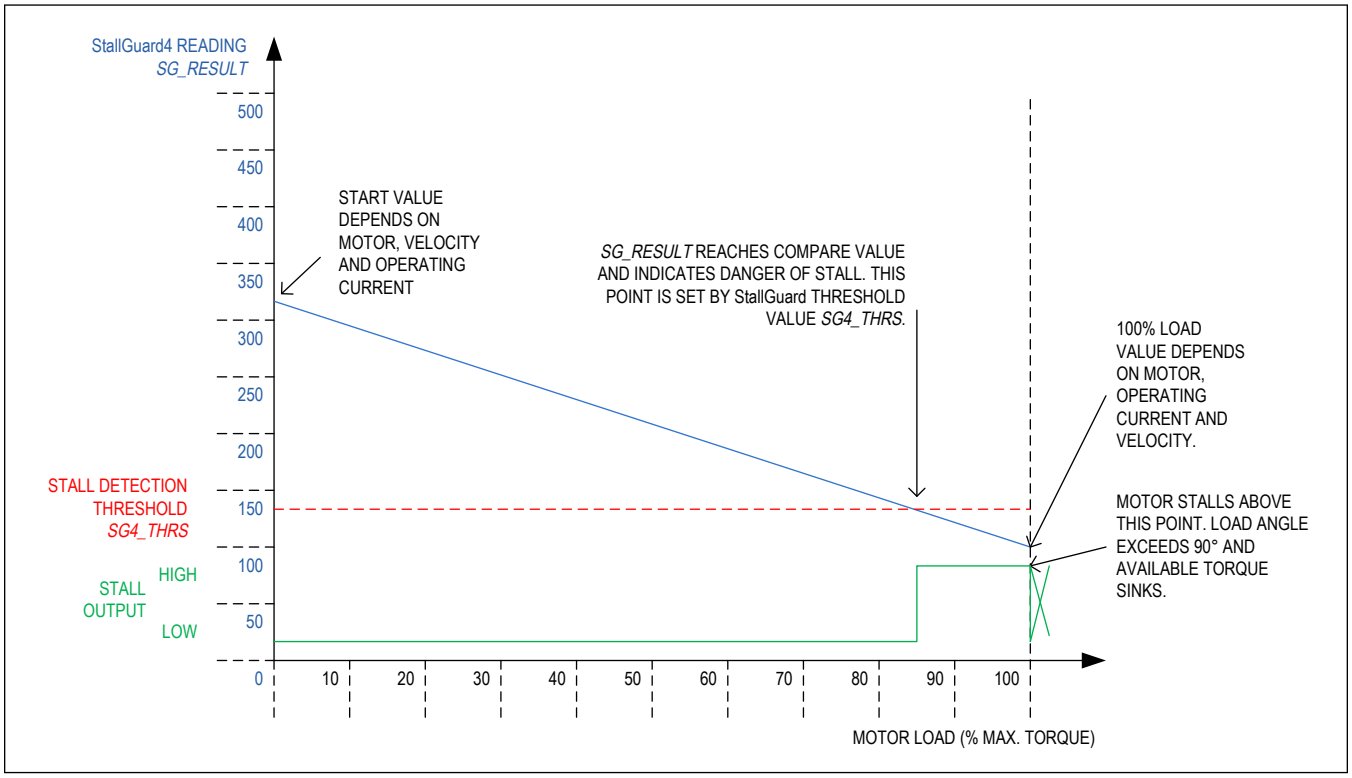


Figure 30. StallGuard4 Mode of Operation

Table 22. StallGuard4-Related Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
SG4_THRS	This value controls the StallGuard4 threshold level for stall detection. It compensates for motor-specific characteristics and controls sensitivity. A higher value gives a higher sensitivity. A higher value makes StallGuard4 more sensitive and requires less torque to indicate a stall.	0… 255	This value is compared to *SG4_RESULT*. The stall output becomes active if *SG4_RESULT* falls below this value.
STATUS WORD	**DESCRIPTION**	**RANGE**	**COMMENT**
SG4_RESULT	This is the *StallGuard4 result*. A higher reading indicates less mechanical load. A lower reading indicates a higher load and thus a higher load angle. This value becomes generated independent of the enabling conditions like the actual chopper mode and velocity thresholds like *VCOOLTHRS*. The result is calculated from *SG4_IND_x* measurements, adding one bit for higher precision and similar range as StallGuard2.	0…510	Low value: highest load High value: low/no load
SG4_IND_3 *SG4_IND_2* *SG4_IND_1* *SG4_IND_0*	Individual measurements for motor phase A falling (*SG4_IND_0*)/rising (*SG4_IND_1*) transition resp. phase B falling (*SG4_IND_2*)/rising (*SG4_IND_3*) transition. Individual measurements are available in filtered mode, only (*sg4_filt_en* = 1). *SG4_IND_0* covers all cases in unfiltered mode (*sg4_filt_en* = 0).	0....255	Low value: highest load High value: low/no load

Table 22. StallGuard4-Related Parameters (continued)

sg4_filt_en	0: Unfiltered operation, *SG4_RESULT* updates with each fullstep. 1: Filtered operation, *SG4_IND_0...3* available, *SG4_RESULT* gives the average of last four *SG4_IND_x* measurements.	0 1	0: Filter off 1: Filtered operation, *SG4_IND* values available
sg_angle_offset	This flag enables optimized switching between StealthChop2 and SpreadCycle, by using the *SG4_RESULT* to determine the phase lag in StealthChop2 and compensate for the phase jump when switching from voltage-controlled to current-controlled operation in SpreadCycle. The phase offset becomes stored and is subtracted again when switching back to StealthChop2.	0 1	0: No angle correction 1: Optimized switching between StealthChop2 and SpreadCycle

Tuning StallGuard4

The StallGuard4 value *SG4_RESULT* is affected by motor-specific characteristics and application-specific demands on load, coil current, and velocity. Therefore, the easiest way to tune the StallGuard4 threshold *SG4_THRS* for a specific motor type and operating conditions is interactive tuning in the actual application.

The initial procedure for tuning StallGuard *SG4_THRS* is as follows:

1. Operate the motor at the normal operation velocity for the application and monitor *SG4_RESULT*.
2. Apply slowly increasing mechanical load to the motor. Check the lowest value of *SG4_RESULT* before the motor stalls. Use this value as starting value for *SG4_THRS* (apply half of the value).
3. Now, monitor the StallGuard output signal through DIAG output (also set *TCOOLTHRS* to match the lower velocity limit for operation) and stop the motor when a pulse is seen on the respective output. Make sure that the motor is safely stopped whenever it is stalled. Increase *SG4_THRS* if the motor stops before a stall occurs.
4. The optimum setting is reached when a stall is safely detected and leads to a pulse at DIAG in the moment where the stall occurs. *SG4_THRS* 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 (example, 80% to 120% of desired velocity) and also under extreme motor conditions (lowest and highest applicable temperature).

DIAG is pulsed by StallGuard, when *SG4_RESULT* falls below *SG4_THRS*. It is only enabled in StealthChop2 mode, and when *TCOOLTHRS* ≥ *TSTEP* > *TPWMTHRS.*

The external motion controller should react to a single pulse by stopping the motor, if desired. Set *TCOOLTHRS* to match the lower velocity threshold where StallGuard delivers a good result.

SG4_RESULT measurement has a high resolution, and there are a few ways to enhance its accuracy, as described in the following sections.

StallGuard4 Update Rate

The StallGuard4 measurement value *SG4_RESULT* is updated with each fullstep of the motor. This is enough to safely detect a stall because a stall always means the loss of four fullsteps.

StallGuard4 provides two options for measurement:

1. *sg4_filt_en* = 0: A single measurement, updated after each fullstep, and valid for each one fullstep. This measurement allows quickest reaction to load variations, as *SG4_RESULT* becomes fully updated with each zero transmission of a coil voltage. Therefore, it is optimum for stall detection with a hard obstacle.

2. *sg4_filt_en* = 1: In this mode, four individual signals are generated: *SG4_IND_0* upon falling 0-transition of the cosine wave (coil A); *SG4_IND_1* upon rising 0-transition of the cosine wave; *SG4_IND_2* upon falling 0-transition of the sine wave (coil B); *SG4_IND_3* upon rising 0-transition of the sine wave. The actual value for *SG4_RESULT* is the mean value of all four measurements, becoming updated once each fullstep. With this, each fullstep has an influence of 25% only on the overall result. This mode is perfect for detection of soft obstacles, or for use of CoolStep on imprecise motors. In filtered mode, sensitivity to a sudden load increase (hard motor blockage) is reduced.

Detecting a Motor Stall

To safely detect a motor stall, the stall threshold must be determined using a specific *SG4_THRS* setting and a specific

motor velocity or velocity range. Further, the motor current setting has a certain influence and should not be modified once optimum values are determined. Therefore, the maximum load must be determined, which the motor can drive without stalling for the given application. At the same time, monitor *SG4_RESULT* at this load. The stall threshold should be a value safely within the operating limits, to allow for parameter stray. More refined evaluation may also react to a change of *SG4_RESULT* rather than comparing to a fixed threshold. This rules out certain effects that influence the absolute value.

Limits of StallGuard4 Operation

StallGuard4 does not operate reliably at extreme motor velocities: Very low motor velocities (for many motors, less than 1Rps) generate a low back EMF and make the measurement unstable and dependent on environment conditions (temperature, etc.). Other conditions also lead to a poor response of the measurement value *SG4_RESULT* to the motor load. Very high motor velocities, in which the full sinusoidal current is not driven into the motor coils, also lead to poor response. These velocities are typically characterized by the motor back EMF exceeding the supply voltage.

CoolStep Load Adaptive Current Scaling

CoolStep is an automatic smart energy optimization for stepper motors based on the motor mechanical load, making them “green.” Depending on the actual chopper mode, CoolStep automatically uses StallGuard4 load measurement result in StealthChop2, or StallGuard2 in SpreadCycle. Coolstep requires that either StallGuard2 or StallGuard4 (depending on the chopper mode being used) be tuned prior to use. A single tuning does not cover all operating points.

Setting Up for CoolStep

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

Table 23. CoolStep Critical Parameters

PARAMETER	DESCRIPTION	RANGE	COMMENT
SEMIN	4-bit unsigned integer that sets a *lower threshold*. If *SG_RESULT* goes below this threshold (indicating a large load), 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_RESULT* 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 x* 32
SEMAX	4-bit unsigned integer that controls an *upper threshold*. If *SG_RESULT* is sampled equal to or above this threshold enough times (indicating a light load), CoolStep decreases the current to both coils. The upper threshold is (*SEMIN* + *SEMAX* + 1) x 32.	0…15	Threshold is (*SEMIN* + *SEMAX* + 1) x 32

The following figure shows the operating regions of CoolStep:

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

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

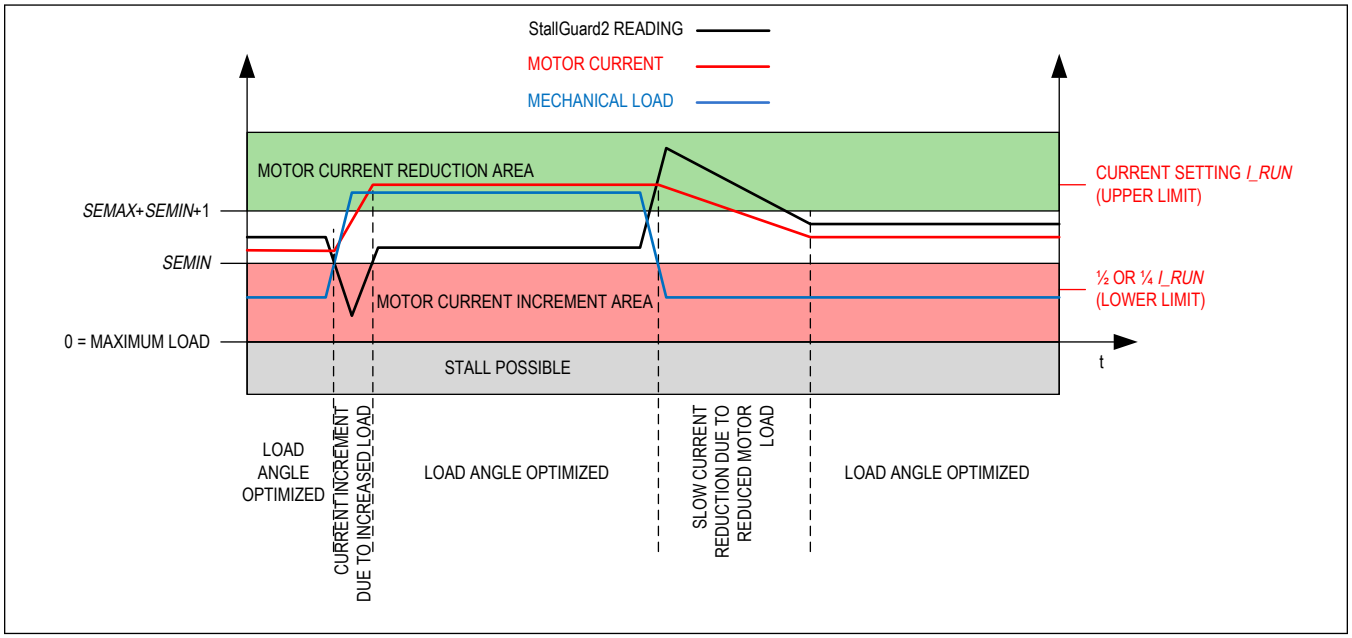


Figure 31. CoolStep Adapts Motor Current to the Load

Table 24. CoolStep Additional Parameters and Status Information

PARAMETER	DESCRIPTION	RANGE	COMMENT
SEUP	Sets the *current increment step*. The motor current becomes incremented by this setting whenever a new StallGuard2 or StallGuard4 value is measured that lies below the lower threshold, as set by *SEMIN*.	0…3	Step width of CS value CS_ACTUAL is 1, 2, 4, 8
SEDN	Sets the number of StallGuard2/StallGuard4 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. When using StealthChop2, make sure to operate well above the minimum motor current as determined for StealthChop2 current regulation, especially when a reduction down to 25% is desired.	0	0: 1/2 of IRUN (when used with StealthChop requires IRUN ≥ 16)
		1	1: 1/4 of IRUN (when used with StealthChop requires IRUN ≥ 28)
TCOOLTHRS	Lower velocity threshold for switching on CoolStep. 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 *VCOOLTHRS* identical to *VMAX*.	1… 2^{20} - 1	Specifies lower CoolStep velocity by comparing the threshold value to *TSTEP*
THIGH	Upper velocity threshold value for CoolStep. Above this velocity CoolStep becomes disabled. Adapt to the velocity range where StallGuard2/ StallGuard4 gives a stable result.	1… 2^{20} - 1	Also controls additional functions like switching to fullstepping.
STATUS WORD	**DESCRIPTION**	**RANGE**	**COMMENT**

Table 24. CoolStep Additional Parameters and Status Information (continued)

CS_ACTUAL	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

Tuning CoolStep

Before tuning CoolStep in conjunction with SpreadCycle, first tune the StallGuard2 threshold level *SGT*, which affects the range of the load measurement value *SG_RESULT*. CoolStep uses *SG_RESULT* to operate the motor near the optimum load angle of +90°. In conjunction with StealthChop2, CoolStep uses *SG4_RESULT*. In this mode, the leveling is done through *SEMIN*.

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.

Attention:

When CoolStep increases motor current, spurious detection of motor stall may occur. For best results, disable CoolStep during StallGuard2-based homing.

In case StallGuard2 is desired in combination with CoolStep, try increasing CoolStep lower threshold *SEIMIN,* as required.

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.

Advice: 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 accordingly. As acceleration and deceleration normally shall be quick, they require the full motor current, while they have only a small contribution to overall power consumption due to their short duration.

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 through *IRUN,* respectively, *IHOLD* is valid. An upper threshold is provided by the *VHIGH* setting. The velocity limits *VHIGH* and *VCOOLTHRS* are determined by the settings *THIGH* and *TCOOLTHRS.*

Both thresholds can be set as a result of the StallGuard2 and StallGuard4 tuning process.

Diagnostic Outputs

The DIAG outputs deliver a position compare signal to allow exact triggering of external logic, and an interrupt signal to trigger software to certain conditions within the motion ramp. Either an open drain (active low) output signal can be chosen (default, GCONF register, bit *diag0_int_pushpull* = 0), or an active high push-pull output signal (GCONF register, bit *diag0_int_pushpull* = 1). When using the open-drain output, multiple driver output signals can be ORed. An external pull up resistor in the range 4.7kΩ to 100kΩ is required. DIAG0 also becomes driven low upon a reset condition. However, the end of the reset condition cannot be determined by monitoring DIAG0 in this configuration, because *event_pos_reached* flag also becomes active upon reset and thus the pin stays actively low after the reset condition. To safely determine a reset condition, monitor the *reset* flag by SPI or read out any register to confirm that the chip is powered up.

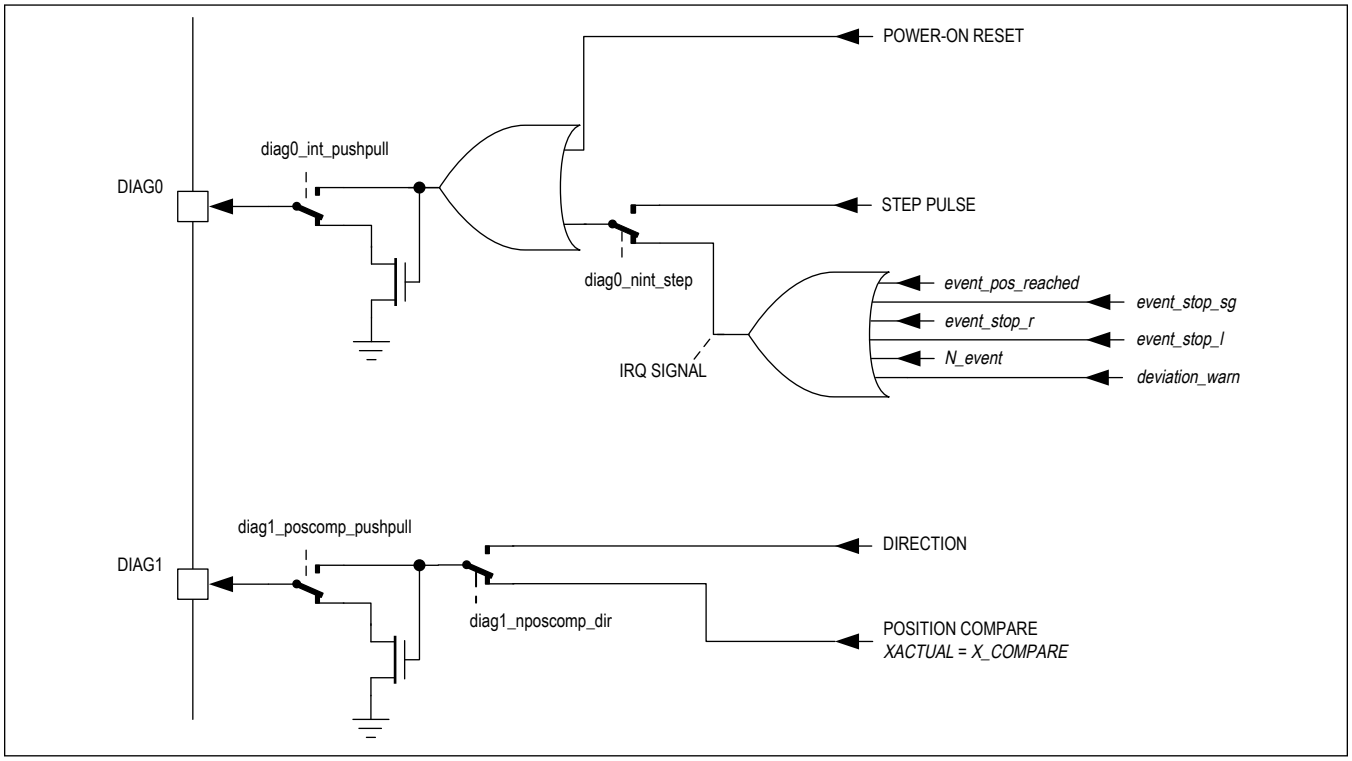


Figure 32. Diagnostic Outputs Configuration Options

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.

Designing-In DcStep

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

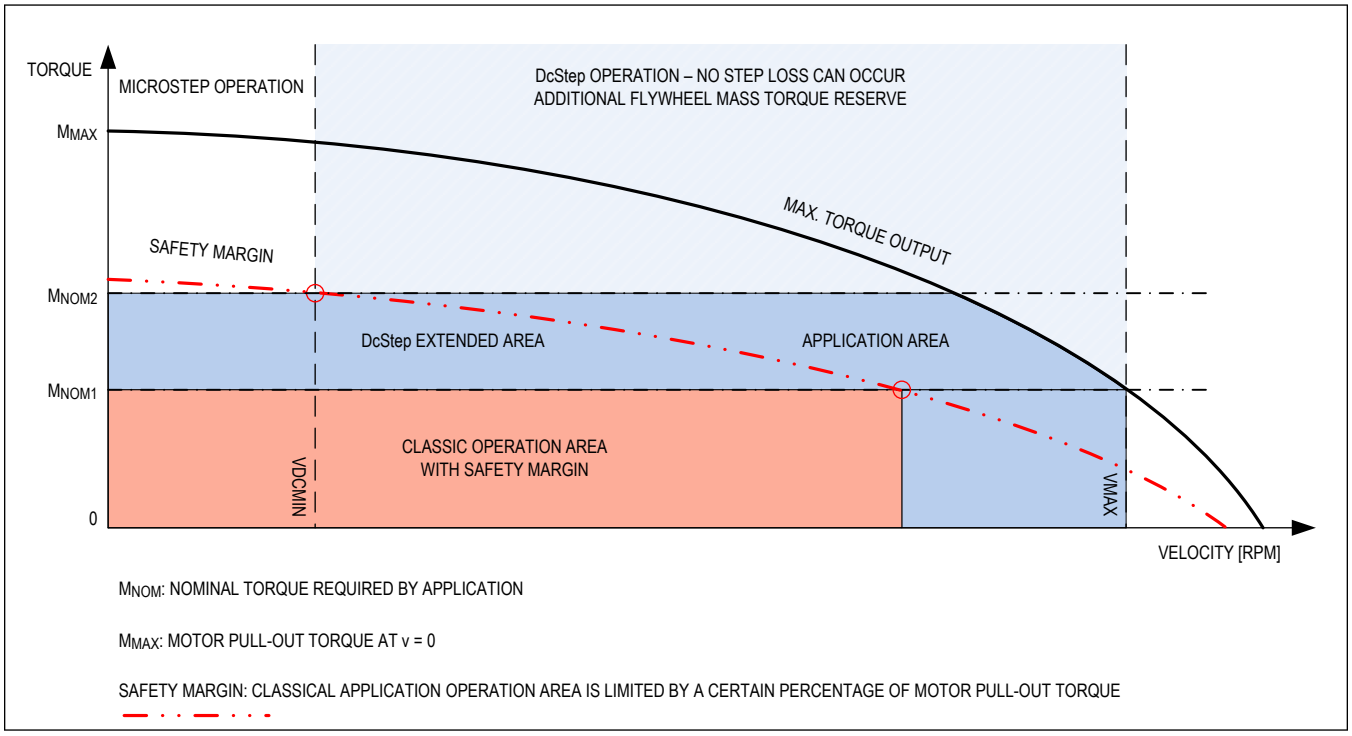


Figure 33. DcStep Extended Application Operation Area

DcStep Integration with the Motion Controller

DcStep requires only a few settings. It directly feeds back motor motion to the ramp generator, so that it becomes seamlessly integrated into the motion ramp, even if the motor becomes overloaded with respect to the target velocity. DcStep operates the motor using the constant T_{OFF} chopper in fullstep mode at the ramp generator target velocity *VMAX* 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 is braked down to a velocity below *VDCMIN*. In case the velocity falls below this value, the motor restarts once its load is released, unless the stall detection becomes enabled (set *sg_stop*). Stall detection is covered by StallGuard2 when the velocity goes below *VDCMIN*.

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 disturbs DcStep operation. Therefore, it is advised to work with the default wave. See *Sine Wave Lookup Table* for an initialization with the default table.

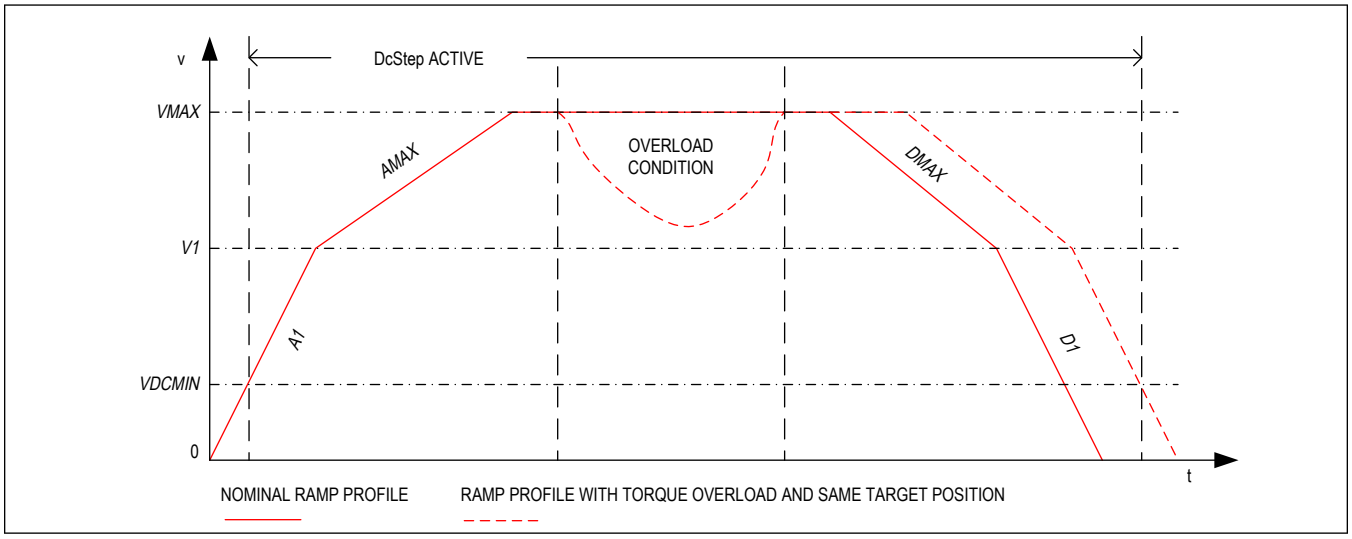


Figure 34. DcStep Velocity Profile with Overload Situation

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 lose steps. 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* is 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. It remains stopped until resetting the *event_sg_stop* flag or disabling stop-on-stall. A StallGuard2 load value also is available during DcStep operation. The range of values is limited to 0 to 255 as DcStep only sees a load angle of 0° to 90°, which is mapped to 0 to 255. In certain situations, the load angle might slip temporarily to the 90° to 180° range, which is not stable, but gives a read out of up to 511. To enable StallGuard2, also set *TCOOLTHRS* corresponding to a velocity slightly above *VDCMIN* or up to *VMAX*.

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

PARAMETER	DESCRIPTION	RANGE	COMMENT
vhighfs & *vhighchm*	These chopper configuration flags in *CHOPCONF* must be set for DcStep operation. As soon as *VDCMIN* is exceeded, the chopper becomes switched to constant T_{OFF} chopper and fullstepping .	0/1	set to 1 for DcStep
TOFF	DcStep often benefits from an increased off time value in *CHOPCONF*. Settings >2 should be preferred.	2..15	Settings 8…15 do not make any difference to setting 8 for DcStep operation.
VDCMIN	This is the lower threshold for DcStep operation when using internal ramp generator. Below this threshold, the motor operates in normal microstep mode. In DcStep operation, the motor operates at minimum *VDCMIN* even when it is completely blocked. Tune together with *DC_TIME* setting. Activation of StealthChop also disables DcStep.	0..2^{22}	0: Disable DcStep Set to the lower velocity limit for DcStep operation..

DC_TIME	This setting controls the reference pulse width for DcStep load measurement. It must be optimized for robust operation with maximum motor torque. A higher value allows higher torque and higher velocity, a lower value allows operation down to a lower velocity as set by *VDCMIN*. Check best setting under nominal operation conditions, and recheck under extreme operating conditions (example, lowest operation supply voltage, highest motor temperature, and highest supply voltage, lowest motor temperature).	0..1023	Lower limit for the setting is: t_{BLANK} (as defined by *TBL*) in clock cycles + *n* with *n* in the range 1 to 100 (for a typical motor).
DC_SG	This setting controls stall detection in DcStep mode. Increase for higher sensitivity. A stall can be used as an error condition by issuing a hard stop for the motor. Enable *sg_stop* flag for stopping the motor upon a stall event. This way, the motor is stopped once it stalls.	0..255	Set slightly higher than *DC_TIME* / 16

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:

$$\text{VACTUAL}_{\text{DCSTEP}} = \frac{\text{XACTUAL(time2)} - \text{XACTUAL(time1)}}{\text{time2} - \text{time1}} \times \frac{2^{24}}{f_{\text{CLK}}}$$

Example:

At 16.0MHz clock frequency, a 0.954s measurement delay directly yields in the velocity value, a 9.54ms delay yields in 1/100 of the actual DcStep velocity.

To get 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.

Sine Wave Lookup Table

The TMC5240 provides a programmable lookup table to store the microstep current wave. As a default, the table is preprogrammed 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. The benefits are:

- Microstepping: Extremely improved with low cost motors.
- Motor: Runs smooth and quiet.
- Torque: Reduced mechanical resonances yield improved torque.
- Low frequency motor noise: Reduced by adapting the sine and cosine wave shift for the actual motor's manufacturing tolerance.

Microstep Table

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

When modifying the wave, take care to ensure a smooth and symmetrical zero transition when the quarter wave becomes expanded to a full wave. The maximum resulting swing of the wave should be adjusted to a range of -248 to +248 to give

the best possible resolution while leaving headroom for the hysteresis-based chopper to add an offset.

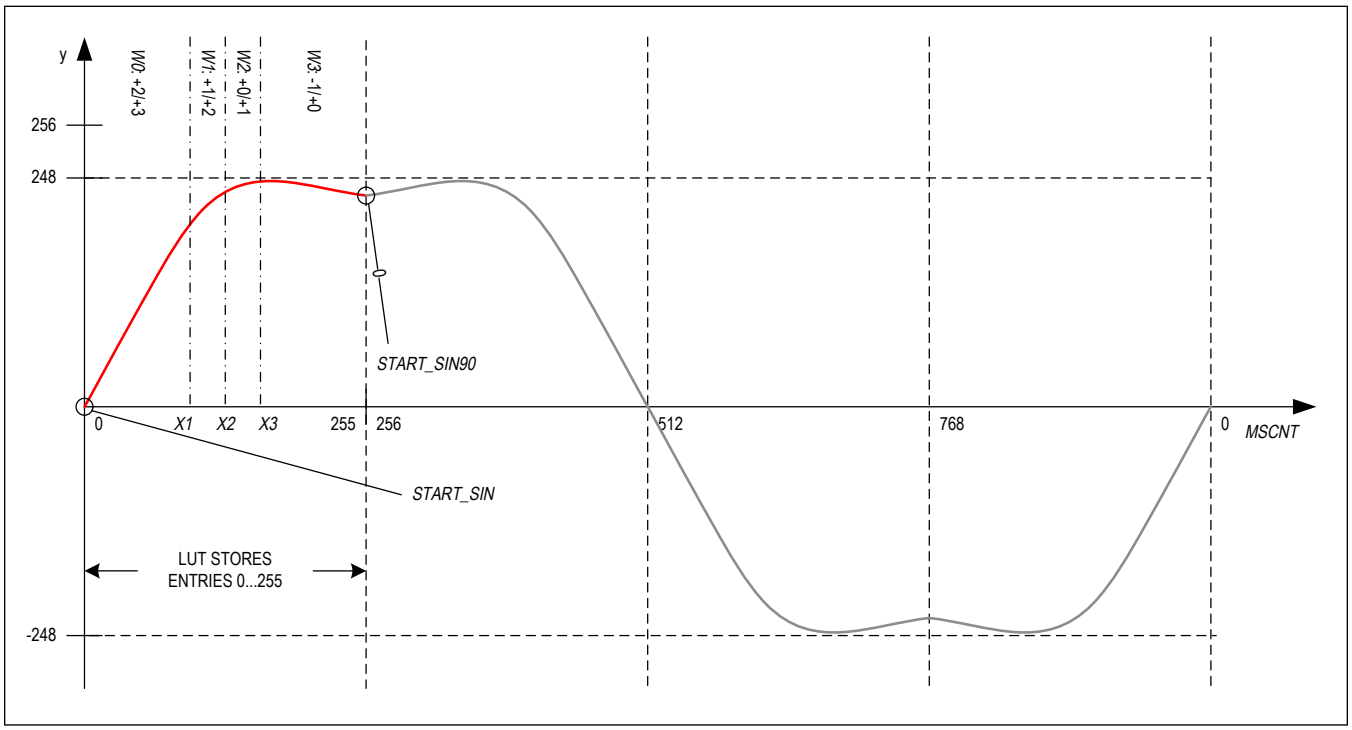


Figure 35. LUT Programming Example

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.

Matching the phase shift to the motor:

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 must be stored in *START_SIN90*. This register stores the resulting table entry for a phase shift of 90° for a two-phase motor. To adapt for motor tolerances, the phase shift can be modified from 90° (256 microsteps) to anywhere between 45° and 135°, by adding a microstep offset in the range of -127 to +127 (register *OFFSET_SIN90*). Motor tolerance requires moderate adaptations to a few 10 steps, maximum. The required correction offset can be found out using StallGuard4 individual values *SG4_IND* and trimming the offset until both coils give a symmetrical result.

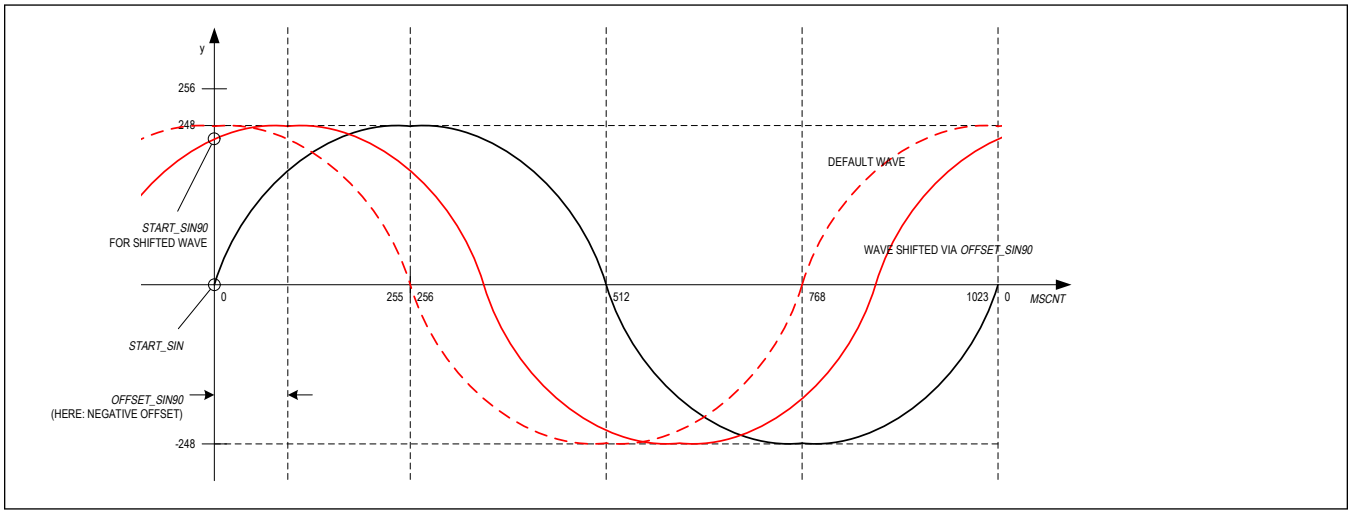


Figure 36. Shifting the Cosine Wave through OFFSET_SIN90

The default table is a good base for realizing an own table. This is an initialization example for the reset default microstep table:

MSLUT[0] = %10101010101010101011010101010100 = 0xAAAAB554
MSLUT[1] = %01001010100101010101010010101010 = 0x4A9554AA
MSLUT[2] = %00100100010010010010100100101001 = 0x24492929
MSLUT[3] = %00010000000100000100001000100010 = 0x10104222
MSLUT[4] = %11111011111111111111111111111111 = 0xFBFFFFFF
MSLUT[5] = %10110101101110110111011011101101 = 0xB5BB777D
MSLUT[6] = %01001001001010010101010101010110 = 0x49295556

MSLUT[7] = %00000000010000000100001000100010 = 0x00404222

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

MSLUTSTART = 0x00F70000:

START_SIN_0 = 0, *START_SIN90* = 247

To optimize the motor phase shift, run the motor at a medium velocity in StealthChop2 and set *sg4_filt_en* = 1. Adapt the phase offset to match the StallGuard4 results for phase A (*SG4_IND_0*+*SG4_IND_1*) to phase B (*SG4_IND_2*+*SG4_IND_3*).

If phase A value is > phase B value, increment *OFFSET_SIN90*, otherwise decrement. Repeat until best match is found.

Be sure to enter the correct value for *START_SIN90*. For an offset of -10 to +9 use *START_SIN90* = 247; up to -17 or +17 use *START_SIN90* = 246. *START_SIN* is always 0.

ABN Incremental Encoder Interface

The TMC5240 is equipped with an incremental encoder interface for ABN encoders. The encoder gives positions through digital incremental quadrature signals (usually named A and B) and an index signal (usually named N for null, Z for zero, or I for index).

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.

Checking for encoder latched event:

- Option 1: Check ENC_LATCH for change. It starts up with 0, and shows the encoder count where the N-event occurred, after starting motion for the first time. For consecutive rotations, it shows increased/decreased values and thus always changes.
- Option 2: Check for the interrupt output active and read the flag only following active interrupt output. DIAG0 pin must be configured for the interrupt lines using bit *diag0_nint_step* from GCONF register.

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 channels.

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

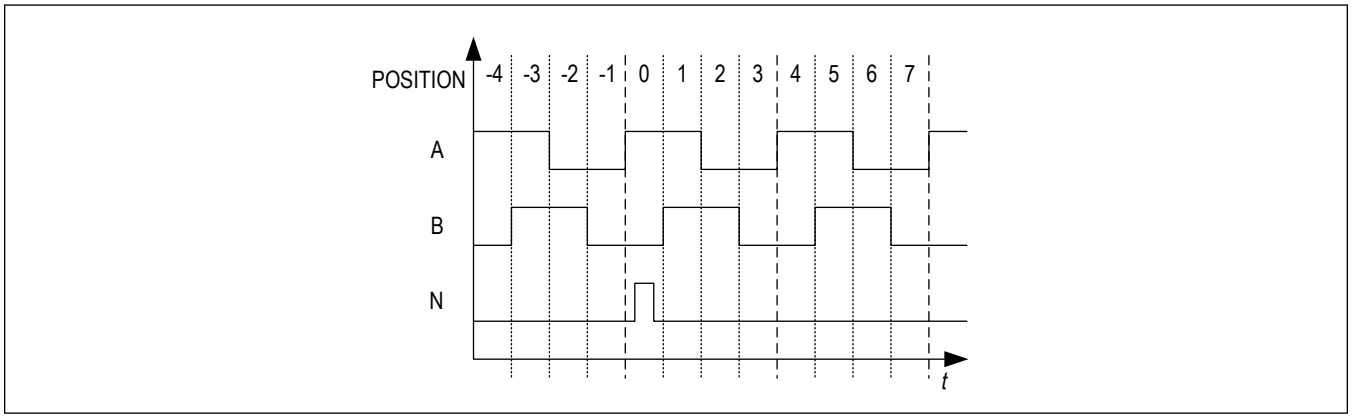


Figure 37. Outline of ABN Signals of an Incremental Encoder

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.

The Register *ENC_STATUS*

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

The Encoder Constant *ENC_CONST*

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

Examples:

- Encoder factor of 1.0: *ENC_CONST* = 0x0001.0x0000 = FACTOR.FRACTION
- Encoder factor of -1.0: *ENC_CONST* = 0xFFFF.0x0000. This is the two's complement of 0x00010000. It equals (2^{16} - (FACTOR + 1)) × (2^{16} - FRACTION)
- Decimal mode encoder factor 25.6: 00025.6000 = 0x0019.0x1770 = FACTOR.DECIMALS (DECIMALS = first 4 digits of fraction)
- Decimal mode encoder factor -25.6: (2^{16} – (25 + 1)) × (10000 - 6000) = (2^{16} - 26) × (4000) = 0xFFE6.0x0FA0
- A negative encoder constant is calculated using the following equation: (2^{16} - (FACTOR + 1)) × (10000 - DECIMALS)

Setting the Encoder to Match Motor Resolution

Encoder example settings for motor parameters:

1. USC = 256 microsteps
2. FSC = 200 fullstep motor
3. Factor = FSC x USC/encoder resolution

Table 25. 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

Reset, Disable/Stop and Power Down

Emergency Stop

The driver provides a negative active enable pin DRV_ENN to safely switch off all power MOSFETs. This allows putting the motor into freewheeling. Further, it is a safe hardware function whenever an emergency stop not coupled to software is required. Some applications may require the driver to be put into a state with active holding current or with a passive braking mode. This is possible by programming the pin ENCA to act as a step disable function. Set GCONF flag *stop_enable* to activate this option. Whenever ENCA becomes pulled high and as long as it stays high, the motor stops abruptly and goes to the power down state, as configured through *IHOLD*, *IHOLD_DELAY,* and StealthChop2 standstill options (in case StealthChop2 is in use).

External Reset and Sleep Mode

The reset and sleep mode are controlled with the SLEEPN pin.

A short pulse on SLEEPN with a duration >30µs results in a chip reset (also visible at the diagnostics outputs).

Very short pulses of <30µs are filtered out and do not have an effect on operation.

If SLEEPN is kept at GND, the IC goes into low-power standby state (sleep mode). All internal supplies are switched off.

In both cases, reset and standby, all internal register values and configurations are cleared and set to their defaults and power bridges are off.

After power-up or leaving sleep mode and reset condition, the registers must be reconfigured.

While reconfiguring the IC, it is advised to still hold the bridge drivers disabled with DRV_ENN.

Do not use during high motor velocity as energy fed back from the motor might damage the chip!

If not used, connect to V_S or V_{CC_IO} (this is a high-voltage pin).

Protections and Driver Diagnostics

The TMC5240 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* register table for details.

Besides the status flags, the TMC5240 allows measurement and read out of the chip temperature as well as feedback on the motor phase winding temperature.

For improved system reliability and overall circuit protection, the TMC5240 contains an overvoltage comparator and a trigger output OV to control external switches in terms of excessive supply voltage increase.

Overcurrent Protection

Overcurrent protection (OCP) protects the device against short circuits to the rails (supply voltage and ground) and between the outputs (OUT1A, OUT2A, OUT1B, OUT2B).

The OCP threshold depends on the selected full-scale current range, or see the *Electrical Characteristics* table for the respective threshold values.

The full-scale range is selected with the *CURRENT_RANGE* parameter in *DRV_CONF* register.

If the output current is greater than the OCP threshold for longer than the deglitch time (blanking time), then an OCP event is detected.

When an OCP event is detected, the H-bridge is immediately disabled.

The short protection tries three times before a fault flag (*s2ga*, *s2gb*, *s2vsa*, *s2vsb* in *DRV_STATUS* register) is set and the bridge becomes continuously disabled.

The device is still alive and allows configuration and status read out.

To re-enable the power bridge, DRV_ENN pin must be cycled.

Another option is to disable the power bridge with T_{OFF} = 0 in *CHOPCONF* and re-enable the bridges with T_{OFF} > *0.*

Thermal Protection and Shutdown

The TMC5240 has an internal thermal protection.

If the die temperature exceeds 165°C (typical value), a fault indication through a fault flag (*ot* in *DRV_STATUS*) is raised and the driver is three-stated until the junction temperature drops below approximately 145°C (typical value). After that, the driver is re-enabled.

In addition, the TMC5240 supports ADC-based configurable thermal prewarning levels. This can be configured in register *OTW_OV_VTH* using parameter *OVERTEMPPREWARNING_VTH*. The ADC senses the chip average temperature, while the driver stages may be at a much higher temperature. This is only to specify that TMC5240 can go in thermal shutdown and the prewarning may not be asserted, even if it is set at a low temperature.

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 can be overheated, are avoided when enabling the short to GND protection. For many applications, the overtemperature prewarning indicates 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.

Temperature Measurement

The TMC5240 offers functions to measure the internal chip temperature as well as the motor temperature.

These diagnostic functions can be helpful in applications to monitor the chip or PCB temperature and the motor temperature development over time to increase system robustness or gather additional information for predictive maintenance.

Chip Temperature Measurement

Besides the overtemperature prewarning and flags, the chip temperature itself can be determined using the *ADC_TEMP* parameter in the *ADC_TEMP* register.

The final temperature in degree Celsius can be calculated using the following formula:

$$ADC\_TEMP = 7.7 \times TEMP + 2038$$

$$TEMP[^\circ C] = \frac{ADC\_TEMP - 2038}{7.7}$$

Motor Temperature Measurement

PWM_SCALE register shows the actual duty cycle in StealthChop2 operation. For a given motor current, the duty cycle depends on the phase resistance of the motor.

As the phase resistance is temperature dependent, *PWM_SCALE* can be used to estimate the actual motor temperature and monitor changes in the motor temperature over time.

This measurement is preferably done during motor standstill or slow movements.

Typically, the motor temperature does not change quickly.

Overvoltage Protection and OV Pin

A stepper motor application can generate significant overvoltage, especially when the motor becomes quickly decelerated from a high velocity, or when the motor stalls.

This voltage becomes fed back to the supply rails by the driver output stage.

For typical NEMA17 or larger motors, and also for smaller motors with sufficient flywheel mass, the energy fed back can be substantial, so that the power capacitors and circuit consumption are not sufficient to keep the supply within its limits.

To protect the driver as well as connected circuitry, the TMC5240 has an overvoltage detection and protection mechanism.

The OV output allows attaching an NPN or MOSFET with a power resistor (brake resistor) to dump the excess energy into the resistor.

The transistor chops with approximately 3kHz to 4kHz (depending on the clock frequency) to keep the supply within the limits.

The supply voltage is permanently monitored with the internal ADC.

The upper level for the supply voltage for a given application can be configured in register *OTW_OV_VTH* using parameter *OVERVOLTAGE_VTH.*

The actual ADC value for the supply voltage can be read through register *ADC_VSUPPLY_AIN* as parameter *ADC_VSUPPLY.*

Use the following equation to convert from the ADC value to V_S and vice versa:

$$V_S = ADC\_VSUPPLY \times 9.732mV$$

In a typical application, the maximum current fed back from the motor to the supply is less than a single coil RMS coil current.

A good resistor value can be calculated as follows:

$$R_{Dump} = \frac{U_{supply}}{I_{Coil}}$$

U_{Supply} is the nominal driver supply voltage V_S. I_{Coil} is the nominal motor coil current.

Make sure the MOSFET is capable of switching the resulting current.

The OV output pin shows the actual state of the overvoltage monitor.

As soon as and as long as *ADC_VSUPPLY* becomes greater or equal to *OVERVOLTAGE_VTH,* the OV output pin changes to three-state/'Z'.

The OV output pin is an open-drain pin. The following diagram shows an example brake chopper circuit.

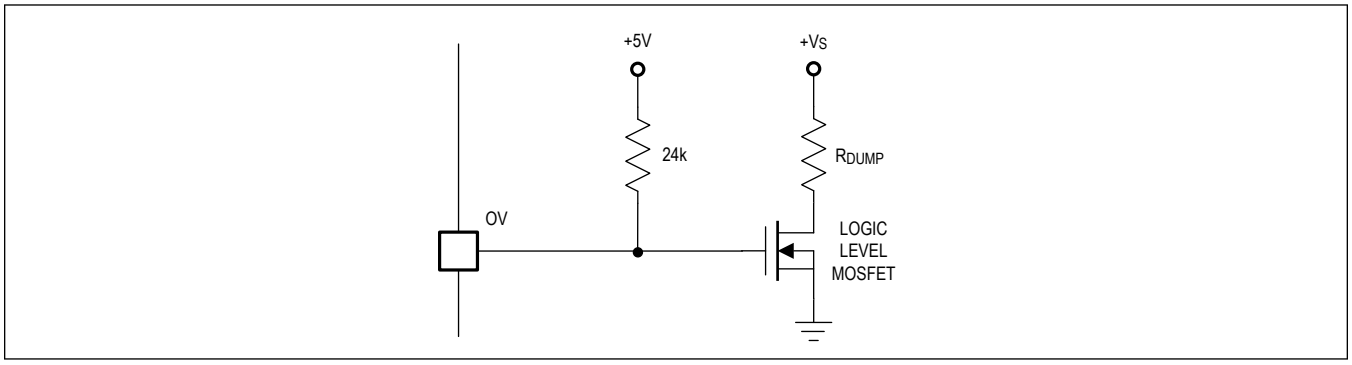


Figure 38. Brake Chopper Circuit Example

Short Protection (Short to GND and Short to VS)

The TMC5240 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, example, when touching the conducting parts connected to the system ground. The short detection is protected against spurious triggering, example, 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. To restart the motor, intervene by disabling and re-enabling the driver. Note 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.

Depending on the full-scale current setting, the low-side short protection triggers at different overcurrent protection thresholds.

Table 26. Overcurrent Protection Thresholds Based on the Full-Scale Current Setting

FULL-SCALE CURRENT SETTING (BITS)	OVERCURRENT PROTECTION THRESHOLD [A]
10 (and 11)	5.0
01	3.33
00	1.67

Open Load Diagnostics

Interrupted cables are a common cause for systems failing, example, when connectors are not firmly plugged. The TMC5240 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 trigger the open load flag. In motor standstill, open load cannot be measured as the coils might eventually have zero current.

To safely detect an interrupted coil connection, operate in SpreadCyle and check the open load flags following a motion of minimum four times the selected microstep resolution (= 4 fullsteps) into a single direction using low or nominal motor velocity operation only. However, the *ola* and *olb* flags have just informative character and do not cause any action of the driver.

Undervoltage Lockout Protection

The TMC5240 features an UVLO protection for V_S, V_{CC_IO}, and the charge pump.

UVLO condition on V_S is triggered below 4.05V (max).

UVLO condition on V_{CC_IO} is triggered below 1.95V (max).

UVLO condition on the charge pump is triggered in case of an error condition of the charge pump, example, due to a wrong capacitor value.

A V_S UVLO condition can be read from register *GSTAT* as flag *vm_uvlo.* This flag is a write-clear flag. It must be actively set to 1 to clear it.

During a V_{CC_IO} UVLO, no communication with the IC is possible and the driver is disabled. The DIAG0 pin is active low (open-drain).

ESD Protection

The chip has internal ESD protection on every pin.

The TMC5240 motor phase output pins are protected up to 8kV HBM in the application when using a bypass capacitor of at least 1uF on the positive voltage supply (V_S pins).

This is not protection against hot plugging of a motor.

External Analog Input AIN Monitoring

The TMC5240 offers an external analog input AIN, which is continuously sampled with the internal ADC.

The ADC sample value can be read out from parameter ADC_AIN in register ADC_VSUPPLY_AIN.

Use the following equation to convert from the ADC value to V_{AIN} and vice versa:

V_{AIN} = ADC_AIN × 305.2uV

The AIN input can be used to monitor external analog variables and parameters that may represent system level conditions and provide additional feedback on the system state.

Clock Oscillator and Clock Input

Using the Internal Clock

Directly tie the CLK input pin to GND close to the IC if the internal clock oscillator is to be used.

The internal clock is running at a typical frequency of 12.5MHz.

Using an External Clock

When an external clock is available, a frequency of 8MHz to 20MHz is recommended for optimum performance.

The required minimum and maximum duty cycle of the clock signal is defined in the *Electrical Characteristics* section.

Especially at clock frequencies close to 20MHz, the clock's duty cycle requirements must be satisfied.

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 as soon as an external clock is provided at the CLK pin.

Reading out bit *ext_clk* in register *IOIN* gives feedback on which clock source is currently in use (1 = external clock).

In case the external clock fails or is switched off, the internal clocks takes over seamlessly and automatically to protect the driver from damage.

Quick Configuration Guide

This guide is a practical tool for a first register configuration, and for a minimum set of measurements and decisions to tune the driver. It does not cover all advanced functionalities and options, but concentrates on the basic function set to make a motor run smoothly. Once the motor runs, explore additional features and further functionality in more detail. A current probe on one motor coil is a good aid to find the best settings.

Current Setting

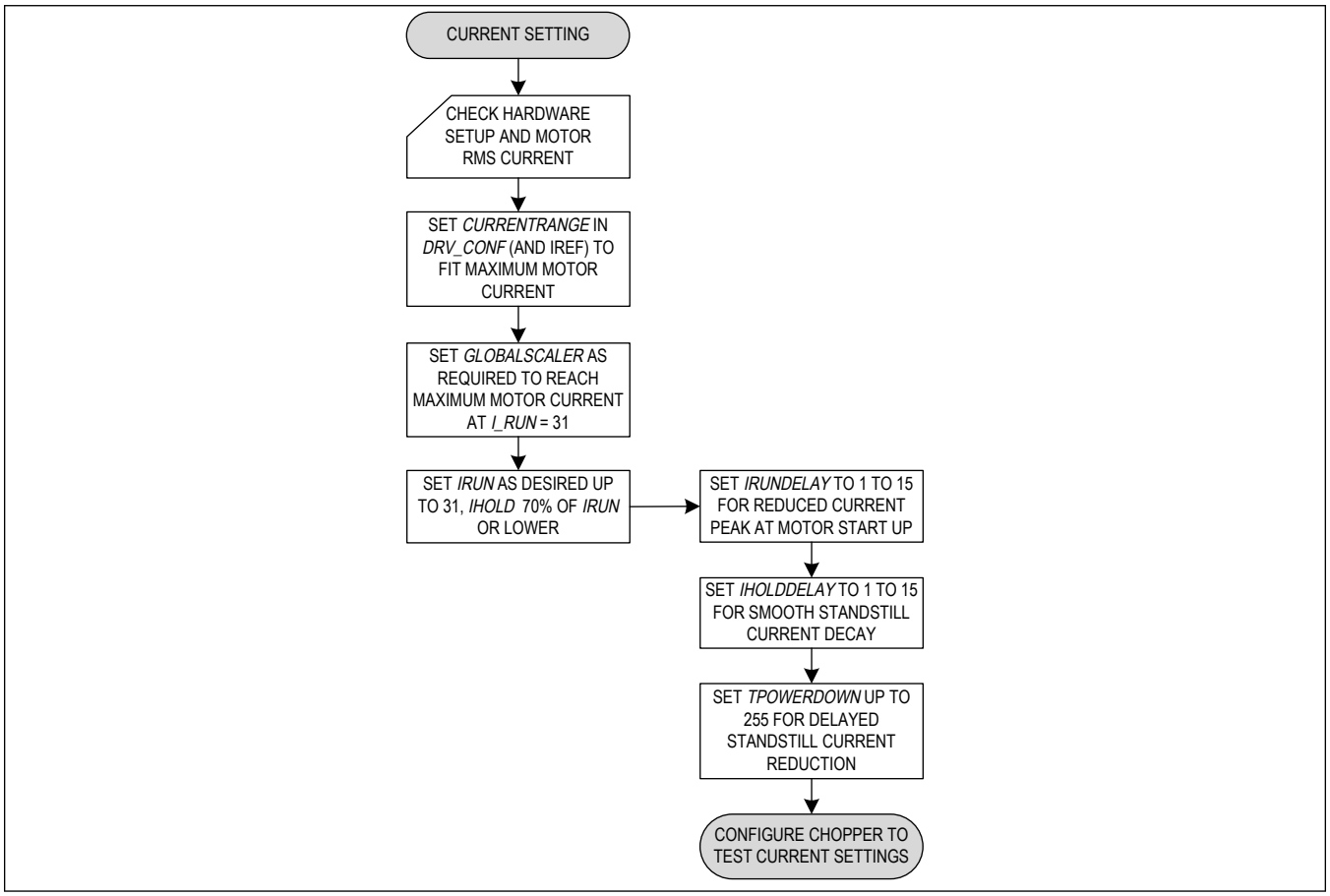


Figure 39. Quick Configuration Guide for Current Setting

StealthChop2 Configuration

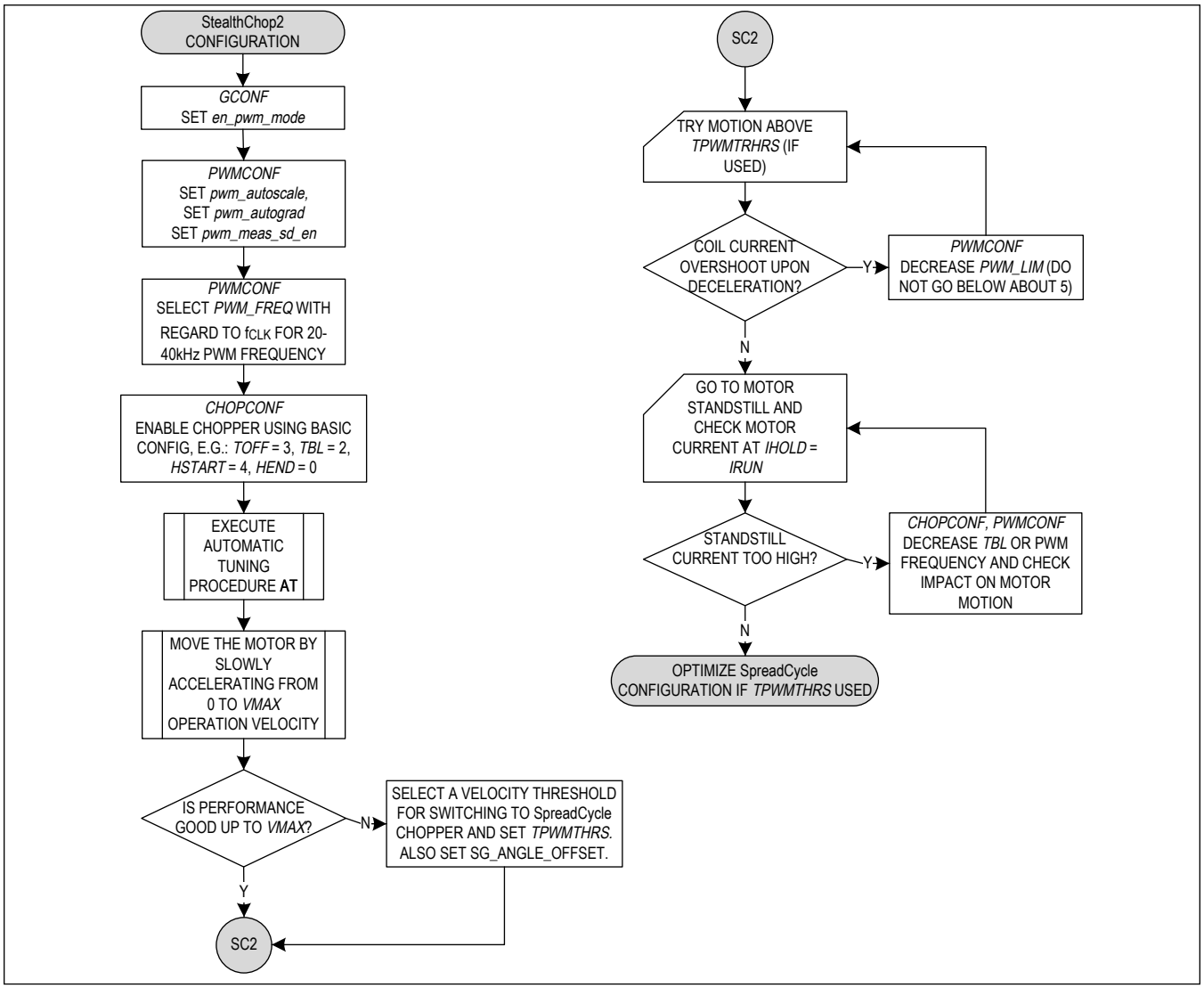


Figure 40. Quick Configuration Guide for StealthChop2 Configuration

SpreadCycle Configuration

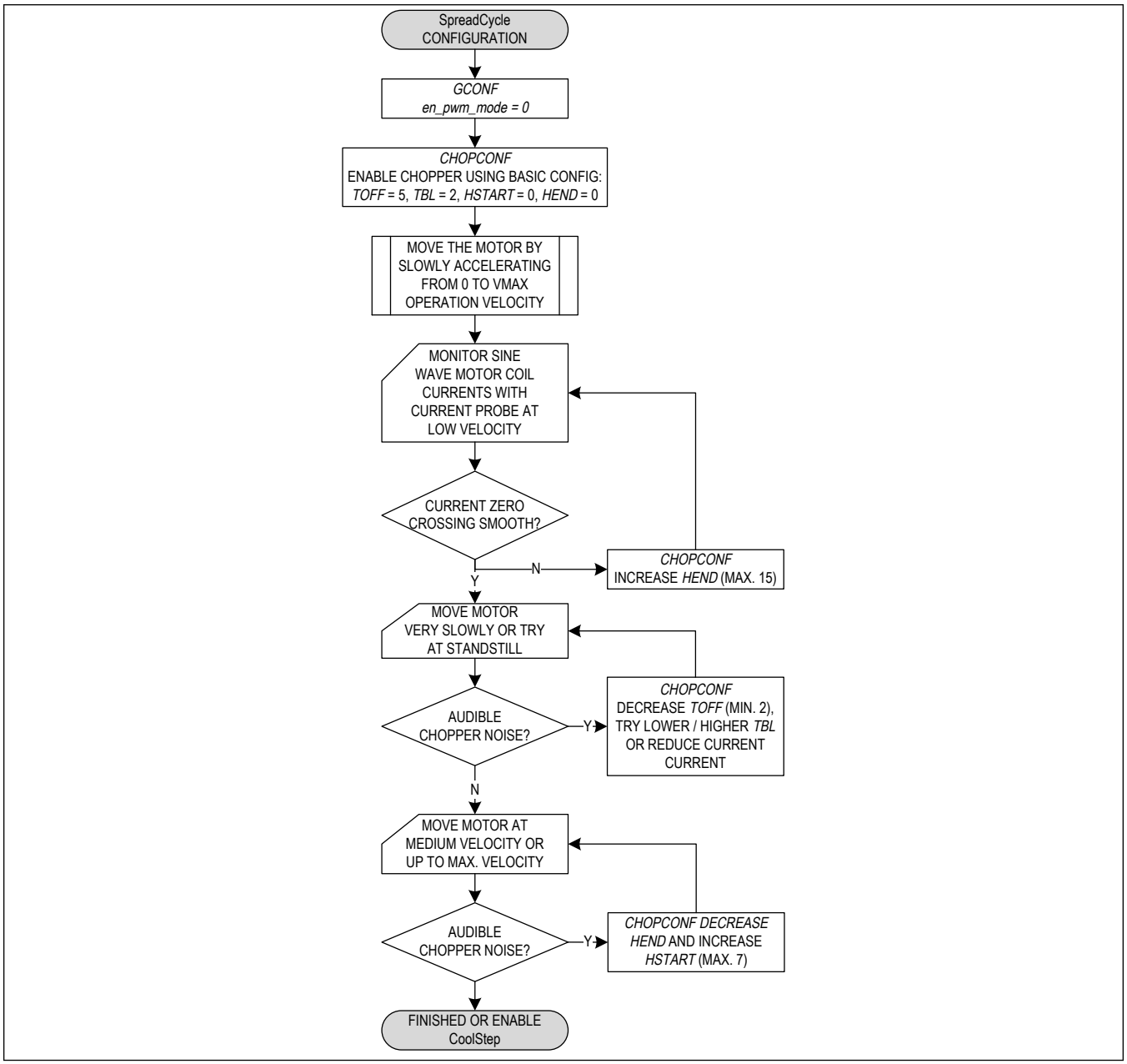


Figure 41. Quick Configuration Guide for SpreadCycle

Enabling CoolStep in Combination with StealthChop2

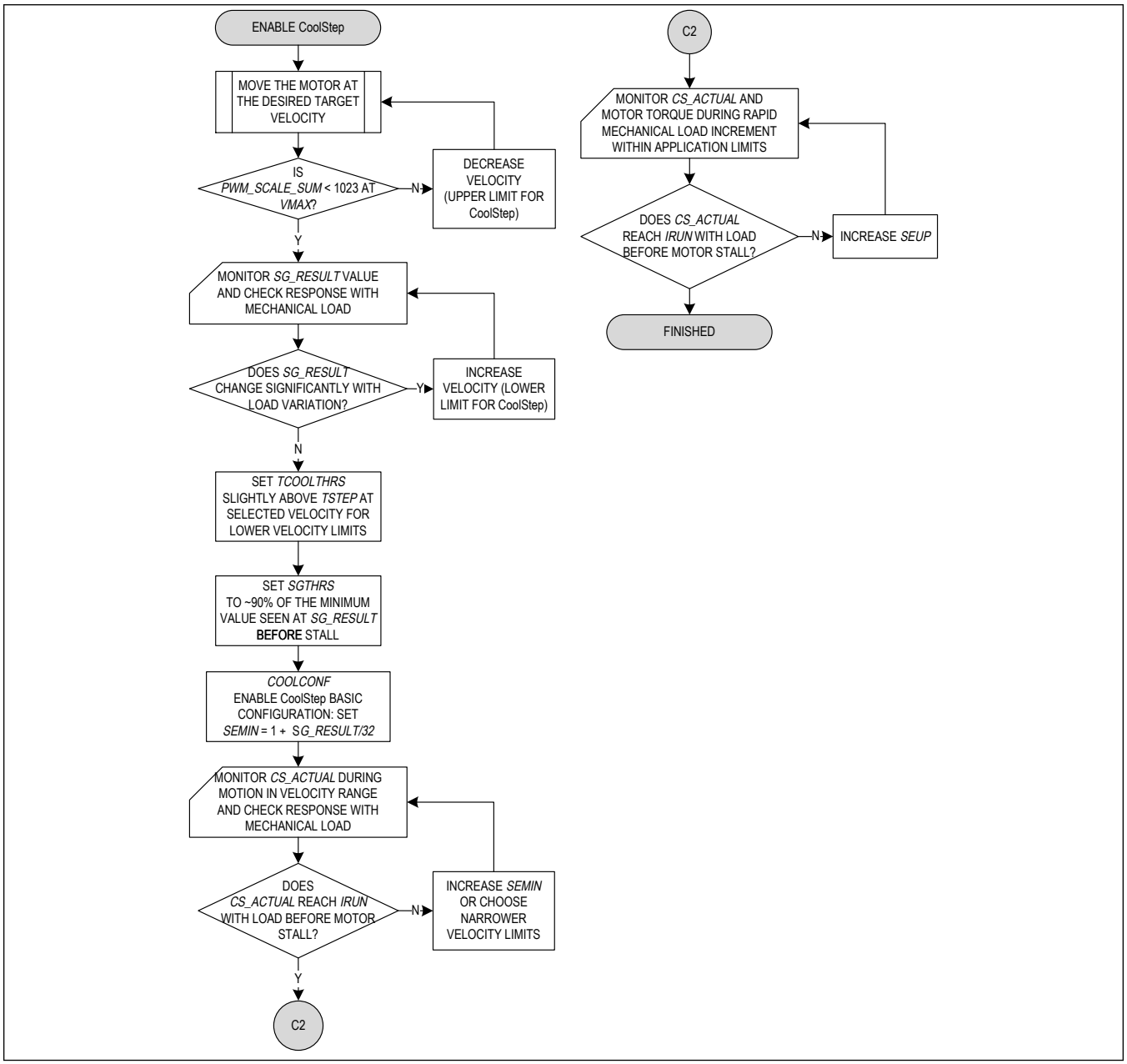


Figure 42. Quick Configuration Guide for CoolStep with StealthChop2

Enabling CoolStep in Combination with SpreadCycle

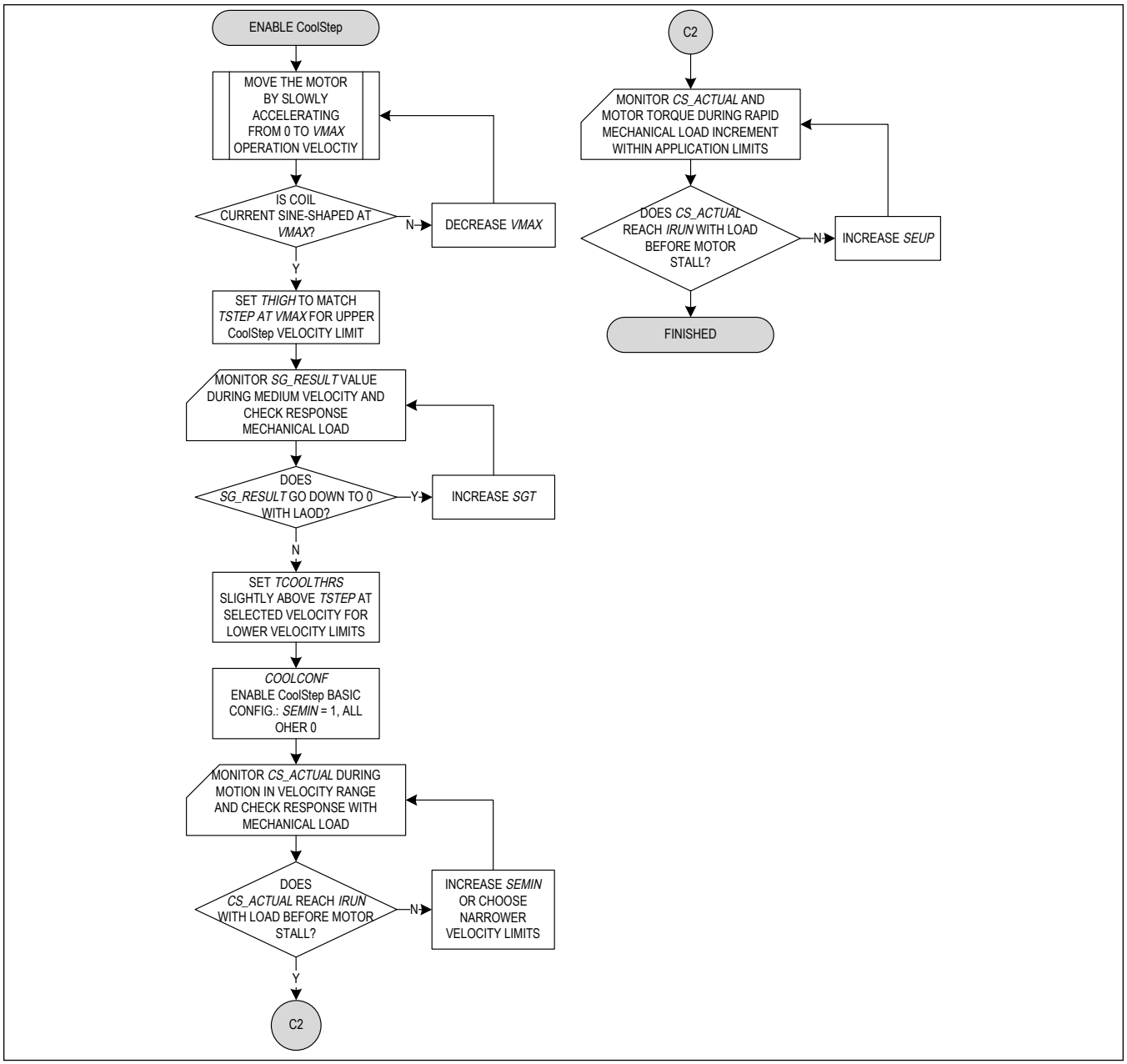


Figure 43. Quick Configuration Guide for CoolStep with SpreadCycle

Moving the Motor Using the Motion Controller

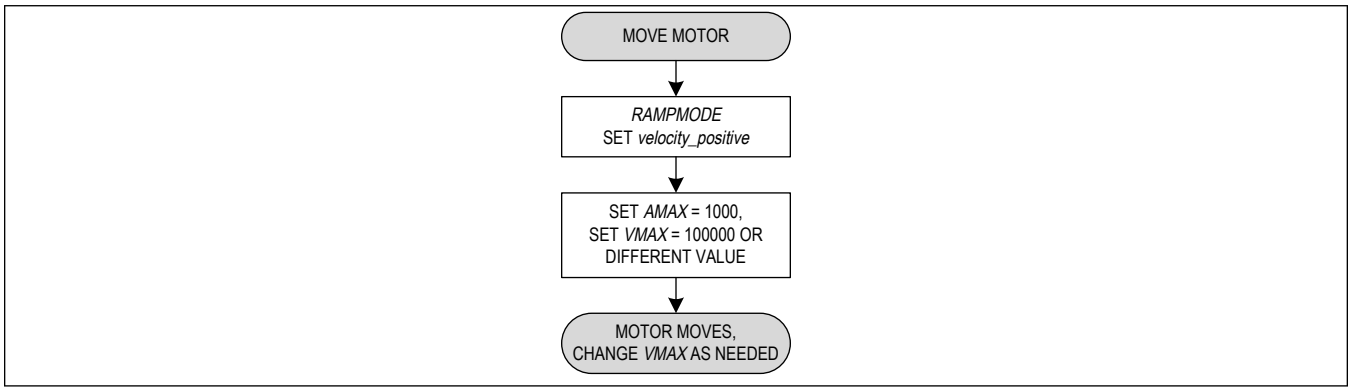


Figure 44. Quick Config Guide for Moving a Motor in Velocity Mode

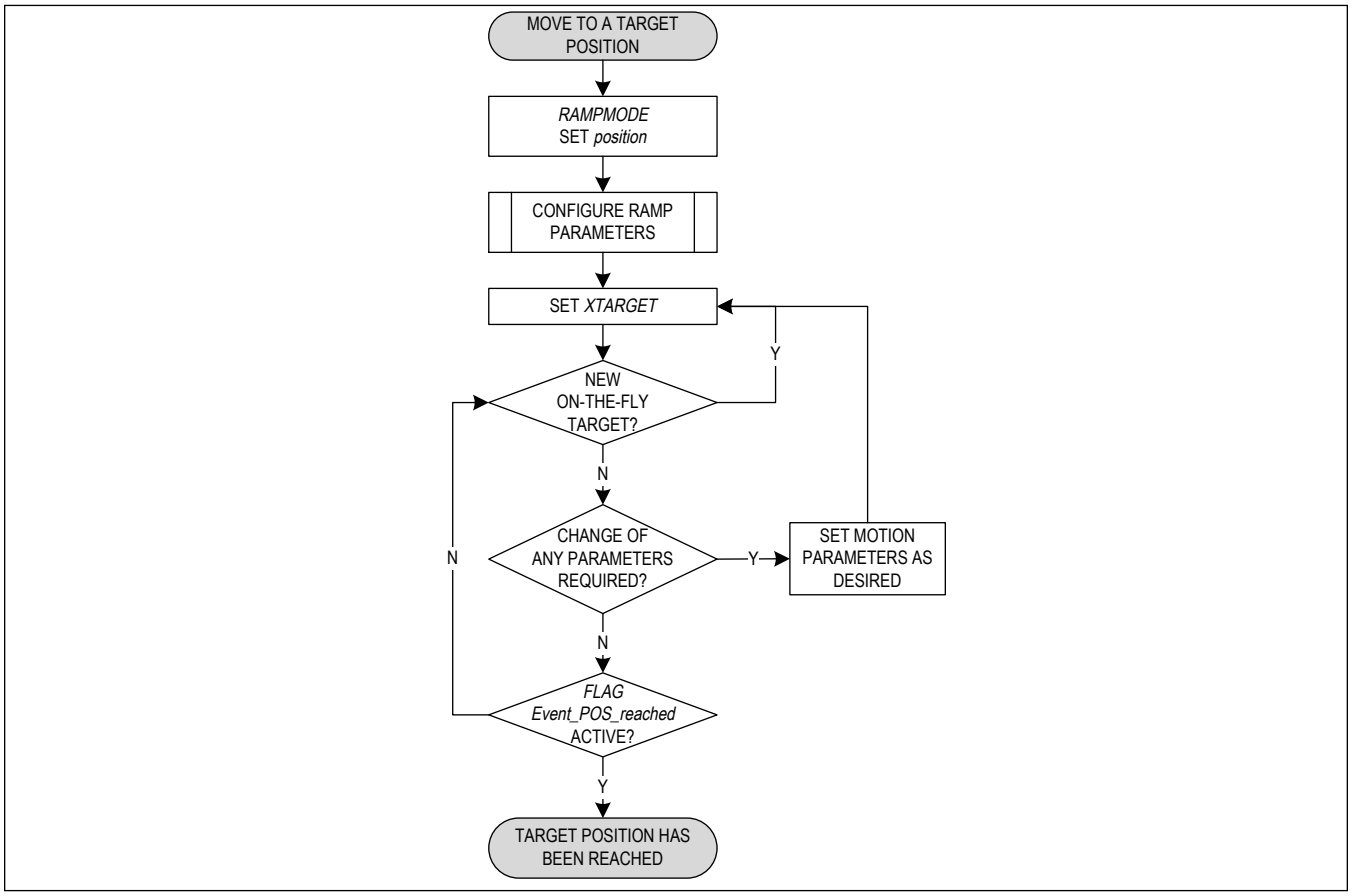


Figure 45. Quick Configuration Guide for Moving a Motor to a Target Position

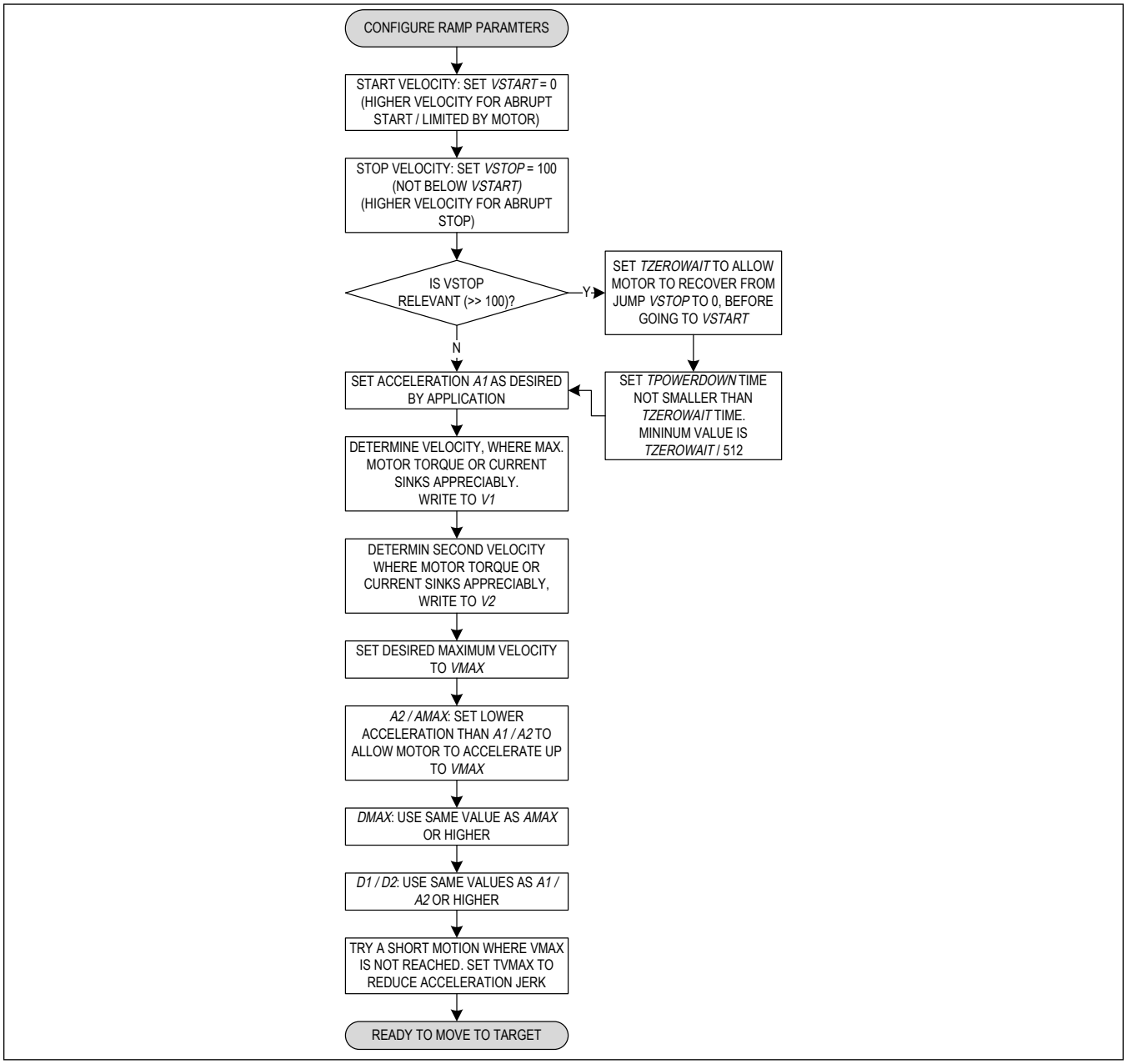


Figure 46. Quick Config Guide for Motion Ramp Parameter Setting

Enabling DcStep Operation

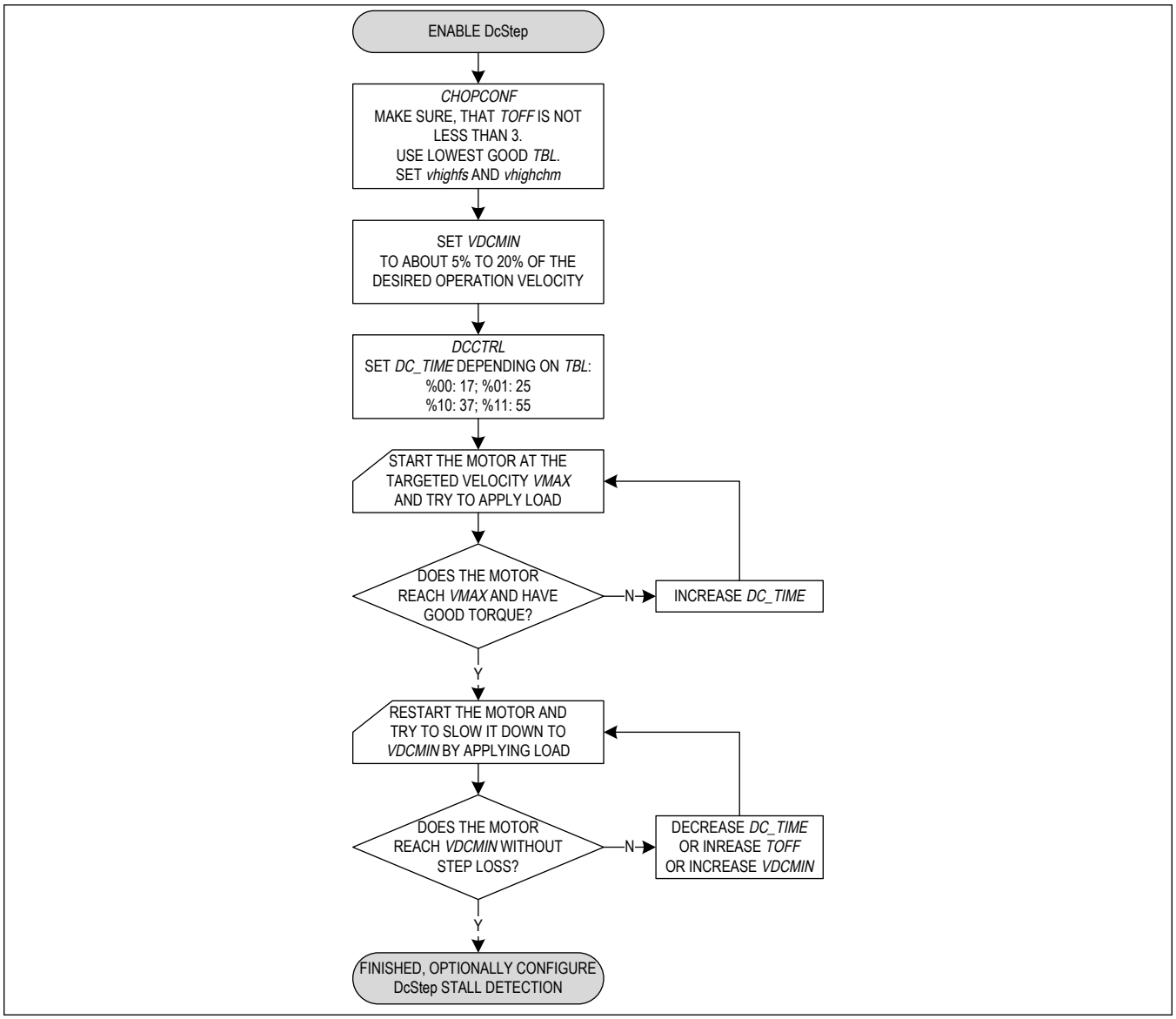


Figure 47. Quick Config Guide for DcStep

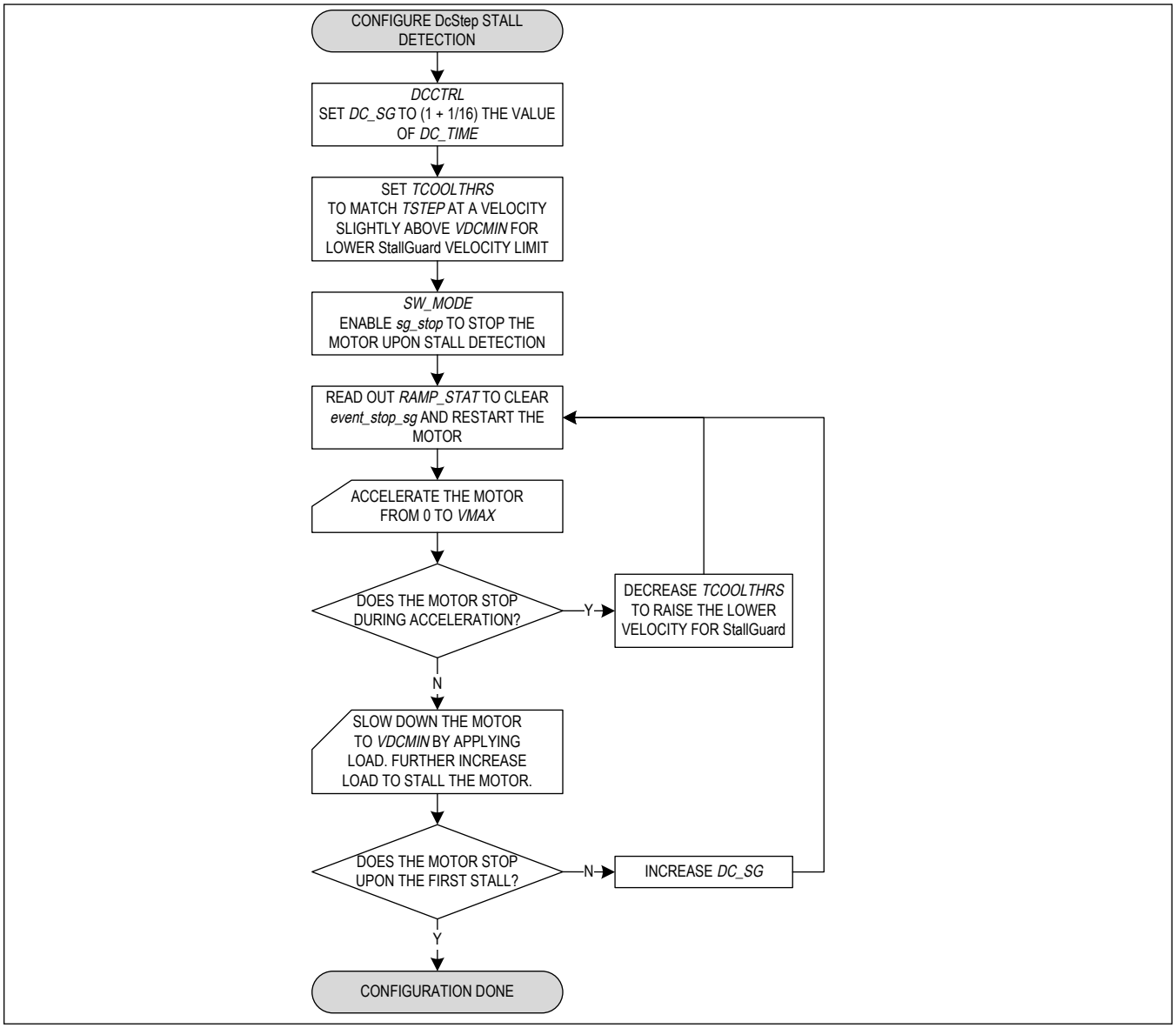


Figure 48. Quick Configuration Guide for Using Stall Detection with DcStep

General Register Mapping and Register Information

This section gives some general information on the register map.

Details on all registers and their content are given in the *Register Map* section.

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

Table 27. Overview of Register Map

REGISTERS	DESCRIPTION

Table 27. Overview of Register Map (continued)

General Configuration Registers	These registers contain: ● Global configuration ● Global status flags ● Interface configuration ● And I/O signal configuration
Ramp Generator Motion Control Register Set	This register set offers registers for: ● Choosing a ramp mode ● Choosing velocities ● Homing ● Acceleration and deceleration ● Target positioning ● Reference switch and StallGuard2 event configuration ● Ramp and reference switch status
Velocity Dependent Driver Feature Control Register Set	This register set offers registers for: ● Driver current control ● Setting thresholds for CoolStep operation ● Setting thresholds for different chopper modes ● Setting thresholds for DcStep operation
Direct Mode Registers	This register group offers registers used for the direct coil current control mode.
Encoder Register Set	The encoder register set offers all registers needed for proper ABN encoder operation.
ADC Registers	This register group offers registers to control and read the internal ADC.
Motor Driver Register Set	This register set offers registers for ● Setting/reading out microstep table and counter ● Chopper and driver configuration ● CoolStep and StallGuard configuration ● DcStep configuration ● Reading out StallGuard values and driver error flags

Register Map

TMC5240

ADDRESS	NAME	MSB							LSB
General Configuration Registers									
0x00	GCONF[31:24]	–	–	–	–	–	–	–	–
	GCONF[23:16]	–	–	–	length_step_pulse[3:0]				direct_mode
	GCONF[15:8]	stop_enable	small_hysteresis	diag1_poscomp_pushpull	diag0_int_pushpull	–	–	–	diag1_noscomp_dir
	GCONF[7:0]	diag0_int_step	–	–	shaft	–	en_pwm_mode	fast_standstill	–
0x01	GSTAT[31:24]	–	–	–	–	–	–	–	–
	GSTAT[23:16]	–	–	–	–	–	–	–	–
	GSTAT[15:8]	–	–	–	–	–	–	–	–
	GSTAT[7:0]	–	–	–	vm_uvlo	register_reset	uv_cp	drv_err	reset
0x02	IFCNT[31:24]	–	–	–	–	–	–	–	–
	IFCNT[23:16]	–	–	–	–	–	–	–	–
	IFCNT[15:8]	–	–	–	–	–	–	–	–
	IFCNT[7:0]	IFCNT[7:0]							
0x03	NODECONF[31:24]	–	–	–	–	–	–	–	–
	NODECONF[23:16]	–	–	–	–	–	–	–	–
	NODECONF[15:8]	–	–	–	–	SENDDELAY[3:0]			
	NODECONF[7:0]	NODEADDR[7:0]							
0x04	IOIN[31:24]	VERSION[7:0]							
	IOIN[23:16]	–	–	–	–	–	SILICON_RV[2:0]		
	IOIN[15:8]	ADC_ERR	EXT_CLK	EXT_RES_DET	OUTPUT	COMP_B1_B2	COMP_A1_A2	COMP_B	COMP_A
	IOIN[7:0]	reserved	UART_EN	ENCN	DRV_EN	ENCA	ENCB	REFR	REFL
0x05	X_COMPARE[31:24]	X_COMPARE[31:24]							
	X_COMPARE[23:16]	X_COMPARE[23:16]							
	X_COMPARE[15:8]	X_COMPARE[15:8]							
	X_COMPARE[7:0]	X_COMPARE[7:0]							
0x06	X_COMPARE_REPEAT[31:24]	–	–	–	–	–	–	–	–
	X_COMPARE_REPEAT[23:16]	X_COMPARE_REPEAT[23:16]							
	X_COMPARE_REPEAT[15:8]	X_COMPARE_REPEAT[15:8]							
	X_COMPARE_REPEAT[7:0]	X_COMPARE_REPEAT[7:0]							
0x0A	DRV_CONF[31:24]	–	–	–	–	–	–	–	–
	DRV_CONF[23:16]	–	–	–	–	–	–	–	–

ADDRESS	NAME	MSB							LSB
	DRV_CONF[15:8]	–	–	–	–	–	–	–	–
	DRV_CONF[7:0]	–	–	SLOPE_CONTROL[1:0]		–	–	CURRENT_RANGE[1:0]	
0x0B	GLOBAL SCALER[31:24]	–	–	–	–	–	–	–	–
	GLOBAL SCALER[23:16]	–	–	–	–	–	–	–	–
	GLOBAL SCALER[15:8]	–	–	–	–	–	–	–	–
	GLOBAL SCALER[7:0]	GLOBALSCALER[7:0]							
Velocity Dependent Configuration Registers									
0x10	IHOLD_IRUN[31:24]	–	–	–	–	IRUNDELAY[3:0]			
	IHOLD_IRUN[23:16]	–	–	–	–	IHOLDDELAY[3:0]			
	IHOLD_IRUN[15:8]	–	–	–	IRUN[4:0]				
	IHOLD_IRUN[7:0]	–	–	–	IHOLD[4:0]				
0x11	TPOWERDOWN[31:24]	–	–	–	–	–	–	–	–
	TPOWERDOWN[23:16]	–	–	–	–	–	–	–	–
	TPOWERDOWN[15:8]	–	–	–	–	–	–	–	–
	TPOWERDOWN[7:0]	TPOWERDOWN[7:0]							
0x12	TSTEP[31:24]	–	–	–	–	–	–	–	–
	TSTEP[23:16]	–	–	–	–	TSTEP[19:16]			
	TSTEP[15:8]	TSTEP[15:8]							
	TSTEP[7:0]	TSTEP[7:0]							
0x13	TPWMTHRS[31:24]	–	–	–	–	–	–	–	–
	TPWMTHRS[23:16]	–	–	–	–	TPWMTHRS[19:16]			
	TPWMTHRS[15:8]	TPWMTHRS[15:8]							
	TPWMTHRS[7:0]	TPWMTHRS[7:0]							
0x14	TCOOLTHRS[31:24]	–	–	–	–	–	–	–	–
	TCOOLTHRS[23:16]	–	–	–	–	TCOOLTHRS[19:16]			
	TCOOLTHRS[15:8]	TCOOLTHRS[15:8]							
	TCOOLTHRS[7:0]	TCOOLTHRS[7:0]							
0x15	THIGH[31:24]	–	–	–	–	–	–	–	–
	THIGH[23:16]	–	–	–	–	THIGH[19:16]			
	THIGH[15:8]	THIGH[15:8]							
	THIGH[7:0]	THIGH[7:0]							
Ramp Generator Registers									
0x20	RAMPMODE[31:24]	–	–	–	–	–	–	–	–
	RAMPMODE[23:16]	–	–	–	–	–	–	–	–
	RAMPMODE[15:8]	–	–	–	–	–	–	–	–
	RAMPMODE[7:0]	–	–	–	–	–	–	RAMPMODE[1:0]	
0x21	XACTUAL[31:24]	XACTUAL[31:24]							
	XACTUAL[23:16]	XACTUAL[23:16]							
	XACTUAL[15:8]	XACTUAL[15:8]							
	XACTUAL[7:0]	XACTUAL[7:0]							

ADDRESS	NAME	MSB							LSB
0x22	VACTUAL[31:24]	–	–	–	–	–	–	–	–
	VACTUAL[23:16]	VACTUAL[23:16]							
	VACTUAL[15:8]	VACTUAL[15:8]							
	VACTUAL[7:0]	VACTUAL[7:0]							
0x23	VSTART[31:24]	–	–	–	–	–	–	–	–
	VSTART[23:16]	–	–	–	–	–	–	VSTART[17:16]	
	VSTART[15:8]	VSTART[15:8]							
	VSTART[7:0]	VSTART[7:0]							
0x24	A1[31:24]	–	–	–	–	–	–	–	–
	A1[23:16]	–	–	–	–	–	–	A1[17:16]	
	A1[15:8]	A1[15:8]							
	A1[7:0]	A1[7:0]							
0x25	V1[31:24]	–	–	–	–	–	–	–	–
	V1[23:16]	–	–	–	–	V1[19:16]			
	V1[15:8]	V1[15:8]							
	V1[7:0]	V1[7:0]							
0x26	AMAX[31:24]	–	–	–	–	–	–	–	–
	AMAX[23:16]	–	–	–	–	–	–	AMAX[17:16]	
	AMAX[15:8]	AMAX[15:8]							
	AMAX[7:0]	AMAX[7:0]							
0x27	VMAX[31:24]	–	–	–	–	–	–	–	–
	VMAX[23:16]	–	VMAX[22:16]						
	VMAX[15:8]	VMAX[15:8]							
	VMAX[7:0]	VMAX[7:0]							
0x28	DMAX[31:24]	–	–	–	–	–	–	–	–
	DMAX[23:16]	–	–	–	–	–	–	DMAX[17:16]	
	DMAX[15:8]	DMAX[15:8]							
	DMAX[7:0]	DMAX[7:0]							
0x29	TVMAX[31:24]	–	–	–	–	–	–	–	–
	TVMAX[23:16]	–	–	–	–	–	–	–	–
	TVMAX[15:8]	TVMAX[15:8]							
	TVMAX[7:0]	TVMAX[7:0]							
0x2A	D1[31:24]	–	–	–	–	–	–	–	–
	D1[23:16]	–	–	–	–	–	–	D1[17:16]	
	D1[15:8]	D1[15:8]							
	D1[7:0]	D1[7:0]							
0x2B	VSTOP[31:24]	–	–	–	–	–	–	–	–
	VSTOP[23:16]	–	–	–	–	–	–	VSTOP[17:16]	
	VSTOP[15:8]	VSTOP[15:8]							
	VSTOP[7:0]	VSTOP[7:0]							
0x2C	TZEROWAIT[31:24]	–	–	–	–	–	–	–	–
	TZEROWAIT[23:16]	–	–	–	–	–	–	–	–

ADDRESS	NAME	MSB							LSB
	TZEROWAIT[15:8]	TZEROWAIT[15:8]							
	TZEROWAIT[7:0]	TZEROWAIT[7:0]							
0x2D	XTARGET[31:24]	XTARGET[31:24]							
	XTARGET[23:16]	XTARGET[23:16]							
	XTARGET[15:8]	XTARGET[15:8]							
	XTARGET[7:0]	XTARGET[7:0]							
0x2E	V2[31:24]	–	–	–	–	–	–	–	–
	V2[23:16]	–	–	–	–	V2[19:16]			
	V2[15:8]	V2[15:8]							
	V2[7:0]	V2[7:0]							
0x2F	A2[31:24]	–	–	–	–	–	–	–	–
	A2[23:16]	–	–	–	–	–	–	A2[17:16]	
	A2[15:8]	A2[15:8]							
	A2[7:0]	A2[7:0]							
0x30	D2[31:24]	–	–	–	–	–	–	–	–
	D2[23:16]	–	–	–	–	–	–	D2[17:16]	
	D2[15:8]	D2[15:8]							
	D2[7:0]	D2[7:0]							
Ramp Generator Driver Feature Control Registers									
0x33	VDCMIN[31:24]	–	–	–	–	–	–	–	–
	VDCMIN[23:16]	–	VDCMIN[14:8]						
	VDCMIN[15:8]	VDCMIN[7:0]							
	VDCMIN[7:0]	reserved[7:0]							
0x34	SW_MODE[31:24]	–	–	–	–	–	–	–	–
	SW_MODE[23:16]	–	–	–	–	–	–	–	–
	SW_MODE[15:8]	–	virtual_stop_enc	en_virtual_stop_r	en_virtual_stop_l	en_softstop	sg_stop	en_latch_encoder	latch_r_inactive
	SW_MODE[7:0]	latch_r_active	latch_l_inactive	latch_l_active	swap_lr	pol_stop_r	pol_stop_l	stop_r_enable	stop_l_enable
0x35	RAMP_STAT[31:24]	–	–	–	–	–	–	–	–
	RAMP_STAT[23:16]	–	–	–	–	–	–	–	–
	RAMP_STAT[15:8]	status_virtual_stop_r	status_virtual_stop_l	status_sg	second_move	t_zerowait_active	vzero	position_reached	velocity_reached
	RAMP_STAT[7:0]	event_pos_reached	event_stop_sg	event_stop_r	event_stop_l	status_latch_r	status_latch_l	status_stop_r	status_stop_l
0x36	XLATCH[31:24]	XLATCH[31:24]							
	XLATCH[23:16]	XLATCH[23:16]							
	XLATCH[15:8]	XLATCH[15:8]							
	XLATCH[7:0]	XLATCH[7:0]							
Encoder Registers									
0x38	ENCMODE[31:24]	–	–	–	–	–	–	–	–
	ENCMODE[23:16]	–	–	–	–	–	–	–	–

ADDRESS	NAME	MSB							LSB
	ENCMODE[15:8]	–	–	–	–	–	enc_sel_decimal	latch_x_act	clr_enc_x
	ENCMODE[7:0]	pos_neg_edge[1:0]		clr_once	clr_cont	ignore_AB	pol_N	pol_B	pol_A
0x39	X_ENC[31:24]	X_ENC[31:24]							
	X_ENC[23:16]	X_ENC[23:16]							
	X_ENC[15:8]	X_ENC[15:8]							
	X_ENC[7:0]	X_ENC[7:0]							
0x3A	ENC_CONST[31:24]	ENC_CONST[31:24]							
	ENC_CONST[23:16]	ENC_CONST[23:16]							
	ENC_CONST[15:8]	ENC_CONST[15:8]							
	ENC_CONST[7:0]	ENC_CONST[7:0]							
0x3B	ENC_STATUS[31:24]	–	–	–	–	–	–	–	–
	ENC_STATUS[23:16]	–	–	–	–	–	–	–	–
	ENC_STATUS[15:8]	–	–	–	–	–	–	–	–
	ENC_STATUS[7:0]	–	–	–	–	–	–	deviation_warn	n_event
0x3C	ENC_LATCH[31:24]	ENC_LATCH[31:24]							
	ENC_LATCH[23:16]	ENC_LATCH[23:16]							
	ENC_LATCH[15:8]	ENC_LATCH[15:8]							
	ENC_LATCH[7:0]	ENC_LATCH[7:0]							
0x3D	ENC_DEVIATION[31:24]	–	–	–	–	–	–	–	–
	ENC_DEVIATION[23:16]	–	–	–	–	ENC_DEVIATION[19:16]			
	ENC_DEVIATION[15:8]	ENC_DEVIATION[15:8]							
	ENC_DEVIATION[7:0]	ENC_DEVIATION[7:0]							
0x3E	VIRTUAL_STOP_L[31:24]	VIRTUAL_STOP_L[31:24]							
	VIRTUAL_STOP_L[23:16]	VIRTUAL_STOP_L[23:16]							
	VIRTUAL_STOP_L[15:8]	VIRTUAL_STOP_L[15:8]							
	VIRTUAL_STOP_L[7:0]	VIRTUAL_STOP_L[7:0]							
0x3F	VIRTUAL_STOP_R[31:24]	VIRTUAL_STOP_R[31:24]							
	VIRTUAL_STOP_R[23:16]	VIRTUAL_STOP_R[23:16]							
	VIRTUAL_STOP_R[15:8]	VIRTUAL_STOP_R[15:8]							
	VIRTUAL_STOP_R[7:0]	VIRTUAL_STOP_R[7:0]							
ADC Registers									
0x50	ADC_VSUPPLY_AIN[31:24]	–	–	–	ADC_AIN[12:8]				
	ADC_VSUPPLY_AIN[23:16]	ADC_AIN[7:0]							

ADDRESS	NAME	MSB							LSB
	ADC_VSUPPLY_AIN[15:8]	–	–	–	ADC_VSUPPLY[12:8]				
	ADC_VSUPPLY_AIN[7:0]	ADC_VSUPPLY[7:0]							
0x51	ADC_TEMP[31:24]	–	–	–	RESERVED[12:8]				
	ADC_TEMP[23:16]	RESERVED[7:0]							
	ADC_TEMP[15:8]	–	–	–	ADC_TEMP[12:8]				
	ADC_TEMP[7:0]	ADC_TEMP[7:0]							
0x52	OTW_OV_VTH[31:24]	–	–	–	OVERTEMPPREWARNING_VTH[12:8]				
	OTW_OV_VTH[23:16]	OVERTEMPPREWARNING_VTH[7:0]							
	OTW_OV_VTH[15:8]	–	–	–	OVERVOLTAGE_VTH[12:8]				
	OTW_OV_VTH[7:0]	OVERVOLTAGE_VTH[7:0]							
Motor Driver Registers									
0x60	MSLUT_0[31:24]	MSLUT_0[31:24]							
	MSLUT_0[23:16]	MSLUT_0[23:16]							
	MSLUT_0[15:8]	MSLUT_0[15:8]							
	MSLUT_0[7:0]	MSLUT_0[7:0]							
0x61	MSLUT_1[31:24]	MSLUT_1[31:24]							
	MSLUT_1[23:16]	MSLUT_1[23:16]							
	MSLUT_1[15:8]	MSLUT_1[15:8]							
	MSLUT_1[7:0]	MSLUT_1[7:0]							
0x62	MSLUT_2[31:24]	MSLUT_2[31:24]							
	MSLUT_2[23:16]	MSLUT_2[23:16]							
	MSLUT_2[15:8]	MSLUT_2[15:8]							
	MSLUT_2[7:0]	MSLUT_2[7:0]							
0x63	MSLUT_3[31:24]	MSLUT_3[31:24]							
	MSLUT_3[23:16]	MSLUT_3[23:16]							
	MSLUT_3[15:8]	MSLUT_3[15:8]							
	MSLUT_3[7:0]	MSLUT_3[7:0]							
0x64	MSLUT_4[31:24]	MSLUT_4[31:24]							
	MSLUT_4[23:16]	MSLUT_4[23:16]							
	MSLUT_4[15:8]	MSLUT_4[15:8]							
	MSLUT_4[7:0]	MSLUT_4[7:0]							
0x65	MSLUT_5[31:24]	MSLUT_5[31:24]							
	MSLUT_5[23:16]	MSLUT_5[23:16]							
	MSLUT_5[15:8]	MSLUT_5[15:8]							
	MSLUT_5[7:0]	MSLUT_5[7:0]							
0x66	MSLUT_6[31:24]	MSLUT_6[31:24]							
	MSLUT_6[23:16]	MSLUT_6[23:16]							
	MSLUT_6[15:8]	MSLUT_6[15:8]							
	MSLUT_6[7:0]	MSLUT_6[7:0]							
0x67	MSLUT_7[31:24]	MSLUT_7[31:24]							
	MSLUT_7[23:16]	MSLUT_7[23:16]							

ADDRESS	NAME	MSB							LSB
	MSLUT_7[15:8]	MSLUT_7[15:8]							
	MSLUT_7[7:0]	MSLUT_7[7:0]							
0x68	MSLUTSEL[31:24]	X3[7:0]							
	MSLUTSEL[23:16]	X2[7:0]							
	MSLUTSEL[15:8]	X1[7:0]							
	MSLUTSEL[7:0]	W3[1:0]		W2[1:0]		W1[1:0]		W0[1:0]	
0x69	MSLUTSTART[31:24]	OFFSET_SIN90[7:0]							
	MSLUTSTART[23:16]	START_SIN90[7:0]							
	MSLUTSTART[15:8]	–	–	–	–	–	–	–	–
	MSLUTSTART[7:0]	START_SIN[7:0]							
0x6A	MSCNT[31:24]	–	–	–	–	–	–	–	–
	MSCNT[23:16]	–	–	–	–	–	–	–	–
	MSCNT[15:8]	–	–	–	–	–	–	MSCNT[9:8]	
	MSCNT[7:0]	MSCNT[7:0]							
0x6B	MSCURACT[31:24]	–	–	–	–	–	–	–	CUR_A[8]
	MSCURACT[23:16]	CUR_A[7:0]							
	MSCURACT[15:8]	–	–	–	–	–	–	–	CUR_B[8]
	MSCURACT[7:0]	CUR_B[7:0]							
0x6C	CHOPCONF[31:24]	diss2vs	diss2g	reserved	intpol	MRES[3:0]			
	CHOPCONF[23:16]	TPFD[3:0]				vhighchm	vhighfs	–	TBL[1]
	CHOPCONF[15:8]	TBL[0]	chm	–	disfdcc	fd3	HEND_OFFSET[3:1]		
	CHOPCONF[7:0]	HEND_OFFSET[0]	HSTRT_TFD210[2:0]			TOFF[3:0]			
0x6D	COOLCONF[31:24]	–	–	–	–	–	–	–	sfilt
	COOLCONF[23:16]	–	sgt[6:0]						
	COOLCONF[15:8]	seimin	sedn[1:0]		–	semax[3:0]			
	COOLCONF[7:0]	–	seup[1:0]		–	semin[3:0]			
0x6E	DCCTRL[31:24]	–	–	–	–	–	–	–	–
	DCCTRL[23:16]	DC_SG[7:0]							
	DCCTRL[15:8]	–	–	–	–	–	–	DC_TIME[9:8]	
	DCCTRL[7:0]	DC_TIME[7:0]							
0x6F	DRV_STATUS[31:24]	stst	olb	ola	s2gb	s2ga	otpw	ot	stallguard
	DRV_STATUS[23:16]	–	–	–	CS_ACTUAL[4:0]				
	DRV_STATUS[15:8]	fsactive	stealth	s2vsb	s2vsa	–	–	SG_RESULT[9:8]	
	DRV_STATUS[7:0]	SG_RESULT[7:0]							
0x70	PWMCONF[31:24]	PWM_LIM[3:0]				PWM_REG[3:0]			
	PWMCONF[23:16]	pwm_dis_reg_stst	pwm_meas_sd_enable	FREEWHEEL[1:0]		pwm_autograd	pwm_autoscale	PWM_FREQ[1:0]	
	PWMCONF[15:8]	PWM_GRAD[7:0]							

ADDRESS	NAME	MSB							LSB
	PWMCONF[7:0]	PWM_OFS[7:0]							
0x71	PWM_SCALE[31:24]	–	–	–	–	–	–	–	PWM_SCALE_AUTO[8]
	PWM_SCALE[23:16]	PWM_SCALE_AUTO[7:0]							
	PWM_SCALE[15:8]	–	–	–	–	–	–	PWM_SCALE_SUM[9:8]	
	PWM_SCALE[7:0]	PWM_SCALE_SUM[7:0]							
0x72	PWM_AUTO[31:24]	–	–	–	–	–	–	–	–
	PWM_AUTO[23:16]	PWM_GRAD_AUTO[7:0]							
	PWM_AUTO[15:8]	–	–	–	–	–	–	–	–
	PWM_AUTO[7:0]	PWM_OFS_AUTO[7:0]							
0x74	SG4_THRS[31:24]	–	–	–	–	–	–	–	–
	SG4_THRS[23:16]	–	–	–	–	–	–	–	–
	SG4_THRS[15:8]	–	–	–	–	–	–	sg_angle_offset	sg4_filt_en
	SG4_THRS[7:0]	SG4_THRS[7:0]							
0x75	SG4_RESULT[31:24]	–	–	–	–	–	–	–	–
	SG4_RESULT[23:16]	–	–	–	–	–	–	–	–
	SG4_RESULT[15:8]	–	–	–	–	–	–	SG4_RESULT[9:8]	
	SG4_RESULT[7:0]	SG4_RESULT[7:0]							
0x76	SG4_IND[31:24]	SG4_IND_3[7:0]							
	SG4_IND[23:16]	SG4_IND_2[7:0]							
	SG4_IND[15:8]	SG4_IND_1[7:0]							
	SG4_IND[7:0]	SG4_IND_0[7:0]							

Register Details

GCONF (0x0)

Global Configuration Flags

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	23	22	21	20	19	18	17	16
Field	–	–	–	length_step_pulse[3:0]				direct_mode
Reset	–	–	–	0x0				0x0
Access Type	–	–	–	Write, Read				Write, Read

BIT	15	14	13	12	11	10	9	8
Field	stop_enable	small_hysteresis	diag1_poscomp_pushpull	diag0_int_pushpull	–	–	–	diag1_nposcomp_dir
Reset	0x0	0x0	0x0	0x0	–	–	–	0x0
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	–	–	–	Write, Read

BIT	7	6	5	4	3	2	1	0
Field	diag0_nint_step	–	–	shaft	–	en_pwm_mode	fast_standstill	–
Reset	0x0	–	–	0x0	–	0x0	0x0	–
Access Type	Write, Read	–	–	Write, Read	–	Write, Read	Write, Read	–

BITFIELD	BITS	DESCRIPTION	DECODE
length_step_pulse	20:17	cDriver only: length_step_pulse = 0: STEP output toggles upon each step; length_step_pulse = 1...15: STEP pin high time in number of clock cycles	
direct_mode	16	Enable direct motor phase current control through serial interface.	0x0: Normal operation 0x1: Motor coil currents and polarity directly programmed through serial interface: Register *XTARGET* (0x2D) specifies signed coil A current (bits 8..0) and coil B current (bits 24..16). In this mode, the current is scaled by *IHOLD* setting. Velocity based current regulation of StealthChop2 is not available in this mode. The automatic StealthChop2 current regulation works only for low stepper motor velocities.
stop_enable	15	Motor hard stop function enable.	0x0: Normal operation 0x1: Emergency stop: ENCA stops the sequencer when tied high (no steps become executed by the sequencer, motor goes to standstill state).
small_hysteresis	14		0x0: Hysteresis for step frequency comparison is 1/16 0x1: Hysteresis for step frequency comparison is 1/32
diag1_poscomp_pushpull	13	DIAG1 output type configuration.	0x0: DIAG1 is open collector output (active low) 0x1: Enable DIAG1 push pull output (active high)
diag0_int_pushpull	12	DIAG0 output type configuration.	0x0: DIAG0_SW is open collector output (active low) 0x1: Enable DIAG0_SW push pull output (active high)
diag1_nposcomp_dir	8	DIAG1 output configuration, when not using UART.	0x0: DIAG1 outputs position compare signal 0x1: Enable DIAG1 as DIR output for external STEP/DIR driver
diag0_nint_step	7	DIAG0 output configuration.	0x0: DIAG0 outputs interrupt signal 0x1: Enable DIAG0 as STEP output (half frequency, dual edge triggered or frequency when length_step_pulse != 0) for external STEP/DIR driver
shaft	4	Change motor direction / direction sign	0x0: Default motor direction 0x1: Inverse motor direction

BITFIELD	BITS	DESCRIPTION	DECODE
en_pwm_mode	2	Enable the StealthChop2 mode	0x0: no StealthChop2 0x1: StealthChop2 voltage PWM mode enabled (depending on velocity thresholds). Switch from off to on state while in standstill and at IHOLD = nominal IRUN current, only.
fast_standstill	1	Timeout for step execution until standstill detection	0x0: Normal time: 2^20 clocks 0x1: Short time: 2^18 clocks

GSTAT (0x1)

Global Status Flags

(Re-Write with '1' bit to clear respective flags)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	–	vm_uvlo	register_reset	uv_cp	drv_err	reset
Reset	–	–	–	0x1	0x1	0x1	0x0	0x1
Access Type	–	–	–	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read

BITFIELD	BITS	DESCRIPTION	DECODE
vm_uvlo	4	1: V_S undervoltage occured since last reset. (Hint: is active after initial bootup. Clear flag after bootup to detect fault when device is operating).	
register_reset	3	Hint: is active after initial bootup. Clear flag after bootup to detect regmap reset when device is operating.	0x0: Normal operation 0x1: Indicates that the register map is reset. All registers are cleared to reset values.
uv_cp	2	Charge pump undervoltage condition flag. (Hint: is active after initial bootup. Clear flag after bootup to detect fault when device is operating).	0x0: Normal operation 0x1: Indicates an undervoltage on the charge pump. The driver is disabled during undervoltage. This flag is latched for information.

BITFIELD	BITS	DESCRIPTION	DECODE
drv_err	1	Driver error flag	0x0: Normal operation 0x1: Indicates that the driver is shut down due to overtemperature or short circuit detection. Read DRV_STATUS for details. The flag can only be cleared when the temperature is below the limit again.
reset	0	Reset flag (Hint: is active after initial bootup. Clear flag after bootup to detect device is reset during operation)	0x0: Normal operation 0x1: Indicates that the IC is reset.

IFCNT (0x2)

Interface transmission counter.

This register becomes incremented with each successful UART interface write access. It can be read out to check the serial transmission for lost data. Read accesses do not change the content. Disabled in SPI operation. The counter wraps around from 255 to 0.

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	IFCNT[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
IFCNT	7:0	Interface transmission counter. This register becomes incremented with each successful UART interface write access. It can be read out to check the serial transmission for lost data. Read accesses do not change the content. Disabled in SPI operation. The counter wraps around from 255 to 0.

NODECONF (0x3)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	SENDDELAY[3:0]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Write, Read			
BIT	7	6	5	4	3	2	1	0
Field	NODEADDR[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
SENDDELAY	11:8	SWUART Node Configuration	0x0: 8 bit times (not allowed with multiple nodes) 0x2: 3 x 8 bit times 0x4: 5 x 8 bit times 0x6: 7 x 8 bit times 0x8: 9 x 8 bit times 0xA: 11 x 8 bit times 0xC: 13 x 8 bit times 0xE: 15 x 8 bit times
NODEADDR	7:0	*NODEADDR*: These eight bits set the address of device for the UART interface. The address becomes incremented by one up to seven as defined by SDI, SCK, CSN. CSN, SCK, SDI 000: +0 001: +1 010: +2 011: +3 100: +4 101: +5 110: +6 111: +7 Range: 0 to 254 (do not increment beyond 254)	

IOIN (0x4)

Reads the state of all input pins available and returns IC revision in highest byte.

BIT	31	30	29	28	27	26	25	24
Field	VERSION[7:0]							
Reset								
Access Type	Read Only							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	SILICON_RV[2:0]		
Reset	–	–	–	–	–	0x0		
Access Type	–	–	–	–	–	Read Only		
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	ADC_ERR	EXT_CLK	EXT_RES_DET	OUTPUT	COMP_B1_B2	COMP_A1_A2	COMP_B	COMP_A
Reset	0x0	0x0	0x0	0x1	0x0	0x0	0x0	0x0
Access Type	Read Only	Read Only	Read Only	Write, Read	Read Only	Read Only	Read Only	Read Only
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	reserved	UART_EN	ENCN	DRV_ENN	ENCA	ENCB	REFR	REFL
Reset		0x0		0x0	0x0	0x0	0x0	0x0
Access Type	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
VERSION	31:24	0x40 = first version of the IC Identical numbers mean full digital compatibility.
SILICON_RV	18:16	Silicon revision number
ADC_ERR	15	1: Signals that the ADC is not working correctly. Do not utilize ADC features.
EXT_CLK	14	0: The internal oscillator is used for generating the clock-signal (12.5 MHz). 1: The external oscillator is used for generating the clock-signal.
EXT_RES_DET	13	1: External resistor between REF and GND 0: No external resistor detected
OUTPUT	12	Output polarity of SDO pin when UART is enabled through pin UART_EN. Its main purpose it to use SDO as NAO next address output signal for chain addressing of multiple ICs. Attention: reset value is 1 for use as NAO to next IC in single wire chain.
COMP_B1_B2	11	COMP_B1_B2 (StallGuard4 comparator B, for IC test)
COMP_A1_A2	10	COMP_A1_A2 (StallGuard4 comparator A, for IC test)
COMP_B	9	COMP_B (chopper comparator B, for IC test)
COMP_A	8	COMP_A (chopper comparator A, for IC test)
reserved	7	
UART_EN	6	1 = UART interface is enabled
ENCN	5	N-channel state
DRV_ENN	4	Driver disabled/enabled state.
ENCA	3	A-channel state
ENCB	2	B-channel state
REFR	1	

BITFIELD	BITS	DESCRIPTION
REFL	0	

X_COMPARE (0x5)

BIT	31	30	29	28	27	26	25	24
Field	X_COMPARE[31:24]							
Reset	0xFFFFFFFF							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	X_COMPARE[23:16]							
Reset	0xFFFFFFFF							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	X_COMPARE[15:8]							
Reset	0xFFFFFFFF							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	X_COMPARE[7:0]							
Reset	0xFFFFFFFF							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
X_COMPARE	31:0	Position comparison register for motion controller position strobe. X_COMPARE is an absolute position. The position pulse is available on output SWP_DIAG1. *XACTUAL = X_COMPARE*: Output signal PP (position pulse) becomes high. It returns to a low state, if the positions mismatch. If X_COMPARE_REPEAT is >1, X_COMPARE is the position reference for the **periodic** position strobe trigger output.

X_COMPARE_REPEAT (0x6)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	23	22	21	20	19	18	17	16
Field	X_COMPARE_REPEAT[23:16]							
Reset	0x0							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	X_COMPARE_REPEAT[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	X_COMPARE_REPEAT[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
X_COMPARE_REPEAT	23:0	This register defines a relative distance in microsteps (based on MRES confguration). If set to >1, the position compare pulse is raised every time a multiple of X_COMPARE_REPEAT µsteps have been made. Thereby, the X_COMPARE register defines the base position for the modulo calculation of X_COMPARE_REPEAT steps have been made into positive or negative direction. X_COMPARE is incremented/decremented by X_COMPARE_REPEAT when the X_COMPARE position has been reached.

DRV_CONF (0xA)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	7	6	5	4	3	2	1	0
Field	–	–	SLOPE_CONTROL[1:0]		–	–	CURRENT_RANGE[1:0]	
Reset	–	–	0x0		–	–	0x0	
Access Type	–	–	Write, Read		–	–	Write, Read	

BITFIELD	BITS	DESCRIPTION	DECODE
SLOPE_CONTROL	5:4	Slope Control Setting	0x0: 100V/µs 0x1: 200V/µs 0x2: 400V/µs 0x3: 800V/µs
CURRENT_RANGE	1:0	This setting allows a basic adaptation of the drivers RDSon current sensing to the motor current range. Select the lowest fitting range for best current precision. The value is the peak current setting.	0x0: 1A 0x1: 2A 0x2: 3A 0x3: 3A

GLOBAL SCALER (0xB)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	7	6	5	4	3	2	1	0
Field	GLOBALSCALER[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
GLOBALSCALER	7:0	Global scaling of motor current. This value is multiplied to the current scaling to adapt a drive to a certain motor type. This value should be chosen before tuning other settings, because it also influences chopper hysteresis. This value is just to finetune the motor current. 0: Full scale (or write 256) 1 … 31: Not allowed for operation 32 … 255: 32/256 … 255/256 of maximum current. *Hint:* Values >128 recommended for best results

IHOLD_IRUN (0x10)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	IRUNDELAY[3:0]			
Reset	–	–	–	–	0x4			
Access Type	–	–	–	–	Write, Read			
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	IHOLDDELAY[3:0]			
Reset	–	–	–	–	0x1			
Access Type	–	–	–	–	Write, Read			
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	IRUN[4:0]				
Reset	–	–	–	0b11111				
Access Type	–	–	–	Write, Read				
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	–	IHOLD[4:0]				
Reset	–	–	–	0b01000				
Access Type	–	–	–	Write, Read				

BITFIELD	BITS	DESCRIPTION
IRUNDELAY	27:24	Controls the number of clock cycles for motor power up after start is detected. 0: instant power up 1..15: Delay per current increment step in multiple of IRUNDELAY x 512 clocks
IHOLDDELAY	19:16	Controls the number of clock cycles for motor power down after a motion as soon as standstill is detected (*stst* = 1) and *TPOWERDOWN* has expired. The smooth transition avoids a motor jerk upon power down. 0: Instant power down 1..15: Delay per current reduction step in multiple of 2^18 clocks
IRUN	12:8	Motor run current (0 = 1/32…31 = 32/32) *Hint:* Use a setting between 16 to 31 for best microstep performance.

BITFIELD	BITS	DESCRIPTION
IHOLD	4:0	Standstill current (0 = 1/32…31 = 32/32) In combination with StealthChop2 mode, setting *IHOLD* = 0 allows to choose freewheeling or coil short circuit for motor standstill.

TPOWERDOWN (0x11)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	7	6	5	4	3	2	1	0
Field	TPOWERDOWN[7:0]							
Reset	0xA							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TPOWERDOWN	7:0	*TPOWERDOWN* sets the delay time after stand still (*stst*) of the motor to motor current power down. Time range is about 0 to 4 seconds. *Attention: A minimum setting of 2 is required to allow automatic tuning of StealthChop2 PWM_OFFS_AUTO.* *Reset Default = 10* *0…((2^8) - 1) x 2^18 t_{CLK}*

TSTEP (0x12)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	TSTEP[19:16]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Read Only			
BIT	15	14	13	12	11	10	9	8
Field	TSTEP[15:8]							
Reset	0x0							
Access Type	Read Only							
BIT	7	6	5	4	3	2	1	0
Field	TSTEP[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
TSTEP	19:0	Actual measured time between two 1/256 microsteps derived from the step input frequency in units of 1/fCLK. Measured value is (2^20)-1 in case of overflow or standstill. All TSTEP related thresholds use a hysteresis of 1/16 of the compare value to compensate for jitter in the clock or the step frequency. The flag *small_hysteresis* modifies the hysteresis to a smaller value of 1/32. (*Txxx x* 15/16) -1 or (*Txxx x* 31/32) -1 is used as a second compare value for each comparison value. This means that the lower switching velocity equals the calculated setting, but the upper switching velocity is higher as defined by the hysteresis setting. When working with the motion controller, the measured TSTEP for a given velocity V is in the range $(2^{24}/V) \leq TSTEP \leq 2^{24}/V - 1$. In DcStep mode, *TSTEP* does not show the mean velocity of the motor, but the velocities for each microstep, which may not be stable and thus does not represent the real motor velocity in case it runs slower than the target velocity.

TPWMTHRS (0x13)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	TPWMTHRS[19:16]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Write, Read			

BIT	15	14	13	12	11	10	9	8
Field	TPWMTHRS[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TPWMTHRS[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TPWMTHRS	19:0	This is the upper velocity for StealthChop2 voltage PWM mode. *TSTEP* ≥ *TPWMTHRS* ● StealthChop2 PWM mode is enabled, if configured ● DcStep is disabled

TCOOLTHRS (0x14)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	TCOOLTHRS[19:16]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	TCOOLTHRS[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TCOOLTHRS[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TCOOLTHRS	19:0	This is the lower threshold velocity for switching on smart energy CoolStep and StallGuard feature. (unsigned) Set this parameter to disable CoolStep at low speeds, where it cannot work reliably. The stop on stall function (enable with *sg_stop* when using internal motion controller) and the stall output signal become enabled when exceeding this velocity. In non-DcStep mode, it becomes disabled again once the velocity falls below this threshold. *TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: ● CoolStep is enabled, if configured *TCOOLTHRS* ≥ *TSTEP* ● Stop on stall is enabled, if configured ● Stall output signal (DIAG0/1) is enabled, if configured

THIGH (0x15)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	THIGH[19:16]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Write, Read			
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	THIGH[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	THIGH[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
THIGH	19:0	This velocity setting allows velocity dependent switching into a different chopper mode and fullstepping to maximize torque. (unsigned) The stall detection feature becomes switched off for 2 to 3 electrical periods whenever passing *THIGH* threshold to compensate for the effect of switching modes. *TSTEP* ≤ *THIGH*: ● CoolStep is disabled (motor runs with normal current scale). ● StealthChop2 voltage PWM mode is disabled. ● If *vhighchm* is set, the chopper switches to *chm* = 1 with *TFD* = 0 (constant off time with slow decay, only). ● If *vhighfs* is set, the motor operates in fullstep mode and the stall detection is switched over to DcStep stall detection.

RAMPMODE (0x20)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	–	–	–	–	RAMPMODE[1:0]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	

BITFIELD	BITS	DESCRIPTION	DECODE
RAMPMODE	1:0	Motion Controller ramping mode	0x0: Positioning mode (using all A, D, and V parameters) 0x1: Velocity mode to positive *VMAX* (using *AMAX* acceleration) 0x2: Velocity mode to negative *VMAX* (using *AMAX* acceleration) 0x3: Hold mode (velocity remains unchanged, unless stop event occurs)

XACTUAL (0x21)

BIT	31	30	29	28	27	26	25	24
Field	XACTUAL[31:24]							
Reset	0x0							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	XACTUAL[23:16]							
Reset	0x0							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	XACTUAL[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	XACTUAL[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
XACTUAL	31:0	Actual motor position (signed) *Hint:* This value normally should only be modified, when homing the drive. In positioning mode, modifying the register content starts a motion.

VACTUAL (0x22)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	VACTUAL[23:16]							
Reset	0x0							
Access Type	Read Only							
BIT	15	14	13	12	11	10	9	8
Field	VACTUAL[15:8]							
Reset	0x0							
Access Type	Read Only							

BIT	7	6	5	4	3	2	1	0
Field	VACTUAL[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
VACTUAL	23:0	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]

VSTART (0x23)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	VSTART[17:16]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	15	14	13	12	11	10	9	8
Field	VSTART[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	VSTART[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
VSTART	17:0	Motor start velocity (unsigned) For universal use, set *VSTOP* ≥ *VSTART.* This is not required if the motion distance is sufficient to ensure deceleration from *VSTART* to *VSTOP*. 0..(2^{18})-1 [µsteps / t]

A1 (0x24)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	A1[17:16]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	15	14	13	12	11	10	9	8
Field	A1[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	A1[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
A1	17:0	First acceleration between *VSTART* and *V1* (unsigned) 0..(2^{18})-1 [µsteps / ta²]

V1 (0x25)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	V1[19:16]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	V1[15:8]							
Reset	0x0							
Access Type	Write, Read							

BIT	7	6	5	4	3	2	1	0
Field	V1[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
V1	19:0	First acceleration / deceleration phase threshold velocity (unsigned) 0: Disables *A1* and *D1* phase, use *AMAX*, *DMAX* only 0..(2^{20})-1 [µsteps / t]

AMAX (0x26)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	AMAX[17:16]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	15	14	13	12	11	10	9	8
Field	AMAX[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	AMAX[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
AMAX	17:0	Second acceleration between *V1* and *VMAX* (unsigned) This is the acceleration and deceleration value for velocity mode. 0..(2^{18})-1 [µsteps / ta²]

VMAX (0x27)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	VMAX[22:16]						
Reset	–	0x0						
Access Type	–	Write, Read						
BIT	15	14	13	12	11	10	9	8
Field	VMAX[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	VMAX[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
VMAX	22:0	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]

DMAX (0x28)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	DMAX[17:16]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	

BIT	15	14	13	12	11	10	9	8
Field	DMAX[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	DMAX[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
DMAX	17:0	Deceleration between *VMAX* and *V1* (unsigned) 0..(2^{18})-1 [µsteps / ta²]

TVMAX (0x29)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	TVMAX[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TVMAX[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TVMAX	15:0	Minimum time for constant velocity segments in multiple of 512 clocks. 0: Disables minimum duration setting for constant velocity phase >0: A minimum duration of constant velocity is inserted in between any change from acceleration to deceleration or vice versa to reduce jerk (0…(2^{16})-1) x 512 t_{CLK} **Note**: Configure this register after setting VMAX when in position mode and standstill. Set TVMAX = 0 during velocity mode to avoid triggering the TVMAX delay when switching back to ramp mode.

D1 (0x2A)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	D1[17:16]	
Reset	–	–	–	–	–	–	0xA	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	D1[15:8]							
Reset	0xA							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	D1[7:0]							
Reset	0xA							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
D1	17:0	Deceleration between *V1* and *VSTOP* (unsigned) **Attention**: Do not set 0 in positioning mode, even if V1=0! 1..(2^{18})-1 [µsteps / ta²] Reset Default = 10

VSTOP (0x2B)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	VSTOP[17:16]	
Reset	–	–	–	–	–	–	0xA	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	15	14	13	12	11	10	9	8
Field	VSTOP[15:8]							
Reset	0xA							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	VSTOP[7:0]							
Reset	0xA							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
VSTOP	17:0	Motor stop velocity (unsigned) Hint: Set VSTOP ≥ VSTART to allow positioning for short distances **Attention**: Do not set 0 in positioning mode, minimum 10 recommended! 1…(2^{18})-1 [µsteps / t] Reset Default = 10

TZEROWAIT (0x2C)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	15	14	13	12	11	10	9	8
Field	TZEROWAIT[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TZEROWAIT[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TZEROWAIT	15:0	Defines the waiting time after ramping down to zero velocity before next movement or direction inversion can start. Time range is about 0 to 2 seconds. This setting avoids excess acceleration e.g. from *VSTOP* to -*VSTART*. 0..(2^{16})-1 x 512 t_{CLK}

XTARGET (0x2D)

BIT	31	30	29	28	27	26	25	24
Field	XTARGET[31:24]							
Reset	0x0							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	XTARGET[23:16]							
Reset	0x0							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	XTARGET[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	XTARGET[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
XTARGET	31:0	Target position for ramp mode (signed). Write a new target position to this register 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 to trigger a second acceleration phase, if desired. -2^{31}.. +(2^{31})-1

V2 (0x2E)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	V2[19:16]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Write, Read			
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	V2[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	V2[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
V2	19:0	Velocity difference from VMAX for activation of acceleration segments with AMAX/2 and DMAX/2. 0: Disables *AMAX/2* and *DMAX/2* phase, use *AMAX*, *DMAX* only 0..(2^{20})-1 [µsteps / t]

A2 (0x2F)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	A2[17:16]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	15	14	13	12	11	10	9	8
Field	A2[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	A2[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
A2	17:0	Acceleration between *V1* and *V2* (unsigned) 0..(2^{18})-1 [µsteps / ta²]

D2 (0x30)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	D2[17:16]	
Reset	–	–	–	–	–	–	0xA	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	15	14	13	12	11	10	9	8
Field	D2[15:8]							
Reset	0xA							
Access Type	Write, Read							

BIT	7	6	5	4	3	2	1	0
Field	D2[7:0]							
Reset	0xA							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
D2	17:0	Deceleration between *V2* and *V1* (unsigned) *Attention: Do not set 0 in positioning mode, even if V2 = 0!* 1..(2^{18})-1 [µsteps / ta²] Reset Default = 10

VDCMIN (0x33)

dcStep start velocity

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	VDCMIN[14:8]						
Reset	–	0x0						
Access Type	–	Write, Read						
BIT	15	14	13	12	11	10	9	8
Field	VDCMIN[7:0]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	reserved[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
VDCMIN	22:8	Automatic commutation DcStep becomes enabled above velocity V*DCMIN* (unsigned) In this mode, the actual position is determined by the sensorless motor commutation and becomes fed back to *XACTUAL*. In case the motor becomes heavily loaded, *VDCMIN* also is used as the minimum step velocity. Activate stop on stall (*sg_stop*) to detect step loss. 0: Disable, DcStep off |*VACT*| ≥ *VDCMIN* ≥ 256: ● Triggers the same actions as exceeding *THIGH* setting. ● Switches on automatic commutation DcStep *Hint:* Also set *DCCTRL* parameters to operate DcStep. (Only bits 22 to 8 are used for value and for comparison).
reserved	7:0	Reads always 0

SW_MODE (0x34)

Switch mode configuration

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	virtual_stop_enc	en_virtual_stop_r	en_virtual_stop_l	en_softstop	sg_stop	en_latch_encoder	latch_r_inactive
Reset	–	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	–	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	latch_r_active	latch_l_inactive	latch_l_active	swap_lr	pol_stop_r	pol_stop_l	stop_r_enable	stop_l_enable
Reset	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
virtual_stop_enc	14	Source for virtual stop (VIRTUAL_STOP_L and VIRTUAL_STOP_R) 0: Virtual stop relates to ramp generator position *XACTUAL* 1: Virtual stop relates to encoder position *X_ENC*	
en_virtual_stop_r	13	1: Enables automatic motor stop during active right virtual stop condition	
en_virtual_stop_l	12	1: Enables automatic motor stop during active left virtual stop condition	
en_softstop	11	0: Hard stop 1: Soft stop The soft stop mode always uses the deceleration ramp settings *DMAX*, V*1*, *D1*, V2, D2, *VSTOP,* and *TZEROWAIT* for stopping the motor. A stop occurs when the velocity sign matches the reference switch position (REFL, *VIRTUAL_STOP_L* for negative velocities, REFR, *VIRTUAL_STOP_R* for positive velocities) and the respective switch stop function is enabled. A hard stop also uses *TZEROWAIT* before the motor becomes released. *Attention: Do not use soft stop in combination with StallGuard2. Use soft stop for StealthChop operation at high velocity. In this case, hard stop must be avoided, as it can result in severe overcurrent.*	0x0: Hard stop 0x1: Soft stop
sg_stop	10	Enable stop by StallGuard2 (also available in DcStep mode). Disable to release motor after stop event. Program *TCOOLTHRS* for velocity threshold. *Hint:* Do not enable during motor spin-up, wait until the motor velocity exceeds a certain value, where StallGuard2 delivers a stable result. This velocity threshold should be programmed using TCOOLTHRS.	0x0: disabled 0x1: enabled
en_latch_encoder	9	1: Latch encoder position to *ENC_LATCH* upon reference switch event.	
latch_r_inactive	8	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.*	
latch_r_active	7	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.*	

BITFIELD	BITS	DESCRIPTION	DECODE
latch_l_inactive	6	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*.	
latch_l_active	5	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*.	
swap_lr	4	1: Swap the left and the right reference switch input REFL and REFR.	
pol_stop_r	3	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.	
pol_stop_l	2	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.	
stop_r_enable	1	1: Enables automatic motor stop during active right reference switch input. *Hint:* The motor restarts in case the stop switch becomes released.	
stop_l_enable	0	1: Enables automatic motor stop during active left reference switch input. *Hint:* The motor restarts in case the stop switch becomes released.	

RAMP_STAT (0x35)

Ramp status and switch event status

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	status_virtual_stop_r	status_virtual_stop_l	status_sg	second_move	t_zerowait_active	vzero	position_reached	velocity_reached
Reset	0x1	0x1	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Read Only	Read Only	Read Only	Write 1 to Clear, Read	Read Only	Read Only	Read Only	Read Only

BIT	7	6	5	4	3	2	1	0
Field	event_pos_reached	event_stop_sg	event_stop_r	event_stop_l	status_latch_r	status_latch_l	status_stop_r	status_stop_l
Reset	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
Access Type	Write 1 to Clear, Read	Write 1 to Clear, Read	Read Only	Read Only	Write 1 to Clear, Read	Write 1 to Clear, Read	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION
status_virtual_stop_r	15	Virtual reference switch right status (1 = active)
status_virtual_stop_l	14	Virtual reference switch left status (1 = active)
status_sg	13	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.
second_move	12	1: Signals that the automatic ramp required moving back in the opposite direction, example, due to on-the-fly parameter change. (Write '1' to clear)
t_zerowait_active	11	1: Signals that *TZEROWAIT* is active after a motor stop. During this time, the motor is in standstill.
vzero	10	1: Signals that the actual velocity is 0.
position_reached	9	1: Signals that the target position is reached. This flag becomes set while *XACTUAL* and *XTARGET* match.
velocity_reached	8	1: Signals that the target velocity is reached. This flag becomes set while *VACTUAL* and *VMAX* match.
event_pos_reached	7	1: Signals, that the target position is reached (*position_reached* becoming active). (Write '1' to clear flag and interrupt condition) This bit is ORed to the *interrupt output* signal.
event_stop_sg	6	1: Signals an active StallGuard2 stop event. Resetting the register clears the stall condition and the motor may restart motion, unless the motion controller is stopped. (Write '1' to clear flag and interrupt condition) This bit is ORed to the *interrupt output* signal.
event_stop_r	5	1: Active stop right condition due to stop switch or virtual stop. 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 remains 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 continues motion. This bit is ORed to the *interrupt output* signal.
event_stop_l	4	1: Active stop left condition due to stop switch or virtual stop. 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 remains 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 continues motion. This bit is ORed to the *interrupt output* signal.
status_latch_r	3	1: Latch right ready (enable position latching using *SW_MODE* settings *latch_r_active* or *latch_r_inactive*) (Write '1' to clear)

BITFIELD	BITS	DESCRIPTION
status_latch_l	2	1: Latch left ready (enable position latching using *SW_MODE* settings *latch_l_active* or *latch_l_inactive*) (Write '1' to clear)
status_stop_r	1	Reference switch right status (1 = active)
status_stop_l	0	Reference switch left status (1 = active)

XLATCH (0x36)

Ramp generator latch position

BIT	31	30	29	28	27	26	25	24
Field	XLATCH[31:24]							
Reset								
Access Type	Read Only							
BIT	23	22	21	20	19	18	17	16
Field	XLATCH[23:16]							
Reset								
Access Type	Read Only							
BIT	15	14	13	12	11	10	9	8
Field	XLATCH[15:8]							
Reset								
Access Type	Read Only							
BIT	7	6	5	4	3	2	1	0
Field	XLATCH[7:0]							
Reset								
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
XLATCH	31:0	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.

ENCMODE (0x38)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	enc_sel_decimal	latch_x_act	clr_enc_x
Reset	–	–	–	–	–	0x0	0x0	0x0
Access Type	–	–	–	–	–	Write, Read	Write, Read	Write, Read
BIT	7	6	5	4	3	2	1	0
Field	pos_neg_edge[1:0]		clr_once	clr_cont	ignore_AB	pol_N	pol_B	pol_A
Reset	0x0			0x0	0x0	0x0	0x0	0x0
Access Type	Write, Read		Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
enc_sel_decimal	10	Encoder prescaler mode selection	0x0: Encoder prescaler divisor binary mode: Counts *ENC_CONST(fractional part)* /65536 0x1: Encoder prescaler divisor decimal mode: Counts in *ENC_CONST(fractional part)* /10000
latch_x_act	9	Position latch configuration	0x0: disabled 0x1: 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*.
clr_enc_x	8	Encoder latch configuration	0x0: Upon N event, *X_ENC* becomes latched to *ENC_LATCH* only 0x1: Latch and additionally clear encoder counter *X_ENC* at N-event
pos_neg_edge	7:6	N channel event sensitivity	0x0: N channel event is active during an active N event level 0x1: N channel is valid upon active going N event 0x2: N channel is valid upon inactive going N event 0x3: N channel is valid upon active going and inactive going N event
clr_once	5	Position latch configuration	0x0: disabled 0x1: Latch or latch and clear *X_ENC* on the next N event following the write access
clr_cont	4	Position latch configuration	0x0: disabled 0x1: 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)
ignore_AB	3	N event configuration	0x0: An N event occurs only when polarities given by *pol_N*, *pol_A* and *pol_B* match. 0x1: Ignore A and B polarity for N channel event
pol_N	2	Defines active polarity of N	0x0: low active 0x1: high active

BITFIELD	BITS	DESCRIPTION	DECODE
pol_B	1	Required B polarity for an N channel event	0x0: neg 0x1: pos
pol_A	0	Required A polarity for an N channel event	0x0: neg 0x1: pos

X_ENC (0x39)

BIT	31	30	29	28	27	26	25	24
Field	X_ENC[31:24]							
Reset	0x0							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	X_ENC[23:16]							
Reset	0x0							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	X_ENC[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	X_ENC[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
X_ENC	31:0	Actual encoder position (signed)

ENC_CONST (0x3A)

BIT	31	30	29	28	27	26	25	24
Field	ENC_CONST[31:24]							
Reset	0x10000							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	ENC_CONST[23:16]							
Reset	0x10000							
Access Type	Write, Read							

BIT	15	14	13	12	11	10	9	8
Field	ENC_CONST[15:8]							
Reset	0x10000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	ENC_CONST[7:0]							
Reset	0x10000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
ENC_CONST	31:0	Accumulation constant (signed) 16 bit integer part, 16 bit fractional part *X_ENC* accumulates +/- *ENC_CONST* / (2^{16} x *X_ENC*) (binary) or +/-*ENC_CONST* / (10^4 x *X_ENC*) (decimal) *ENCMODE* bit *enc_sel_decimal* switches between decimal and binary setting. Use the sign to match rotation direction! binary: ± [µsteps/2^{16}] ±(0 … 32767.999847) decimal: ±(0.0 … 32767.9999) *reset default = 1.0 (=65536)*

ENC_STATUS (0x3B)

Encoder status information

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	–	deviation_warn	n_event
Reset	–	–	–	–	–	–	0x0	0x0
Access Type	–	–	–	–	–	–	Write 1 to Clear, Read	Write 1 to Clear, Read

BITFIELD	BITS	DESCRIPTION	DECODE
deviation_warn	1		0x0: No warning 0x1: deviation_warn cannot be cleared while a warning still persists. Set *ENC_DEVIATION* to zero to disable.
n_event	0		0x0: No event 0x1: Event detected. To clear the status bit, write with a 1 bit at the corresponding position.

ENC_LATCH (0x3C)

BIT	31	30	29	28	27	26	25	24
Field	ENC_LATCH[31:24]							
Reset	0x0							
Access Type	Read Only							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	ENC_LATCH[23:16]							
Reset	0x0							
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	ENC_LATCH[15:8]							
Reset	0x0							
Access Type	Read Only							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ENC_LATCH[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ENC_LATCH	31:0	Encoder position *X_ENC* latched on N event

ENC_DEVIATION (0x3D)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	ENC_DEVIATION[19:16]			
Reset	–	–	–	–	0x0			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	ENC_DEVIATION[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	ENC_DEVIATION[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
ENC_DEVIATION	19:0	Maximum number of steps deviation between encoder counter and XACTUAL for deviation warning. Result in flag ENC_STATUS.*deviation_warn* 0 = Function is off.

VIRTUAL_STOP_L (0x3E)

BIT	31	30	29	28	27	26	25	24
Field	VIRTUAL_STOP_L[31:24]							
Reset	0x0							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	VIRTUAL_STOP_L[23:16]							
Reset	0x0							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	VIRTUAL_STOP_L[15:8]							
Reset	0x0							
Access Type	Write, Read							

BIT	7	6	5	4	3	2	1	0
Field	VIRTUAL_STOP_L[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
VIRTUAL_STOP_L	31:0	Virtual stop switch based on encoder or ramp position. A stop is raised, based on the signed comparison. *virtual_stop_enc* = 1: *X_ENC* <= *VIRTUAL_STOP_L* *virtual_stop_enc* = 0: *XACTUAL* <= *VIRTUAL_STOP_L* -2^31… +(2^31)-1

VIRTUAL_STOP_R (0x3F)

BIT	31	30	29	28	27	26	25	24
Field	VIRTUAL_STOP_R[31:24]							
Reset	0x0							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	VIRTUAL_STOP_R[23:16]							
Reset	0x0							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	VIRTUAL_STOP_R[15:8]							
Reset	0x0							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	VIRTUAL_STOP_R[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
VIRTUAL_STOP_R	31:0	Virtual Stop Switch based on Encoder. A stop is raised, based on the signed comparison. *virtual_stop_enc* = 1: *X_ENC* >= *VIRTUAL_STOP_R* *virtual_stop_enc* = 0: *XACTUAL* >= *VIRTUAL_STOP_R* -2^31… +(2^31)-1

ADC_VSUPPLY_AIN (0x50)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	ADC_AIN[12:8]				
Reset	–	–	–					
Access Type	–	–	–	Read Only				
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	ADC_AIN[7:0]							
Reset								
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	ADC_VSUPPLY[12:8]				
Reset	–	–	–					
Access Type	–	–	–	Read Only				
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ADC_VSUPPLY[7:0]							
Reset								
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ADC_AIN	28:16	Value of voltage at AIN pin in integer. Update rate = each 2048 clocks V_{AIN} = ADC_AIN * 305.2uV
ADC_VSUPPLY	12:0	Actual value of voltage on V_S (filtered with low pass filter). Update rate: each 2048 clocks V_S = ADC_VSUPPLY * 9.732mV

ADC_TEMP (0x51)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	RESERVED[12:8]				
Reset	–	–	–					
Access Type	–	–	–	Read Only				
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	RESERVED[7:0]							
Reset								
Access Type	Read Only							

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	ADC_TEMP[12:8]				
Reset	–	–	–					
Access Type	–	–	–	Read Only				
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ADC_TEMP[7:0]							
Reset								
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
RESERVED	28:16	
ADC_TEMP	12:0	Actual temperature(filtered with low pass filter). Update rate: each 2048 clocks. $\text{TEMP}[^\circ C] = \frac{\text{ADC\_TEMP} - 2038}{7.7}$

OTW_OV_VTH (0x52)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	OVERTEMPPREWARNING_VTH[12:8]				
Reset	–	–	–	0xB92				
Access Type	–	–	–	Write, Read				
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	OVERTEMPPREWARNING_VTH[7:0]							
Reset	0xB92							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	OVERVOLTAGE_VTH[12:8]				
Reset	–	–	–	0xF25				
Access Type	–	–	–	Write, Read				
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	OVERVOLTAGE_VTH[7:0]							
Reset	0xF25							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
OVERTEMPPREWARNING_VTH	28:16	Overtemperature warning threshold register: ADC_TEMP >= OVERTEMPPREWARNING_VTH Overtemperature prewarning is triggered. (Reset: 0xB92 equals 120°C)

BITFIELD	BITS	DESCRIPTION
OVERVOLTAGE_VTH	12:0	Overvoltage threshold for output OV. Default: 38V, 36 V equals 1.125 V at ADC inputs.

MSLUT_0 (0x60)

Microstep table entries 0…31

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_0[31:24]							
Reset	0xAAAAB554							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	MSLUT_0[23:16]							
Reset	0xAAAAB554							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	MSLUT_0[15:8]							
Reset	0xAAAAB554							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	MSLUT_0[7:0]							
Reset	0xAAAAB554							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_0	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUT_1 (0x61)

Microstep table entries 32…63

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_1[31:24]							
Reset	0x4A9554AA							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	MSLUT_1[23:16]							
Reset	0x4A9554AA							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	MSLUT_1[15:8]							
Reset	0x4A9554AA							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	MSLUT_1[7:0]							
Reset	0x4A9554AA							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_1	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUT_2 (0x62)

Microstep table entries 64…95

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_2[31:24]							
Reset	0x24492929							
Access Type	Write, Read							

BIT	23	22	21	20	19	18	17	16
Field	MSLUT_2[23:16]							
Reset	0x24492929							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	MSLUT_2[15:8]							
Reset	0x24492929							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	MSLUT_2[7:0]							
Reset	0x24492929							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_2	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUT_3 (0x63)

Microstep table entries 96…127

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_3[31:24]							
Reset	0x10104222							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	MSLUT_3[23:16]							
Reset	0x10104222							
Access Type	Write, Read							

BIT	15	14	13	12	11	10	9	8
Field	MSLUT_3[15:8]							
Reset	0x10104222							
Access Type	Write, Read							

BIT	7	6	5	4	3	2	1	0
Field	MSLUT_3[7:0]							
Reset	0x10104222							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_3	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUT_4 (0x64)

Microstep table entries 128…159

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_4[31:24]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							

BIT	23	22	21	20	19	18	17	16
Field	MSLUT_4[23:16]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							

BIT	15	14	13	12	11	10	9	8
Field	MSLUT_4[15:8]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							

BIT	7	6	5	4	3	2	1	0
Field	MSLUT_4[7:0]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_4	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUT_5 (0x65)

Microstep table entries 160…191

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_5[31:24]							
Reset	0xB5BB777D							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	MSLUT_5[23:16]							
Reset	0xB5BB777D							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	MSLUT_5[15:8]							
Reset	0xB5BB777D							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MSLUT_5[7:0]							
Reset	0xB5BB777D							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_5	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUT_6 (0x66)

Microstep table entries 192…223

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_6[31:24]							
Reset	0x49295556							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	MSLUT_6[23:16]							
Reset	0x49295556							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	MSLUT_6[15:8]							
Reset	0x49295556							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	MSLUT_6[7:0]							
Reset	0x49295556							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_6	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUT_7 (0x67)

Microstep table entries 224…255

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_7[31:24]							
Reset	0x404222							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	MSLUT_7[23:16]							
Reset	0x404222							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	MSLUT_7[15:8]							
Reset	0x404222							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MSLUT_7[7:0]							
Reset	0x404222							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_7	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W*= %00: -1 %01: +0 %10: +1 %11: +2 1: *W*= %00: +0 %01: +1 %10: +2 %11: +3 This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224* *reset default = sine wave table*

MSLUTSEL (0x68)

BIT	31	30	29	28	27	26	25	24
Field	X3[7:0]							
Reset	0xFF							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	X2[7:0]							
Reset	0xFF							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	X1[7:0]							
Reset	0x80							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	W3[1:0]		W2[1:0]		W1[1:0]		W0[1:0]	
Reset	0x1		0x1		0x1		0x2	
Access Type	Write, Read		Write, Read		Write, Read		Write, Read	

BITFIELD	BITS	DESCRIPTION
X3	31:24	LUT segment 1 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*
X2	23:16	LUT segment 1 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*
X1	15:8	LUT segment 1 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*
W3	7:6	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
W2	5:4	LUT width select from *ofs(X2)* to *ofs(X3-1)* 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

BITFIELD	BITS	DESCRIPTION
W1	3:2	LUT width select from *ofs(X1)* to *ofs(X2-1)* 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
W0	1:0	LUT width select from *ofs00* to *ofs(X1-1)* 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

MSLUTSTART (0x69)

Start values are transferred to the microstep registers *CUR_A* and *CUR_B*, whenever the reference position *MSCNT* = 0 is passed.

BIT	31	30	29	28	27	26	25	24
Field	OFFSET_SIN90[7:0]							
Reset	0x0							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	START_SIN90[7:0]							
Reset	0d247							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	START_SIN[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
OFFSET_SIN90	31:24	Signed offset for cosine wave +/-127 microsteps. Adapt *START_SIN90* to match the microstep wave table at position *MSCNT* = 0.
START_SIN90	23:16	*START_SIN90* gives the absolute value for cosine wave microstep table entry at *MSCNT* = 0 (table position 256 + *OFFSET_SIN90*).
START_SIN	7:0	*START_SIN* gives the absolute value at microstep table entry 0.

MSCNT (0x6A)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	MSCNT[9:8]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Read Only	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MSCNT[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
MSCNT	9:0	Microstep counter. Indicates actual position in the microstep table for *CUR_A*. *CUR_B* uses an offset of 256 (two-phase motor). *Hint:* Move to a position where *MSCNT* is zero before reinitializing *MSLUTSTART* or *MSLUT* and *MSLUTSEL*.

MSCURACT (0x6B)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	CUR_A[8]
Reset	–	–	–	–	–	–	–	0xF7
Access Type	–	–	–	–	–	–	–	Read Only
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	CUR_A[7:0]							
Reset	0xF7							
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	CUR_B[8]
Reset	–	–	–	–	–	–	–	0x0
Access Type	–	–	–	–	–	–	–	Read Only

BIT	7	6	5	4	3	2	1	0
Field	CUR_B[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
CUR_A	24:16	Actual microstep current for motor phase A (cosine wave) as read from MSLUT (not scaled by current).
CUR_B	8:0	Actual microstep current for motor phase B (sine wave) as read from MSLUT (not scaled by current).

CHOPCONF (0x6C)

BIT	31	30	29	28	27	26	25	24
Field	diss2vs	diss2g	reserved	intpol	MRES[3:0]			
Reset	0x0	0x0	0x0	0x1	0x0			
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read			
BIT	23	22	21	20	19	18	17	16
Field	TPFD[3:0]				vhighchm	vhighfs	–	TBL[1]
Reset	0x4						–	0b10
Access Type	Write, Read				Write, Read	Write, Read	–	Write, Read
BIT	15	14	13	12	11	10	9	8
Field	TBL[0]	chm	–	disfdcc	fd3	HEND_OFFSET[3:1]		
Reset	0b10		–	0x0		0x2		
Access Type	Write, Read	Write, Read	–	Write, Read	Write, Read	Write, Read		
BIT	7	6	5	4	3	2	1	0
Field	HEND_OFFSET[0]	HSTRT_TFD210[2:0]			TOFF[3:0]			
Reset	0x2	0x5			0x0			
Access Type	Write, Read	Write, Read			Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
diss2vs	31	Short to supply protection disable	0x0: Short to VS protection is on 0x1: Short to VS protection is disabled
diss2g	30	Short to GND protection disable	0x0: Short to GND protection is on 0x1: Short to GND protection is disabled
reserved	29	Reserved, do not use	
intpol	28	Interpolation to 256 microsteps	0x0: No interpolation 0x1: The actual microstep resolution (*MRES*) becomes extrapolated to 256 microsteps for smoothest motor operation.

BITFIELD	BITS	DESCRIPTION	DECODE
MRES	27:24	Micro step resolution selection %0000: Native 256 microstep setting. Normally use this setting with the internal motion controller. %0001 … %1000: 128, 64, 32, 16, 8, 4, 2, FULLSTEP Reduced microstep resolution. The resolution gives the number of microstep entries per sine quarter wave. The driver automatically uses microstep positions, which result in a symmetrical wave, when choosing a lower microstep resolution. step width = 2^{MRES} [microsteps].	
TPFD	23:20	Passive fast decay time *TPFD* allows dampening of motor mid-range resonances. Passive fast decay time setting controls duration of the fast decay phase inserted after bridge polarity change N_{CLK} = 128 x *TPFD* %0000: Disable %0001 … %1111: 1 … 15	
vhighchm	19	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 T_{OFF} setting automatically becomes doubled during high velocity operation to avoid doubling of the chopper frequency.	
vhighfs	18	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.	
TBL	16:15	*TBL* blank time setting. Sets comparator blank time in numbers of clock cycles. *Hint*: 24 or 36 clocks are recommended for most applications. Restriction for TBL = 0x0 : Use only in combination with external clock oscillator <= 8MHz Restriction for TBL = 0x1 : May be used with internal clock, or if external clock frequency <= 13MHz is applied.	0x0: 16 clocks 0x1: 24 clocks 0x2: 36 clocks 0x3: 48 clocks

BITFIELD	BITS	DESCRIPTION	DECODE
chm	14	Chopper mode selection. This is only effective if *en_pwm_mode* is set to 0 or *TSTEP* < *TPWMTHRS*	0x0: Standard mode (SpreadCycle) 0x1: 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.
disfdcc	12	Fast decay mode for chm = 1	0x0: Enables current comparator usage for termination of the fast decay cycle 0x1: Disables current comparator usage for termination of the fast decay cycle
fd3	11	TFD[3] *with chm* = 1: MSB of fast decay time setting *TFD*	
HEND_OFFSET	10:7	with chm = 0: HEND = hysteresis low value %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. with chm = 1: OFFSET = sine wave offset %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.	
HSTRT_TFD210	6:4	***with chm* = 0:** *HSTRT* hysteresis start value added to *HEND* %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 ***with chm = 1:*** *TFD [2..0]* fast decay time setting Fast decay time setting (MSB: *fd3*): %0000 … %1111: Fast decay time setting *TFD* with N_{CLK} = 32 x *TFD* (%0000: slow decay only)	
TOFF	3:0	T_{OFF} off time and driver enable Off time setting controls duration of slow decay phase N_{CLK} = 24 + 32 x *TOFF* %0000: Driver disable, all bridges off %0001: 1 – use only with *TBL* ≥ 2 %0010 … %1111: 2 … 15	

COOLCONF (0x6D)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	sfilt
Reset	–	–	–	–	–	–	–	0x0
Access Type	–	–	–	–	–	–	–	Write, Read

BIT	23	22	21	20	19	18	17	16
Field	–	sgt[6:0]						
Reset	–	0x0						
Access Type	–	Write, Read						

BIT	15	14	13	12	11	10	9	8
Field	seimin	sedn[1:0]		–	semax[3:0]			
Reset	0x0	0x0		–	0x0			
Access Type	Write, Read	Write, Read		–	Write, Read			

BIT	7	6	5	4	3	2	1	0
Field	–	seup[1:0]		–	semin[3:0]			
Reset	–	0x0		–	0x0			
Access Type	–	Write, Read		–	Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
sfilt	24	StallGuard2 and StallGuard4 filter enable	0x0: Standard mode, high time resolution for StallGuard 0x1: Filtered mode, StallGuard signal updated for each four fullsteps only to compensate for motor pole tolerances
sgt	22:16	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.	
seimin	15	Minimum current for smart current control	0x0: 1/2 of current setting (*IRUN*) (when used with StealthChop requires IRUN ≥ 16) 0x1: 1/4 of current setting (*IRUN*) (when used with StealthChop requires IRUN ≥ 28)

BITFIELD	BITS	DESCRIPTION	DECODE
sedn	14:13	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	
semax	11:8	StallGuard2 hysteresis value for smart current control If the StallGuard2 result is equal to or above (*SEMIN* + *SEMAX* + 1) x 32, the motor current becomes decreased to save energy. %0000 … %1111: 0 … 15	
seup	6:5	Current up step width Current increment steps per measured StallGuard2 value %00 … %11: 1, 2, 4, 8	
semin	3:0	Minimum StallGuard2 value for smart current control and smart current enable If the StallGuard2 result falls below *SEMIN x* 32, the motor current becomes increased to reduce motor load angle. %0000: smart current control CoolStep off %0001 … %1111: 1 … 15	

DCCTRL (0x6E)

DcStep (*DC*) automatic commutation configuration register (enable through pin DCEN or *VDCMIN*).

Hint: Using a higher microstep resolution or interpolated operation, DcStep delivers a better StallGuard signal.

DC_SG is also available above VHIGH if vhighfs is activated. For best result, also set vhighchm.

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	DC_SG[7:0]							
Reset	0x0							
Access Type	Write, Read							

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	DC_TIME[9:8]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	7	6	5	4	3	2	1	0
Field	DC_TIME[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
DC_SG	23:16	Max. PWM on time for step loss detection using DcStep StallGuard2 in DcStep mode (*DC_SG* * 16/f_{CLK}) Set slightly higher than *DC_TIME*/16 0 = disable
DC_TIME	9:0	Upper PWM on time limit for commutation (*DC_TIME* * 1/f_{CLK}). Set slightly above effective blank time *TBL*.

DRV_STATUS (0x6F)

BIT	31	30	29	28	27	26	25	24
Field	stst	olb	ola	s2gb	s2ga	otpw	ot	stallguard
Reset								
Access Type	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	CS_ACTUAL[4:0]				
Reset	–	–	–					
Access Type	–	–	–	Read Only				
BIT	15	14	13	12	11	10	9	8
Field	fsactive	stealth	s2vsb	s2vsa	–	–	SG_RESULT[9:8]	
Reset					–	–		
Access Type	Read Only	Read Only	Read Only	Read Only	–	–	Read Only	
BIT	7	6	5	4	3	2	1	0
Field	SG_RESULT[7:0]							
Reset								
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION	DECODE
stst	31	Standstill indicator This flag indicates motor standstill in each operation mode. This occurs 2^{20} clocks after the last step pulse.	
olb	30	Open load indicator phase B	0x0: normal operation 0x1: Open load detected on phase B. *Hint:* This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion only.
ola	29	Open load indicator phase A	0x0: normal operation 0x1: Open load detected on phase A. *Hint:* This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion only.
s2gb	28	Short to ground indicator phase B	0x0: normal operation 0x1: Short to GND detected on phase B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (T_{OFF} = 0) or by the DRV_ENN input.
s2ga	27	Short to ground indicator phase A	0x0: normal operation 0x1: Short to GND detected on phase A. The driver becomes disabled. The flags stay active, until the driver is disabled by software (T_{OFF} = 0) or by the DRV_ENN input.
otpw	26	Overtemperature prewarning flag	0x0: Normal operation 0x1: Overtemperature pre-warning threshold is exceeded. The overtemperature prewarning flag is common for both bridges.
ot	25	Overtemperature flag	0x0: Normal operation 0x1: Overtemperature limit has been reached. Drivers become disabled until *otpw* is also cleared due to cooling down of the IC. The overtemperature flag is common for both bridges.
stallguard	24	StallGuard2/StallGuard4 status	0x0: Normal operation 0x1: Motor stall detected by StallGuard2 (in SpreadCycle operation) resp. by StallGuard4 (in StealthChop2 operatoin) or DcStep stall (in DcStep mode).
CS_ACTUAL	20:16	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	
fsactive	15	Full step active indicator	0x0: Microstepping active 0x1: Indicates that the driver has switched to fullstep as defined by chopper mode settings and velocity thresholds.

BITFIELD	BITS	DESCRIPTION	DECODE
stealth	14	StealthChop2 indicator	0x0: StealthChop2 not active 0x1: Driver operates in StealthChop2 mode.
s2vsb	13	Short to supply indicator phase B	0x0: no error 0x1: Short to supply detected on phase B. The driver becomes disabled. The flags stay active, until the driver is disabled by software (T_{OFF} = 0) or by the ENN input.
s2vsa	12	short to supply indicator phase A	0x0: no error 0x1: Short to supply detected on phase A. The driver becomes disabled. The flags stay active, until the driver is disabled by software (T_{OFF} = 0) or by the ENN input.
SG_RESULT	9:0	StallGuard2 result, respectively, StallGuard4 result (depending on actual chopper mode) resp. PWM on time for coil A in standstill with SpreadCycle for motor temperature detection. Mechanical load measurement: The StallGuard2/4 result gives a means to measure mechanical motor load. A higher value means lower mechanical load. For StallGuard2, a value of 0 signals highest load. With optimum *SGT* setting, this is an indicator for a motor stall. The stall detection compares *SG_RESULT* to 0 to detect a stall. *SG_RESULT* is used as a base for CoolStep operation, by comparing it to a programmable upper and a lower limit. It is not applicable in StealthChop2 mode. StallGuard2 works best with microstep operation or DcStep. Temperature measurement during SpreadCycle mode: In standstill, no StallGuard2 result can be obtained. *SG_RESULT* shows the chopper on-time for motor coil A instead. Move the motor to a determined microstep position at a certain current setting to get a rough estimation of motor temperature by reading the chopper on-time. As the motor heats up, its coil resistance rises and the chopper on-time increases. For StallGuard4 specifics, see *SG4_RESULT*.	

PWMCONF (0x70)

BIT	31	30	29	28	27	26	25	24
Field	PWM_LIM[3:0]				PWM_REG[3:0]			
Reset	0xC				0x4			
Access Type	Write, Read				Write, Read			

BIT	23	22	21	20	19	18	17	16
Field	pwm_dis_reg_stst	pwm_meas_sd_enable	FREEWHEEL[1:0]		pwm_autograd	pwm_autoscale	PWM_FREQ[1:0]	
Reset	0x0	0x0	0x0		0x1	0x1	0x0	
Access Type	Write, Read	Write, Read	Write, Read		Write, Read	Write, Read	Write, Read	

BIT	15	14	13	12	11	10	9	8
Field	PWM_GRAD[7:0]							
Reset	0x0							
Access Type	Write, Read							

BIT	7	6	5	4	3	2	1	0
Field	PWM_OFS[7:0]							
Reset	0x1D							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
PWM_LIM	31:28	PWM automatic scale amplitude limit when switching on Limit for *PWM_SCALE_AUTO* when switching back from SpreadCycle to StealthChop2. This value defines the upper limit for bits 7 to 4 of the automatic current control when switching back. It can be set to reduce the current jerk during mode change back to StealthChop2. It does not limit *PWM_GRAD* or *PWM_GRAD_AUTO* offset. (Default = 12)	
PWM_REG	27:24	Regulation loop gradient User defined maximum PWM amplitude change per half wave when using *pwm_autoscale*=1. (1…15): 1: 0.5 increments (slowest regulation) 2: 1 increment 3: 1.5 increments 4: 2 increments *(Reset default)*) … 8: 4 increments ... 15: 7.5 increments (fastest regulation)	
pwm_dis_reg_stst	23	1= Disable current regulation when motor is in standstill and current is reduced (less than IRUN). This option eliminates any regulation noise during standstill.	
pwm_meas_sd_enable	22	Default = 0; 1: Uses slow decay phases on low side to measure the motor current to reduce the lower current limit.	

BITFIELD	BITS	DESCRIPTION	DECODE
FREEWHEEL	21:20	Allows different standstill modes Standstill option when motor current setting is zero (*I_HOLD* = 0). %00: Normal operation %01: Freewheeling %10: Coil shorted using LS drivers %11: Coil shorted using HS drivers	
pwm_autograd	19	PWM automatic gradient adaptation	0x0: Fixed value for *PWM_GRAD* (*PWM_GRAD_AUTO* = *PWM_GRAD*) 0x1: Automatic tuning (only with *pwm_autoscale*=1) (*Reset default*) *PWM_GRAD_AUTO* is initialized with *PWM_GRAD* while *pwm_autograd* = 0 and becomes optimized automatically during motion. Preconditions 1. *PWM_OFS_AUTO* has been automatically initialized. This requires standstill at IRUN for >130ms to a) detect standstill b) wait > 128 chopper cycles at *IRUN,* and c) regulate *PWM_OFS_AUTO* so that -1 < *PWM_SCALE_AUTO* < 1 2. Motor running and 1.5 x *PWM_OFS_AUTO x* (*IRUN*+1)/32 < *PWM_SCALE_SUM* < 4 x *PWM_OFS_AUTO x* (*IRUN*+1)/32 and *PWM_SCALE_SUM* < 255. Time required for tuning *PWM_GRAD_AUTO* About 8 fullsteps per change of +/-1. Also enables use of reduced chopper frequency for tuning *PWM_OFS_AUTO.*
pwm_autoscale	18	PWM automatic amplitude scaling	0x0: User defined feed forward PWM amplitude. The current settings *IRUN* and *IHOLD* have no influence! The resulting PWM amplitude (limited to 0…255) is: *PWM_OFS x ((CS_ACTUAL+1) / 32)* + *PWM_GRAD* x 256 / *TSTEP* 0x1: Enable automatic current control *(reset default)*
PWM_FREQ	17:16	PWM frequency selection: %00: f_{PWM}=2/1024 f_{CLK} *(Reset default)* %01: f_{PWM}=2/683 f_{CLK} %10: f_{PWM}=2/512 f_{CLK} %11: f_{PWM}=2/410 f_{CLK}	

BITFIELD	BITS	DESCRIPTION	DECODE
PWM_GRAD	15:8	Velocity dependent gradient for PWM amplitude: *PWM_GRAD* x 256/*TSTEP* This value is added to *PWM_OFS* to compensate for the velocity-dependent motor back-EMF. Use *PWM_GRAD* as initial value for automatic scaling to speed up the automatic tuning process. To do this, set *PWM_GRAD* to the determined, application specific value, with *pwm_autoscale* = 0. Only afterwards, set *pwm_autoscale* = 1. Enable StealthChop2 when finished. *Hint:* After initial tuning, the required initial value can be read out from *PWM_GRAD_AUTO*.	
PWM_OFS	7:0	User-defined PWM amplitude offset (0 to 255) related to full motor current (*CS_ACTUAL* = 31) in stand still. (*reset default = 30*). Use *PWM_OFS* as initial value for automatic scaling to speed up the automatic tuning process. To do this, set *PWM_OFS* to the determined, application specific value, with *pwm_autoscale* = 0. Only afterwards, set *pwm_autoscale* = 1. Enable StealthChop2 when finished. *PWM_OFS* = 0 disables scaling down motor current below a motor specific lower measurement threshold. This setting should only be used under certain conditions, that is, when the power supply voltage can vary up and down by a factor of two or more. It prevents the motor going out of regulation, but it also prevents power down below the regulation limit. *PWM_OFS* > 0 allows automatic scaling to low PWM duty cycles even below the lower regulation threshold. This allows low (standstill) current settings based on the actual (hold) current scale (register *IHOLD_IRUN*).	

PWM_SCALE (0x71)

Results of StealthChop2 amplitude regulator. These values can be used to monitor automatic PWM amplitude scaling (255=max. voltage).

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	PWM_SCALE_AUTO[8]
Reset	–	–	–	–	–	–	–	0x0
Access Type	–	–	–	–	–	–	–	Read Only
BIT	23	22	21	20	19	18	17	16
Field	PWM_SCALE_AUTO[7:0]							
Reset	0x0							
Access Type	Read Only							
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	PWM_SCALE_SUM[9:8]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Read Only	
BIT	7	6	5	4	3	2	1	0
Field	PWM_SCALE_SUM[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
PWM_SCALE_AUTO	24:16	
PWM_SCALE_SUM	9:0	Bits: 9...0: [0...1023]PWM_SCALE_SUM: Actual PWM duty cycle. This value is used for scaling the values CUR_A and CUR_B read from the sine wave table. 1023: maximum duty cycle. This value is extended by two bits [1,0] for higher precision of duty cycle read out. Bits 9..2 correspond to the 8 bit values in other PWM duty cycle related registers.

PWM_AUTO (0x72)

These automatically generated values can be read out in order to determine a default / power up setting for *PWM_GRAD* and *PWM_OFS*.

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	PWM_GRAD_AUTO[7:0]							
Reset	0x0							
Access Type	Read Only							

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	7	6	5	4	3	2	1	0
Field	PWM_OFS_AUTO[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
PWM_GRAD_AUTO	23:16	Automatically determined gradient value
PWM_OFS_AUTO	7:0	Automatically determined offset value

SG4_THRS (0x74)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	sg_angle_offset	sg4_filt_en
Reset	–	–	–	–	–	–	0x1	0x0
Access Type	–	–	–	–	–	–	Write, Read	Write, Read

BIT	7	6	5	4	3	2	1	0
Field	SG4_THRS[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
sg_angle_offset	9	1: Automatic phase shift compensation based on StallGuard4, when switching from StealthChop2 to SpreadCycle controlled through TPWMTHRS.
sg4_filt_en	8	1: Enable SG4 filter, 0: Disable SG4 filter
SG4_THRS	7:0	Detection threshold for stall. The StallGuard4 value *SG4_RESULT* becomes compared to this threshold. A stall is signaled with *SG4_RESULT* ≤ *SG4_THRS*. SG4_THRS covers half of the possible SG4_RESULT range.

SG4_RESULT (0x75)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	SG4_RESULT[9:8]	
Reset	–	–	–	–	–	–	0x0	
Access Type	–	–	–	–	–	–	Read Only	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SG4_RESULT[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
SG4_RESULT	9:0	StallGuard result for StallGuard4 only. *SG4_RESULT* becomes updated with each fullstep, independent of *TCOOLTHRS* and *SG4THRS*. A higher value signals a lower motor load and more torque headroom. Intended for StealthChop2 mode only. Bits 9 and 0 always show 0. Scaling to 10 bit is for compatibility to StallGuard2.

SG4_IND (0x76)

BIT	31	30	29	28	27	26	25	24
Field	SG4_IND_3[7:0]							
Reset	0x0							
Access Type	Read Only							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	SG4_IND_2[7:0]							
Reset	0x0							
Access Type	Read Only							

BIT	15	14	13	12	11	10	9	8
Field	SG4_IND_1[7:0]							
Reset	0x0							
Access Type	Read Only							

BIT	7	6	5	4	3	2	1	0
Field	SG4_IND_0[7:0]							
Reset	0x0							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
SG4_IND_3	31:24	When SG4_filt_en = 1: Displays SG4 measurement 3 used as filter input
SG4_IND_2	23:16	When SG4_filt_en = 1: Displays SG4 measurement 2 used as filter input
SG4_IND_1	15:8	When SG4_filt_en = 1: Displays SG4 measurement 1 used as filter input
SG4_IND_0	7:0	displays SG4 measurement When SG4_filt_en = 1: Displays SG4 measurement 0 used as filter input

Typical Application Circuits

Standard Application Circuit

The standard application circuit uses a minimum set of additional components. Use low ESR electrolytic capacitors to filter the power supply. The capacitors need to cope with the current ripple caused by chopper operation. A minimum capacity of 100µF at V_S is recommended for best performance. Current ripple in the supply capacitors also depends on the power supply internal resistance and cable length. V_{CC_IO} must be supplied from an external source, example, a low-drop 3.3V regulator.

Place all filter capacitors as close as possible to the related IC pins. Use a solid common ground plane for all GND connections. Connect V_{DD1V8} filtering capacitor directly to the V_{DD1V8} pin.

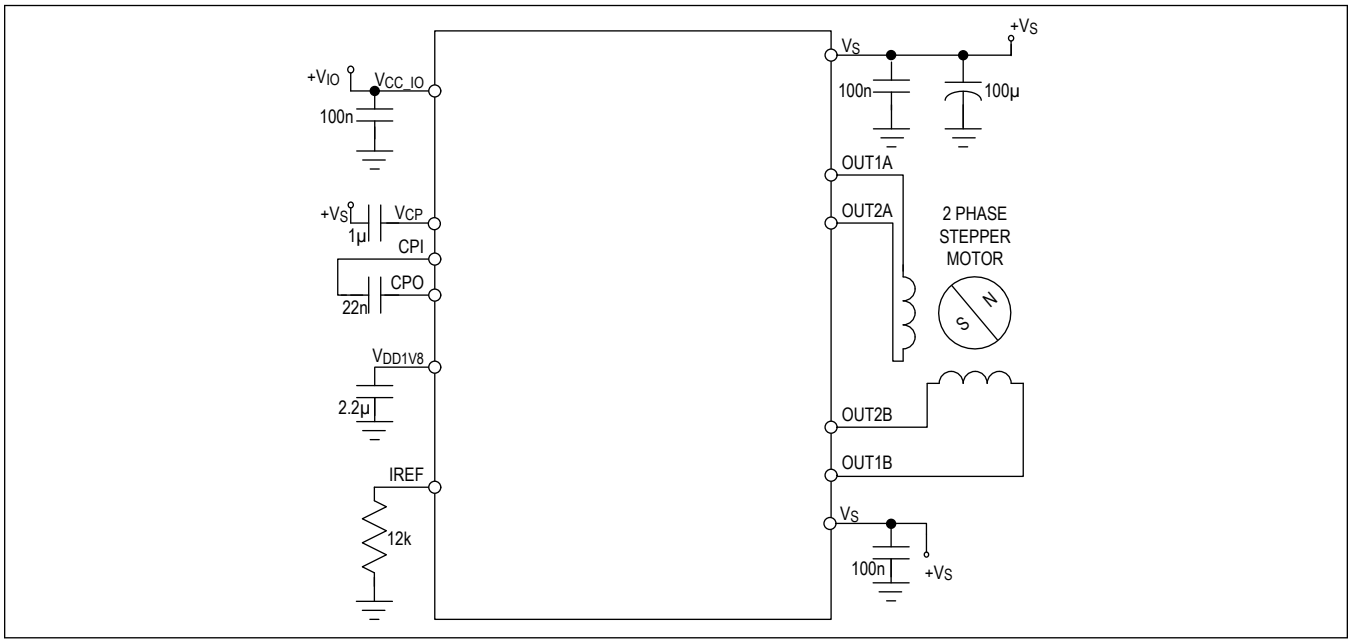


Figure 49. Standard Application Circuit

High Motor Current

When operating at a high motor current, the driver power dissipation due to MOSFET switch-on resistance significantly heats up the driver. This power dissipation heats up the PCB cooling infrastructure also, if operated at an increased duty cycle. This in turn leads to a further increase of driver temperature. An increase of temperature by about 100°C increases MOSFET resistance by roughly 50%. This is a typical behavior of MOSFET switches. Therefore, under high duty cycle, high load conditions, thermal characteristics have to be carefully taken into account, especially when increased environment temperatures are to be supported. See *Package Information* for the thermal characteristics and the online evaluation kit information for the layout example.

As a rule of thumb, thermal properties of the PCB design may become critical at or above 1.5A RMS motor current for increased periods of time. Note that the resistive power dissipation raises with the square of the motor current. On the other hand, this means that a small reduction of motor current significantly saves heat dissipation and energy.

Driver Protection and EME Circuitry

Some applications have to cope with ESD events caused by motor operation or external influence. Despite ESD

Typical Application Circuits (continued)

circuitry within the driver chips, ESD events occurring during operation can cause a reset or even a destruction of the motor driver, depending on their energy. Especially, plastic housings and belt drive systems tend to cause ESD events of several kV. It is best practice to avoid ESD events by attaching all conductive parts, especially the motors themselves to PCB ground, or to apply electrically conductive plastic parts. In addition, the driver can be protected up to a certain degree against ESD events or live plugging/pulling the motor, which also causes high voltages and high currents into the motor connector terminals.

A simple scheme uses capacitors at the driver outputs to reduce the dV/dt caused by ESD events. Larger capacitors 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 application PCB circuitry and thus reduce electromagnetic emission.

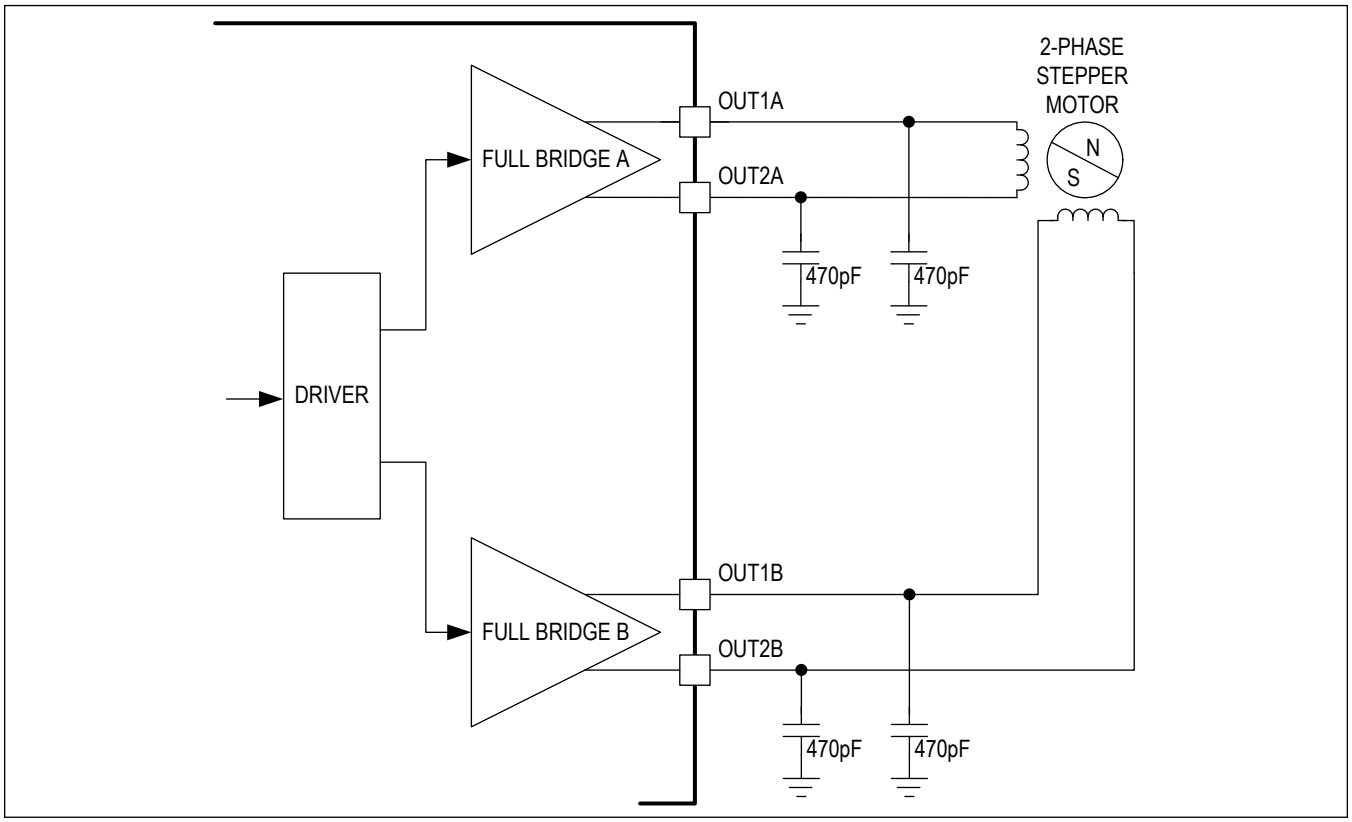


Figure 50. Simple ESD Enhancement

A more elaborate scheme uses LC filters to decouple the driver outputs from the motor connector. Varistors V1 and V2 in between of the coil terminals eliminate coil overvoltage caused by live plugging. Optionally, protect all outputs by a varistor (V1A, V1B, V2A, V2B) against the ESD voltage. Fit the varistors to the supply voltage rating. The SMD inductivities conduct full motor coil current and need to be selected accordingly.

Typical Application Circuits (continued)

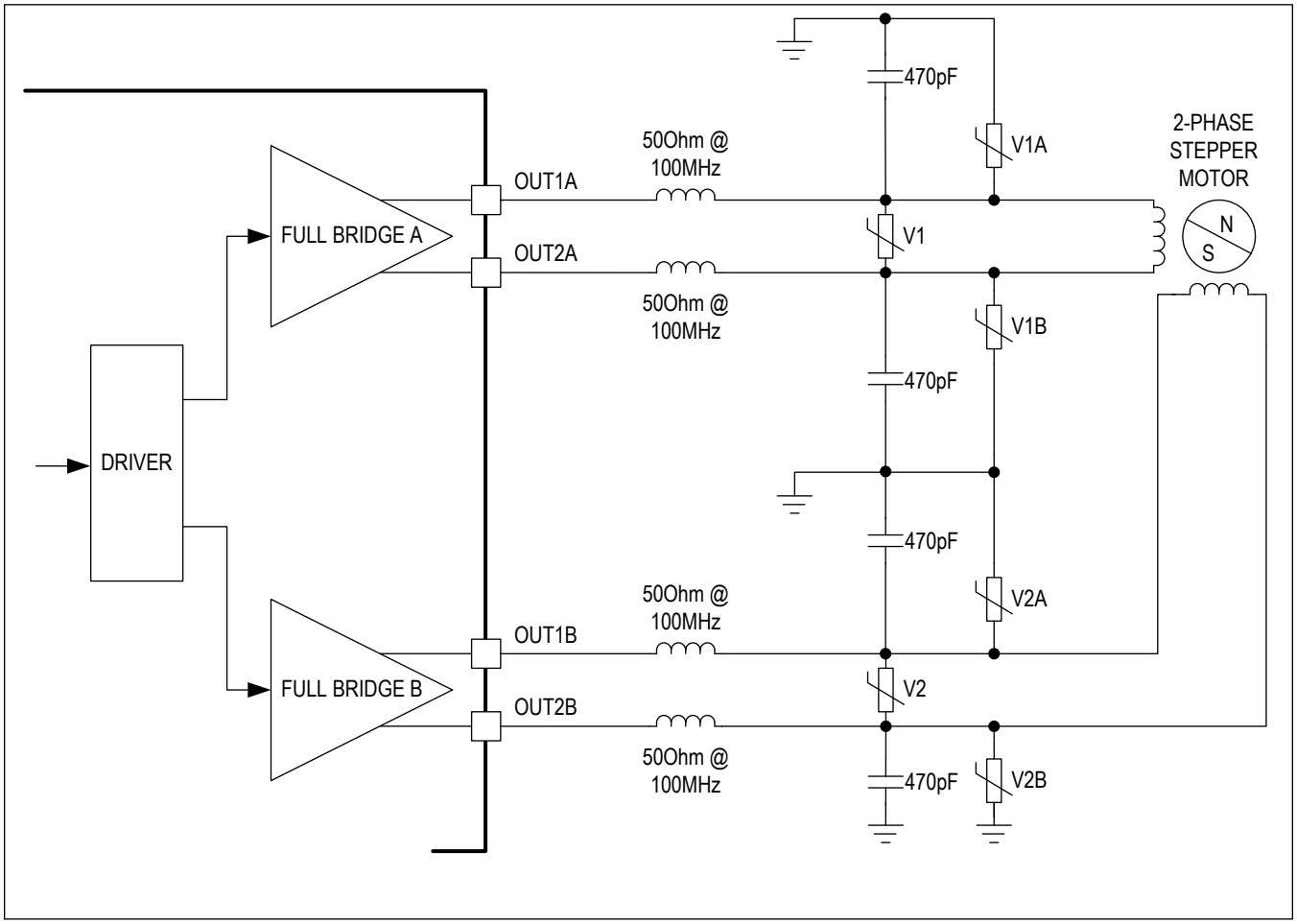


Figure 51. Extended Motor Output Protection

Ordering Information

PART NUMBER	TEMPERATURE RANGE	PIN-PACKAGE
TMC5240ATJ+	-40°C to +125°C	32 TQFN - 5mm x 5mm
TMC5240ATJ+T	-40°C to +125°C	32 TQFN - 5mm x 5mm
TMC5240AUU+	-40°C to +125°C	38 TSSOP-EP 4.4mm x 9.7mm
TMC5240AUU+T	-40°C to +125°C	38 TSSOP-EP 4.4mm x 9.7mm

+ Denotes a lead(Pb)-free/RoHS-compliant package.

T Denotes tape-and-reel.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	12/22	Release for market intro	—
1	4/24	Updated *Open Load Flags, added Quick Configuration Guide,* additional calculations for for R_{REF} = 16kΩ, 24kΩ, and 48kΩ, updated description of SEIMIN, added note/ description to *Tuning CoolStep, added explanation on Overvoltage Protection and OV Pin, d*escription corrected from 2^16 to 2^18 in register D1, updated/extended description in register COLLCONF, fixed typos and formatting errors across the document.	38, 62, 47, 69, 70, 80, 121, 154