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

TMC2240

36V $2A_{RMS}$+ Smart Integrated Stepper Driver with S/D and SPI

General Description

The TMC2240 is a smart high-performance stepper motor driver IC with serial communication interfaces (SPI, UART) and extensive diagnosis capabilities.

It combines industries' most advanced stepper motor driver based on the 256 microsteps built-in indexer and two fully integrated 36V, $3.0A_{MAX}$ H-Bridges plus non-dissipative integrated current sensing (ICS).

ADI-Trinamic's 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 while giving best in class performance.

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

The maximum RMS current per H-Bridge is I_{RMS} = $2.1A_{RMS}$ at room temperature assuming a 4-layers PCB.

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 resistor.

The TMC2240 features abundant diagnostics and protections such as short protection/OCP, thermal shutdown, undervoltage lockout (UVLO).

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

Furthermore, the TMC2240 provides functions to measure the driver temperature, estimate the motor temperature, and measure one external analog input.

The TMC2240 is available in a small TQFN32 5mm x 5mm package and a thermally optimized TSSOP38 9.7mm x 4.4mm with 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
- POS, Massage Chairs
- ATM, Cash Recycler, Bill Validators, Cash Machines
- 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): 230mΩ Typical (T_A = 25ºC)
- Current Ratings per H-Bridge (Typical at 25ºC):
  - I_{MAX} = 5.0A (Bridge Peak Current)
  - I_{RMS} = $2.1A_{RMS}$ (3A Sine Wave Peak)
- Fully Integrated Lossless Current Sensing
- Step/Direction (S/D) Interface with MicroPlyer Step Interpolation
- SPI and Single Wire UART
- Incremental Encoder Interface
- 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 and 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.

19-101548; Rev 0; 7/22

Simplified Block Diagram

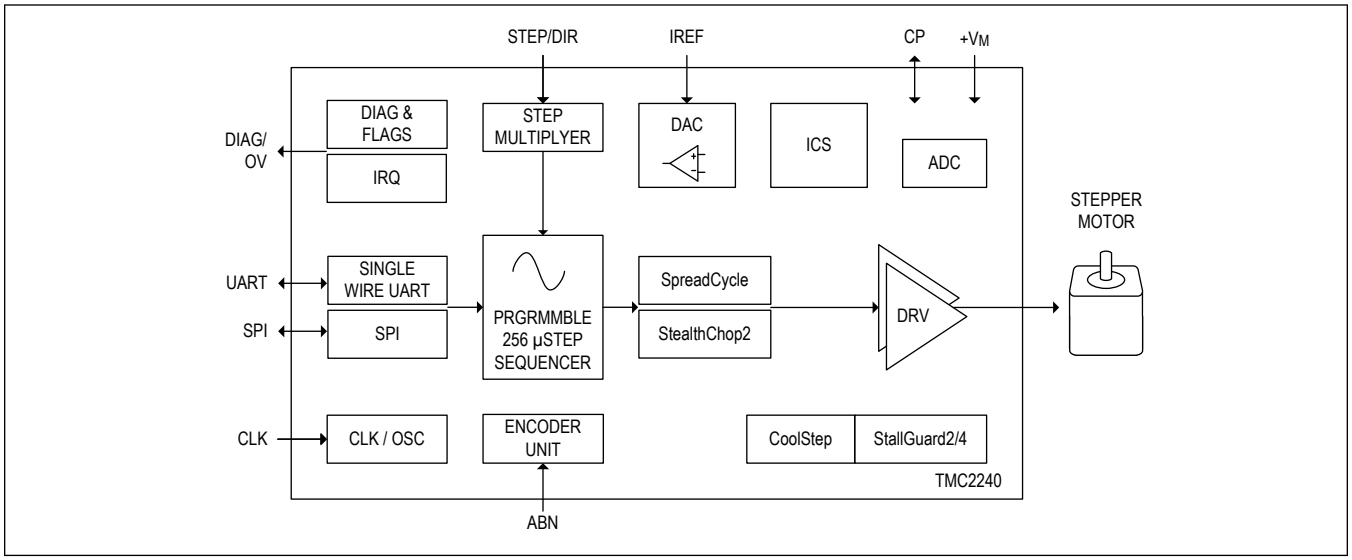


TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF FIGURES

LIST OF TABLES

Absolute Maximum Ratings

V_S to GND ..... -0.3V to 41V
V_{DD} 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_{DD} + 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 ..... +160°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. Specifications marked "GBD" are guaranteed by design and not production tested.)

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. Specifications marked "GBD" are guaranteed by design and not production tested.)

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}	V(SLEEPN) = 0		5	10	µA
Quiescent Current Consumption	I_{VCC}	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
Pullup/Pulldown Resistance	R_{PD}	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 Pull-down 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. Specifications marked "GBD" are guaranteed by design and not production tested.)

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 VS	UVLO	V_S falling	3.75	3.9	4.05	V
UVLO Threshold on VS Hysteris	UVLOHYS			0.12		V
UVLO Threshold on VCC_IO	UVLO	V_{CC_IO} falling	0.9	1.5	1.95	
VCC_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	RREF		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
Current Trip Regulation Accuracy	DITRIP1	ITRIG 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. Specifications marked "GBD" are guaranteed by design and not production tested.)

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}		12	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
STEP/DIR TIMINGS						
Step Frequency	f_{STEP}	dedge = 1			f_{CLK}/8	
		dedge = 0			f_{CLK}/4	
Fullstep Frequency	f_{FS}				f_{CLK}/512	
STEP High Time	t_{SH}		tCLK + 20			ns
STEP Low Time	t_{SL}		tCLK + 20			ns
DIR/STEP to CLK Setup Time	t_{SU}		10			ns
DIR/STEP to CLK Hold Time	t_{SH}		10			ns

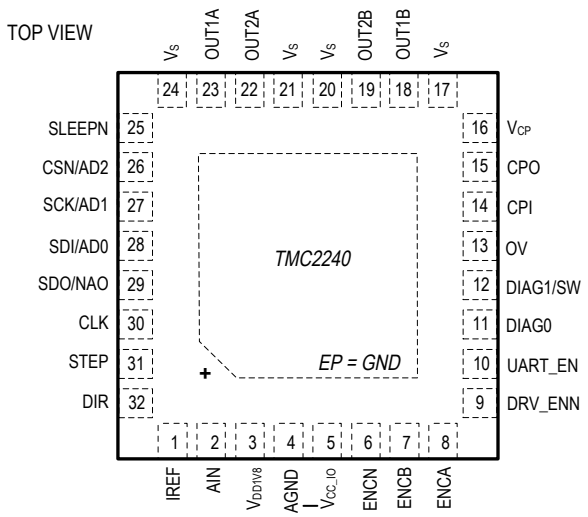
Electrical Characteristics (continued)

(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. Specifications marked "GBD" are guaranteed by design and not production tested.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
DIR to STEP Setup Time	t_{SU}		20			ns
DIR to STEP HoldTime	t_H		20			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				13		Bit
Analog Input Voltage Range	V_{AIN}		0		1.25	V
Analog Input Leakage	$I_{AIN,leak}$		-1		+1	uA
Analog Input Frequency	f_{AIN}				70	kHz
Driver Temperature Accuracy	T_{DRIVER}			±10		°C
Supply Voltage Measurement Accuracy			-5		+5	%
ADC Sample Rate	$f_{SAMPLE, ADC}$			$f_{CLK} / 2048$		

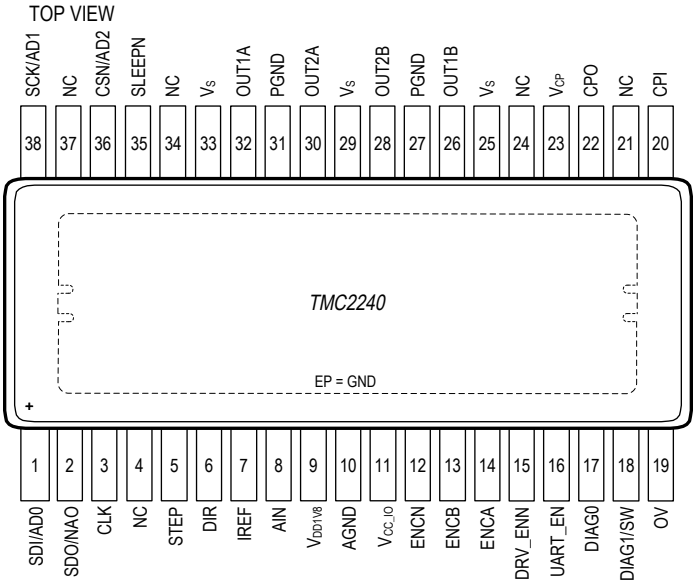
Pin Configurations

TMC2240 TQFN Pin Configuration



TQFN
5mm x 5mm

TMC2240 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
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.		Output
15	22	CPO	Charge pump capacitor output.		Output
14	20	CPI	Charge pump capacitor input. Tie to CPO using 22nF 50V capacitor.		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}	DI
31	5	STEP	STEP input.	V_{CC_IO}	DI
32	6	DIR	Direction input.	VCC_IO	DI
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}	DI (pd)
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}	DI (pd)
28	1	SDI/AD0	SPI data input (UART_EN = 0) or address input 0 (+1) for single wire interface (UART_EN = 1).	V_{CC_IO}	DI (pd)
29	2	SDO/NAO	SPI data output (three-state) (UART_EN = 0) or next address output (NAO) for single wire interface (UART_EN = 1).	V_{CC_IO}	DIO (pd)
1	7	IREF	Analog reference current for current scaling. Provide external resistor to GND.	V_{CC_IO}	AI
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}	DI (pd)
7	13	ENCB	Encoder B-channel input.	V_{CC_IO}	DI
8	14	ENCA	Encoder A-channel input.	V_{CC_IO}	DI
6	12	ENCN	Encoder N-channel input.	V_{CC_IO}	DI
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}	DI (pu)

Pin Description (continued)

PIN		NAME	FUNCTION	REF SUPPLY	TYPE
TQFN32	TSSOP38				
11	17	DIAG0	Diagnostics output DIAG0. 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}	DO
12	18	DIAG1/SW	Diagnostics output DIAG1. Use external pullup resistor in open drain mode. Single wire I/O in UART mode.	V_{CC_IO}	DIO
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-up. 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. All TMC2240 pullup/pulldown inputs become switched to level holding mode, to avoid current draw at V_{CC_IO}. Do not use while at high motor velocity!	V_S	AI (pd)
19	28	OUT2B	Motor coil B output 2	V_S	A
18	26	OUT1B	Motor coil B output 1	V_S	A
22	30	OUT2A	Motor coil A output 2	V_S	A
23	32	OUT1A	Motor coil A output 1	V_S	A
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.

Pin Description (continued)

PIN		NAME	FUNCTION	REF SUPPLY	TYPE
TQFN32	TSSOP38				
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}	DO (OD)
2	8	AIN	General-purpose analog input measured by internal 12-bit ADC with $f_{CLK} / 2048$. Input range 0 to 1.25V. Value available via SPI/UART.	V_{CC_IO}	AI
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}	AI

Functional Diagrams

TMC2240

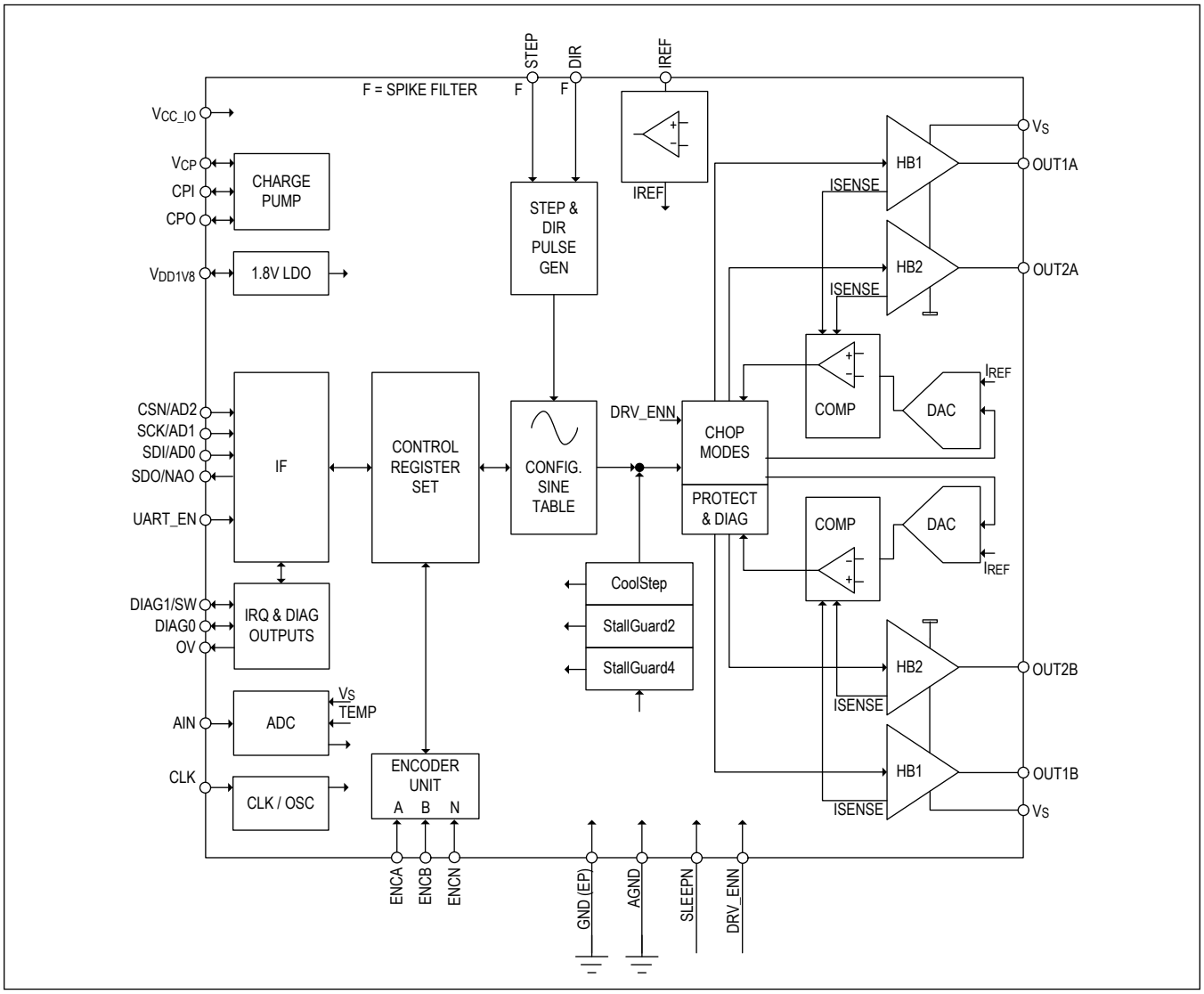


Figure 1. Block Diagram

Detailed Description

Principles of Operation

Step and Direction Driver with Serial Interface and Diagnostic Feedback

The TMC2240 is a smart Step and Direction stepper motor driver with serial interface (SPI, UART) for parameterization and monitoring & diagnostics.

An external high-performance motion controller like the TMC4361A or a CPU generates step and direction signals synchronized to other components like additional motors within the system. The TMC2240 takes care of intelligent current control and provides feedback on the state of the motor via one of its serial interfaces.

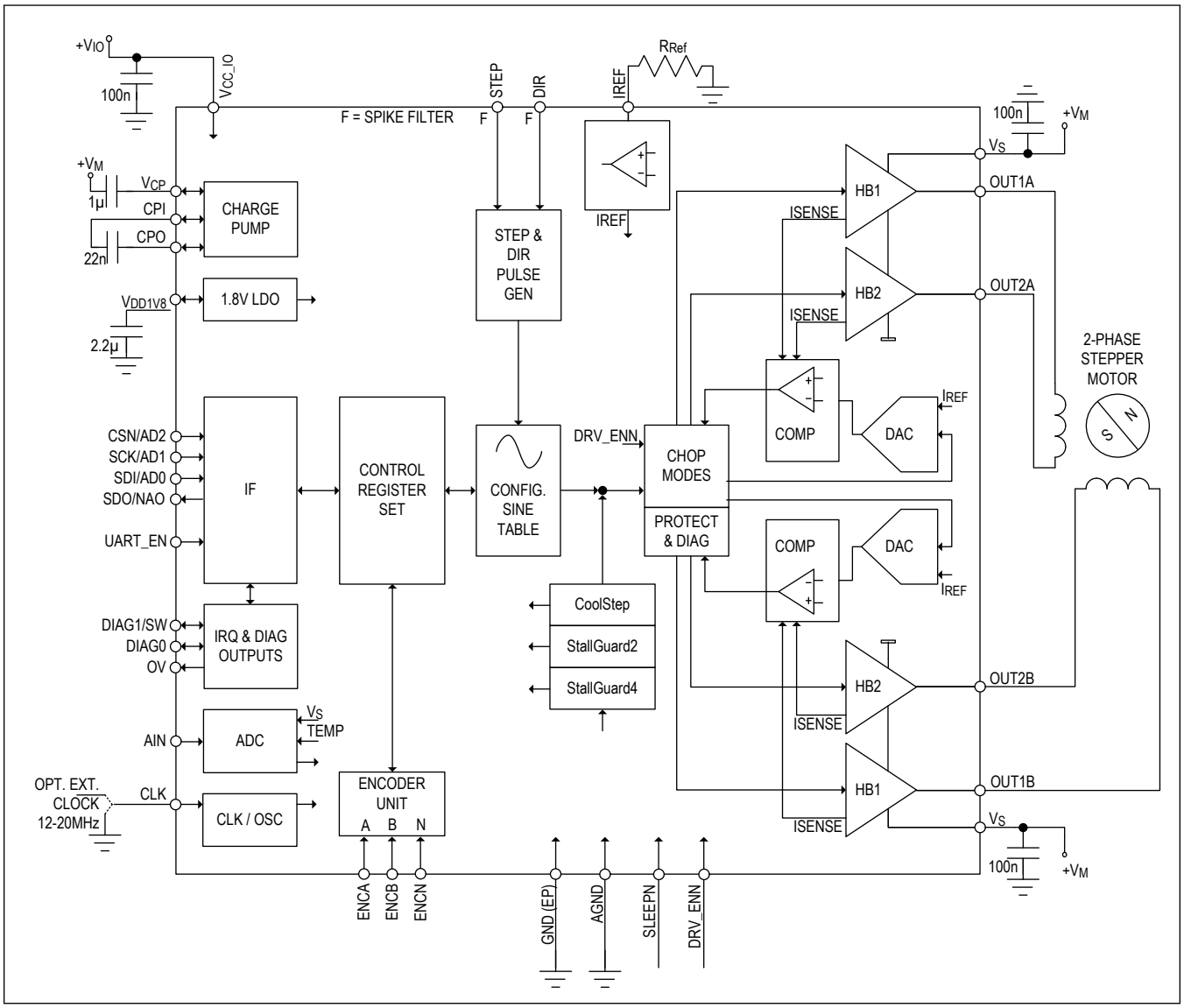


Figure 2. Block Diagram with Typical External Components

Key Concepts

The TMC2240 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 standstill of the motor
SpreadCycle	High-precision cycle-by-cycle current control for highest dynamic movements
StallGuard2	Sensorless stall detection and mechanical load measurement
StallGuard4	Sensorless homing safes end switches and warns in case of motor overload
CoolStep	Active peak current control based on StallGuard feedback for best efficiency and lowest motor and driver temperature
MicroPlyer	Microstep interpolator to run at full 256 microstepping with low resolution step input

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 to enhance safety and recovery from equipment malfunctions.

Control Interfaces

The TMC2240 supports both, an SPI interface 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 interface is a bit-serial interface synchronous to a bus clock. For every bit sent from the bus master to the bus slave another bit is sent simultaneously from the slave to the master. Communication between an SPI master and a TMC2240 slave 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.

Step and Direction Interface

The motor is controlled using a step and a direction input. Active edges on the STEP input can be rising edges or both rising and falling edges as controlled by mode bit (*dedge*). Using both edges cuts the toggle rate of the STEP signal in half, which is useful for control over slow interfaces such as optically isolated couplers. On each active edge, the state sampled from the DIR input determines whether to step forward or back. Each step can be a fullstep or a microstep, in which there are 2, 4, 8, 16, 32, 64, 128, or 256 microsteps per fullstep. A step impulse with a low state on DIR increases the microstep counter and a high state decreases the counter by an amount controlled by the microstep resolution. An internal table translates the counter value into the sine and cosine values which control the motor current for microstepping.

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 via the serial control interfaces.

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

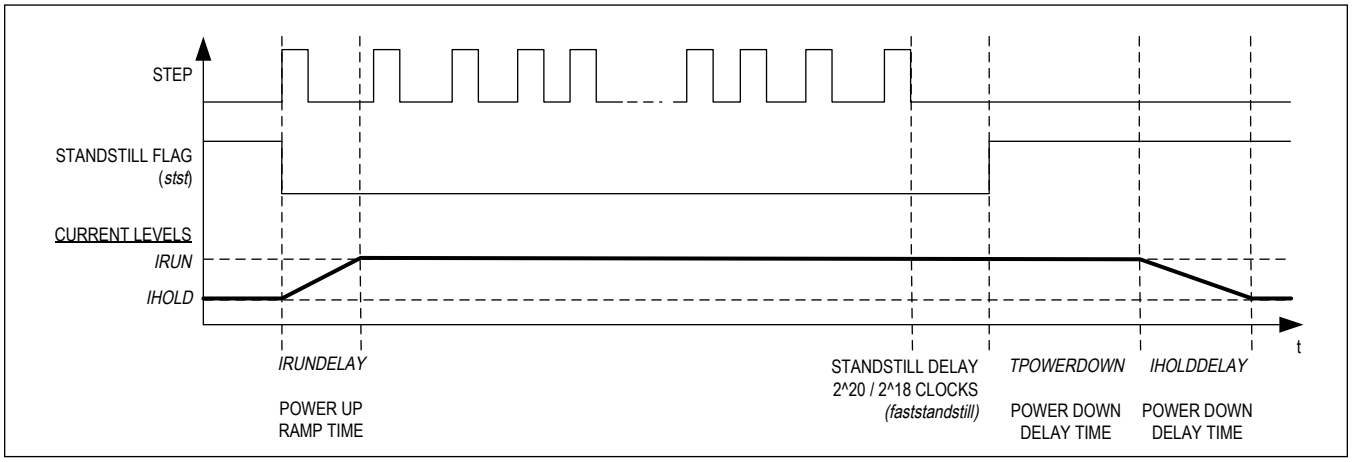


Figure 3. Automatic Motor Current Control at Standstill and Ramp-up

StealthChop2 and SpreadCycle Driver

StealthChop is a voltage-chopper based principle. It especially guarantees that the motor is absolutely quiet in standstill and in slow motion, except for noise generated by ball bearings. Unlike other voltage mode choppers, StealthChop2 does not require any configuration. It automatically learns the best settings during the first motion after power up and further optimizes the settings in subsequent motions. An initial homing sequence is sufficient for learning. Optionally, initial learning parameters can be 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 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

StallGuard – Mechanical Load Sensing

StallGuard2 provides an accurate measurement of the load on the motor. It can be used for stall detection as well as other uses at loads below those which stall the motor, such as CoolStep load-adaptive current reduction. This gives more information on the drive allowing functions like sensorless homing and diagnostics of the drive mechanics. 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. This saves energy and keeps the components cool. Due to driving the motor with the optimum current, CoolStep increases the motor efficiency compared to standard operation with ca. 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 TMC2240 provides an encoder interface for external incremental encoders. The encoder can be used for consistency checks on-the-fly between encoder position and external 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 Interface

SPI Datagram Structure

The TMC2240 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 8 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 via 32 data bits even if it uses less than 32 data bits.

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

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

All registers are readable, most of them are read write, some read only and some write 1 to clear (e.g., 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 interface always delivers data back to the master, independent of the W bit. The data transferred back is the data read from the address which was transmitted with the *previous* datagram, if the previous access was a read access. If the previous access was a write access, then the data read back mirrors the previously received write data. So, the difference between a read and a write access is that the read access does not transfer data to the addressed register but it transfers the address only and its 32 data bits are dummies, and, further the following read or write access delivers back the data read from the address transmitted in the preceding read cycle.

A read access request datagram uses dummy write data. Read data is transferred back to the master with the

subsequent read or write access. Hence, reading multiple registers can be done in a pipelined fashion.

Whenever data is read from or written to the TMC2240, 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	DATA RECEIVED FROM
read *XACTUAL*	0x2100000000	0xSS & unused data*
read *XACTUAL*	0x2100000000	0xSS & XACTUAL
write *VMAX* = 0x00ABCDEF	0xA700ABCDEF	0xSS & 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
7:4	*don't care*	not used in TMC2240
3	*standstill*	*DRV_STATUS*[31] – 1: Signals motor stand still
2	*sg2*	*DRV_STATUS*[24] – 1: Signals StallGuard flag active
1	*driver_error*	*GSTAT*[1] – 1: Signals driver 1 driver error (clear by reading *GSTAT*)
0	*reset_flag*	*GSTAT*[0] – 1: Signals, that a reset has occurred (clear by reading *GSTAT*)

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 TMC2240 has four signals:

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

The slave is enabled for an SPI transaction by a low on the chip select input CSN. Bit transfer is synchronous to the bus clock SCK, with the slave latching the data from SDI on the rising edge of SCK and driving data to SDO following the falling edge. The most significant bit is sent first. A minimum of 40 SCK clock cycles is required for a bus transaction with the TMC2240.

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 master to the slave. If more than 40 bits are sent, only the last 40 bits received before the rising edge of CSN are recognized as the command.

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 figure shows the timing parameters of an SPI bus transaction. Timing values are given in the EC table.

The SPI interfaces use SPI MODE 3.

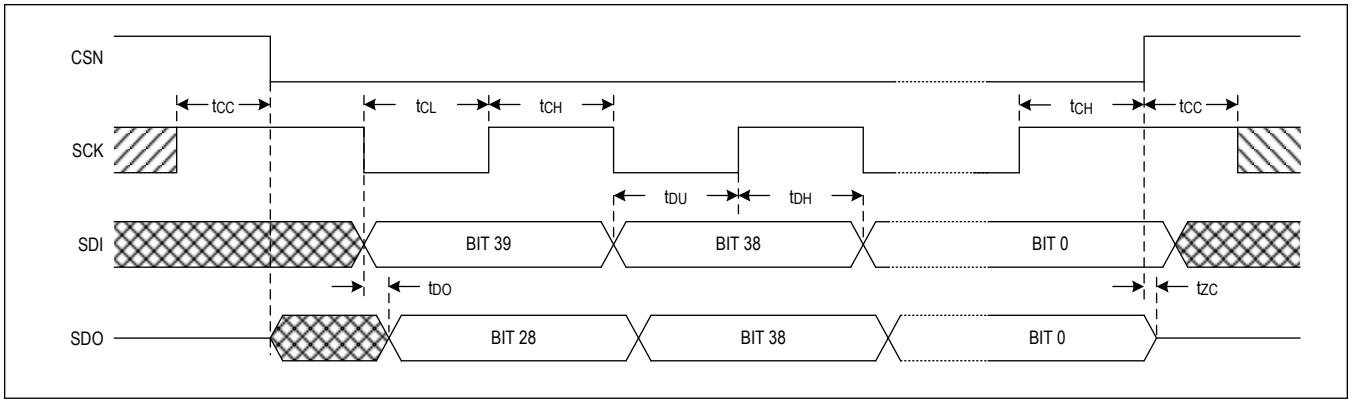


Figure 4. SPI Timing Diagram

UART Single Wire Interface

The UART single wire interface allows control of the TMC2240 with any microcontroller UART. It shares transmit and receive line like an RS485-based interface. Data transmission is secured using a cyclic redundancy check, so that increased interface distances (e.g., over cables between two PCBs) can be bridged without 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 slave address			RW + 7 bit register addr.			32 bit data			CRC		
0…7								8…15			16…23			24…55			56…63		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*			register address		1	data bytes 3, 2, 1, 0 (high to low byte)			CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	55	56	…	63

A sync nibble precedes each transmission to and from the TMC2240 and is embedded into the first transmitted byte, followed by an addressing byte. Each transmission allows a synchronization of the internal baud rate divider to the master clock. The actual baud rate is adapted and variations of the internal clock frequency are compensated. Thus, the baud rate can be freely chosen within the valid range. Each transmitted byte starts with a start bit (logic 0, low level on 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 slave address *SLAVEADDR* is selected by CSN_AD2, SCK_AD1, SDI_AD0 in the range 0 to 7.

The slave address is determined by the sum of the register *SLAVEADDR* and the pin selection given above. This means, that a high level on SDI (with CSN low and SCK low) increments the *SLAVEADDR* 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 master to reset communication in case of transmission errors. Any pulse on an idle data line below 16 clock cycles will be treated as a glitch and leads to a timeout of 12 bit times, for which the data line must be idle. Other errors like wrong CRC are also treated the same way. This allows a safe resynchronization of the transmission after any error conditions. Remark, 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 master 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 slave address			RW + 7 bit register address			CRC		
0...7								8…15			16…23			24…31		
1	0	1	0	Reserved (don't cares but included in CRC)				*SLAVEADDR*			register address		0	CRC		
0	1	2	3	4	5	6	7	8	…	15	16	…	23	24	…	31

The read access request datagram structure is identical to the write access datagram structure, but uses a lower number of user bits. Its function is the addressing of the slave and the transmission of the desired register address for the read access. The TMC2240 responds with the same baud rate as the master uses for the read request.

In order to ensure a clean bus transition from the master to the slave, the TMC2240 does not immediately send the reply to a read access, but it uses a programmable delay time after which the first reply byte becomes sent following a read request. This delay time can be set in multiples of eight bit times using *SENDDELAY* time setting (default = 8 bit times) according to the needs of the master. In a multi-slave system, set *SENDDELAY* to min. 2 for all slaves. Otherwise a non-addressed slave might detect a transmission error upon read access to a different slave.

Table 6. UART Read Access Reply Datagram Structure

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

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

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

CRC Calculation

An 8-bit CRC polynomial is used for checking both read and write access. It allows detection of up to eight single bit errors. The CRC8-ATM polynomial with an initial value of zero is applied LSB to MSB, including the sync- and addressing byte. The sync nibble is assumed to always be correct. The TMC2240 responds only to correctly transmitted datagrams

containing its own slave address. It increases its datagram counter for each correctly received write access datagram.

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

Serial calculation example

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

C-Code Example for CRC calculation

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

  *crc = 0;

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

UART Signals

The UART interface on the TMC2240 comprises five signals. In UART mode, the slave 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. TMC2240 UART Interface Signals

SIGNAL	DESCRIPTION
DIAG1/SW	Data input and output
CSN/AD2	Bit 2 of initial UART address increment (+4)
SCK/AD1	Bit 1 of initial UART address increment (+2)
SDI/AD0	Bit 0 of initial UART address increment (+1), tie to NAO of previous IC in chain
SDO/NAO	Next address output (NAO) pin for chained sequential addressing scheme (reset default = high)

Addressing Multiple Slaves

If only one or up to eight TMC2240 are addressed by a master using a single UART bus interface, a simple hardware address selection can be used. The individual UART addresses are set by connecting the UART address pins (SDI, SCK, CSN) to V_{CC_IO} and GND.

If more than eight slaves need to be connected to the same UART bus, then a different approach must be used. This

approach can address up to 255 devices 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 your first TMC2240 to GND.
- Connect SDO/NAO output of the first TMC2240 to the next drivers address[0] pin (SDI/AD0). Connect further drivers in the same fashion.
- Now, the first driver responds to address 0. Following drivers are set to address 1.
- Program the first driver to its dedicated slave address. **Note**: Once a driver is initialized with its slave address, its SDO/NAO output which is tied to the next drivers address[0] pin (SDI/AD0) has to be programmed to logic 0 in order to differentiate the next driver from all following devices.
- Now, the second driver is accessible and can get its slave address. Further units can be programmed to their slave addresses sequentially.

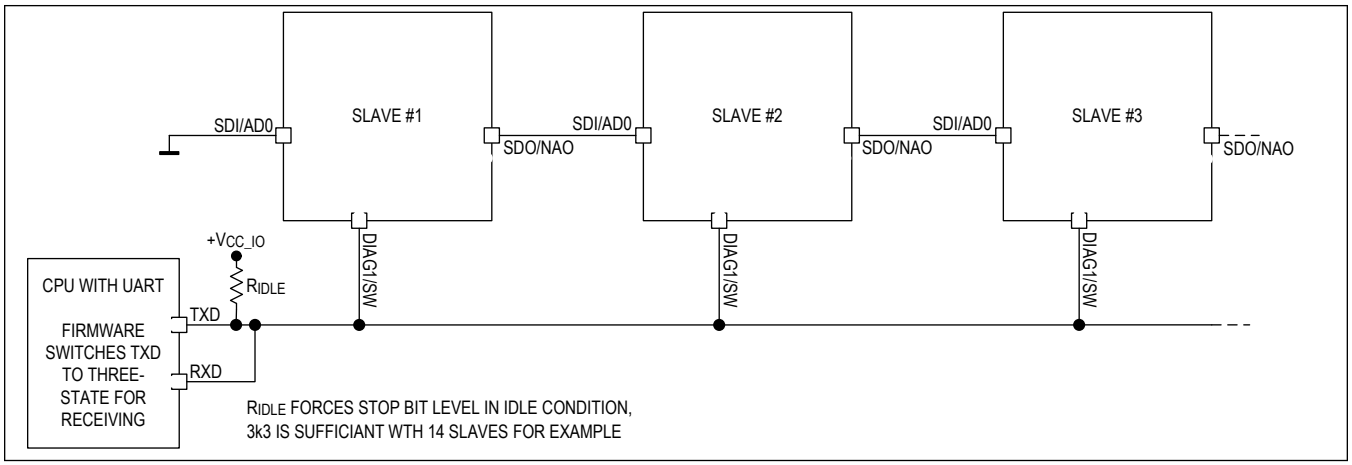


Figure 5. UART Daisy Chaining Example

Table 8. UART Example for Addressing up to 255 Slaves

PHASE	SLAVE #1	SLAVE #2	SLAVE #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 & set NAO low	address 0
Addressing phase 4	address 254	address 253	program to address 252 & set NAO low
Addressing phase x	continue procedure		

Step/Direction Interface

The STEP and DIR inputs provide a simple, standard interface compatible with many existing motion controllers. The MicroPlyer step pulse interpolator brings the smooth motor operation of high-resolution microstepping to applications originally designed for coarser stepping.

Timing

The figure below shows the timing parameters for the STEP and DIR signals. When the *dedge* mode bit in the *CHOPCONF* register is set, both edges of STEP are active. If *dedge* is cleared, only rising edges are active. STEP and DIR are sampled and synchronized to the system clock. An internal analog filter of ca. 10ns removes glitches on the

signals, such as those caused by long PCB traces. If the signal source is far from the chip, and especially if the signals are carried on cables, the signals should be filtered or transmitted differentially.

See the *Electrical Characteristics* table for the specified timing parameters.

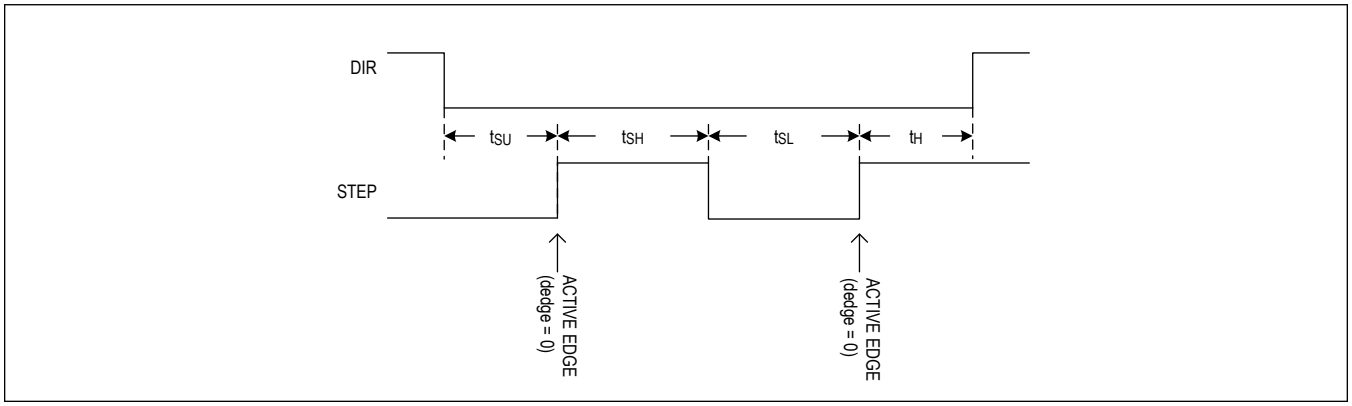


Figure 6. STEP/DIR Signal Timing

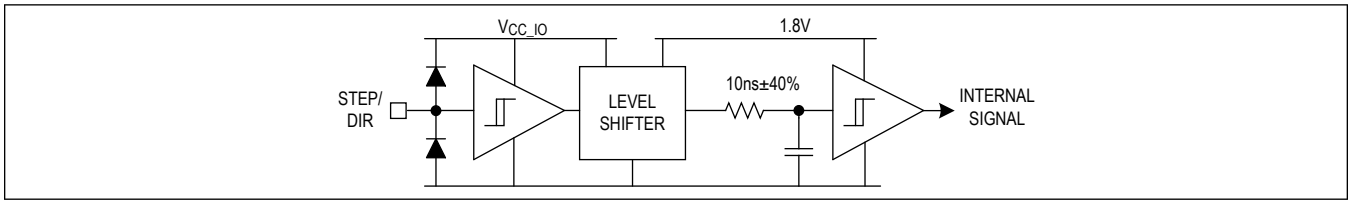


Figure 7. STEP/DIR Signal Input Filter Structure

Changing Resolution

A reduced microstep resolution allows limitation of the step frequency for the STEP/DIR interface, or compatibility to an older, less performing driver. The internal microstep table with 1024 sine wave entries generates sinusoidal motor coil currents. These 1024 entries correspond to one electrical revolution or four fullsteps. The microstep resolution setting determines the step width taken within the table. Depending on the DIR input, the microstep counter is increased (DIR = 0) or decreased (DIR = 1) with each STEP pulse by the step width. The microstep resolution determines the increment respectively the decrement. At maximum resolution, the sequencer advances one step for each step pulse. At half resolution, it advances two steps. Increment is up to 256 steps for fullstepping. The sequencer has special provision to allow seamless switching between different microstep rates at any time. When switching to a lower microstep resolution, it calculates the nearest step within the target resolution and reads the current vector at that position. This behavior especially is important for low resolutions like fullstep and halfstep, because any failure in the step sequence would lead to asymmetrical run when comparing a motor running clockwise and counterclockwise.

Examples:

Fullstep: Cycles through table positions: 128, 384, 640, and 896 (45°, 135°, 225°, and 315° electrical position, both coils on at identical current). The coil current in each position corresponds to the RMS-Value (0.71 x amplitude). Step size is 256 (90° electrical)

Half step: The first table position is 64 (22.5° electrical), Step size is 128 (45° steps)

Quarter step: The first table position is 32 (90°/8 = 11.25° electrical), Step size is 64 (22.5° steps)

This way equidistant steps result and they are identical in both rotation directions. Some older drivers also use zero current (table entry 0, 0°) as well as full current (90°) within the step tables. This kind of stepping is avoided because it provides less torque and has a worse power dissipation in driver and motor.

Table 9. Fullstep/Half Step Lookup Table Values for Phase A/B Coil Currents

STEP POSITION	TABLE POSITION	CURRENT COIL A	CURRENT COIL B
Half step 0	64	38.3%	92.4%
Fullstep 0	128	70.7%	70.7%
Half step 1	192	92.4%	38.3%
Half step 2	320	92.4%	-38.3%
Fullstep 1	384	70.7%	-70.7%
Half step 3	448	38.3%	-92.4%
Half step 4	576	-38.3%	-92.4%
Fullstep 2	640	-70.7%	-70.7%
Half step 5	704	-92.4%	-38.3%
Half step 6	832	-92.4%	38.3%
Fullstep 3	896	-70.7%	70.7%
Half step 7	960	-38.3%	92.4%

MicroPlyer Step Interpolator and Standstill Detection

For each active edge on STEP, MicroPlyer produces microsteps at 256x resolution. It interpolates the time in between the two step impulses at the step input based on the last step interval. This way, from 2 microsteps (128 microsteps to 256 microsteps interpolation) up to 256 microsteps (fullstep input to 256 microsteps) are driven for a single-step pulse.

MicroPlyer function is enabled by setting the *intpol* bit in the *CHOPCONF* register.

The step rate for the interpolated 2 microsteps to 256 microsteps is determined by measuring the time interval of the previous step period and dividing it into up to 256 equal parts. The maximum time between 2 microsteps corresponds to 2^{20} (roughly one million system clock cycles), for an even distribution of 256 microsteps. At 16MHz system clock frequency, this results in a minimum step input frequency of 16Hz for MicroPlyer operation. A lower step rate causes the *STST* bit to be set, which indicates a standstill event. At that frequency, microsteps occur at a rate of (system clock frequency)/2^{16} ~ 256Hz. When a standstill is detected, the driver automatically switches the motor to holding current *IHOLD*.

Attention: MicroPlyer only works perfectly with a stable STEP frequency. Do not use the *dedge* option if the STEP signal does not have a 50% duty cycle!

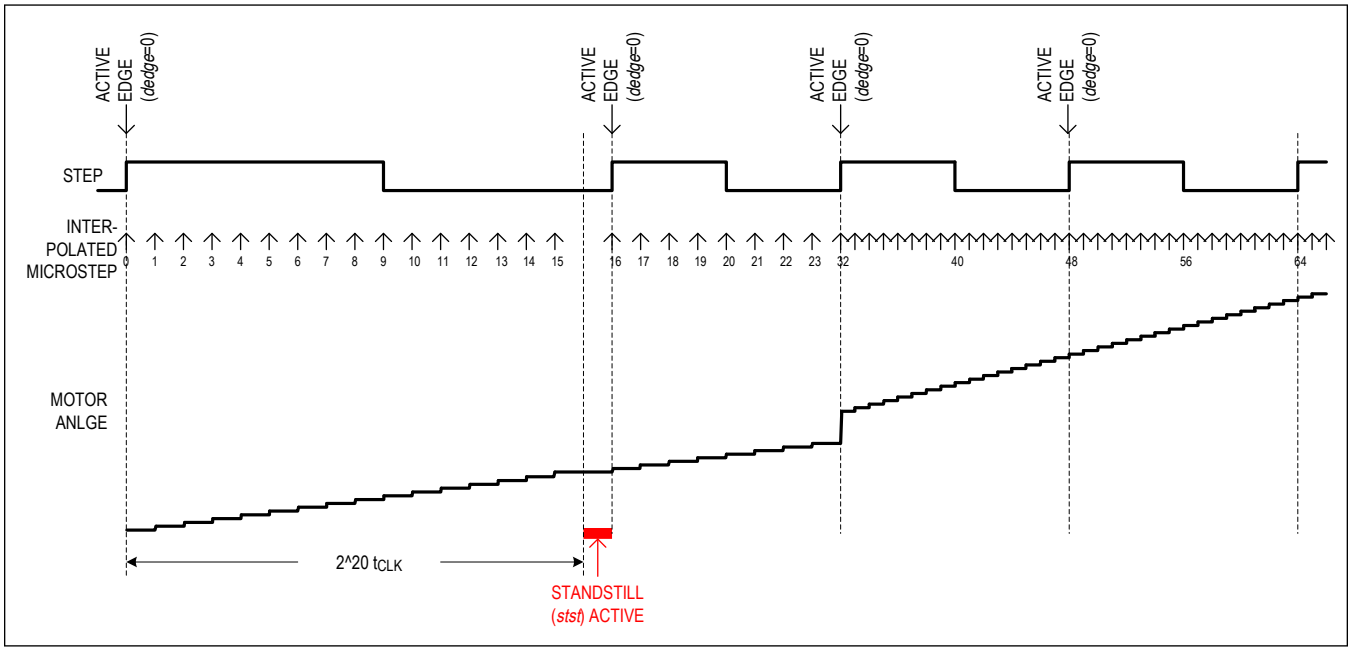


Figure 8. MicroPlyer Microstep Interpolation with Rising STEP Frequency (Example: 16 to 256)

In the figure, the first STEP cycle is long enough to set the standstill bit *stst*. This bit is cleared on the next STEP active edge. Then, the external STEP frequency increases. After one cycle at the higher rate MicroPlyer adapts the interpolated microstep rate to the higher frequency. During the last cycle at the slower rate, MicroPlyer did not generate all 16 microsteps, so there is a small jump in motor angle between the first and second cycles at the higher rate.

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.

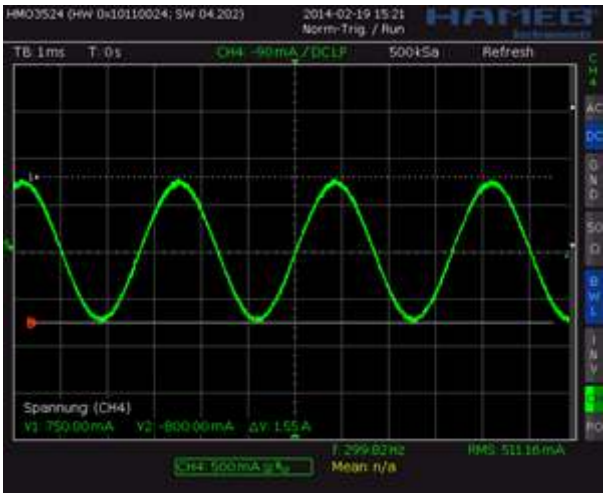


Figure 9. Motor Coil Sine Wave Current with StealthChop (Measured with Current Probe)

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, e.g., as part of a homing procedure (AT#2). The flowchart in the next figure shows the tuning procedure.

Table 10. 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–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.

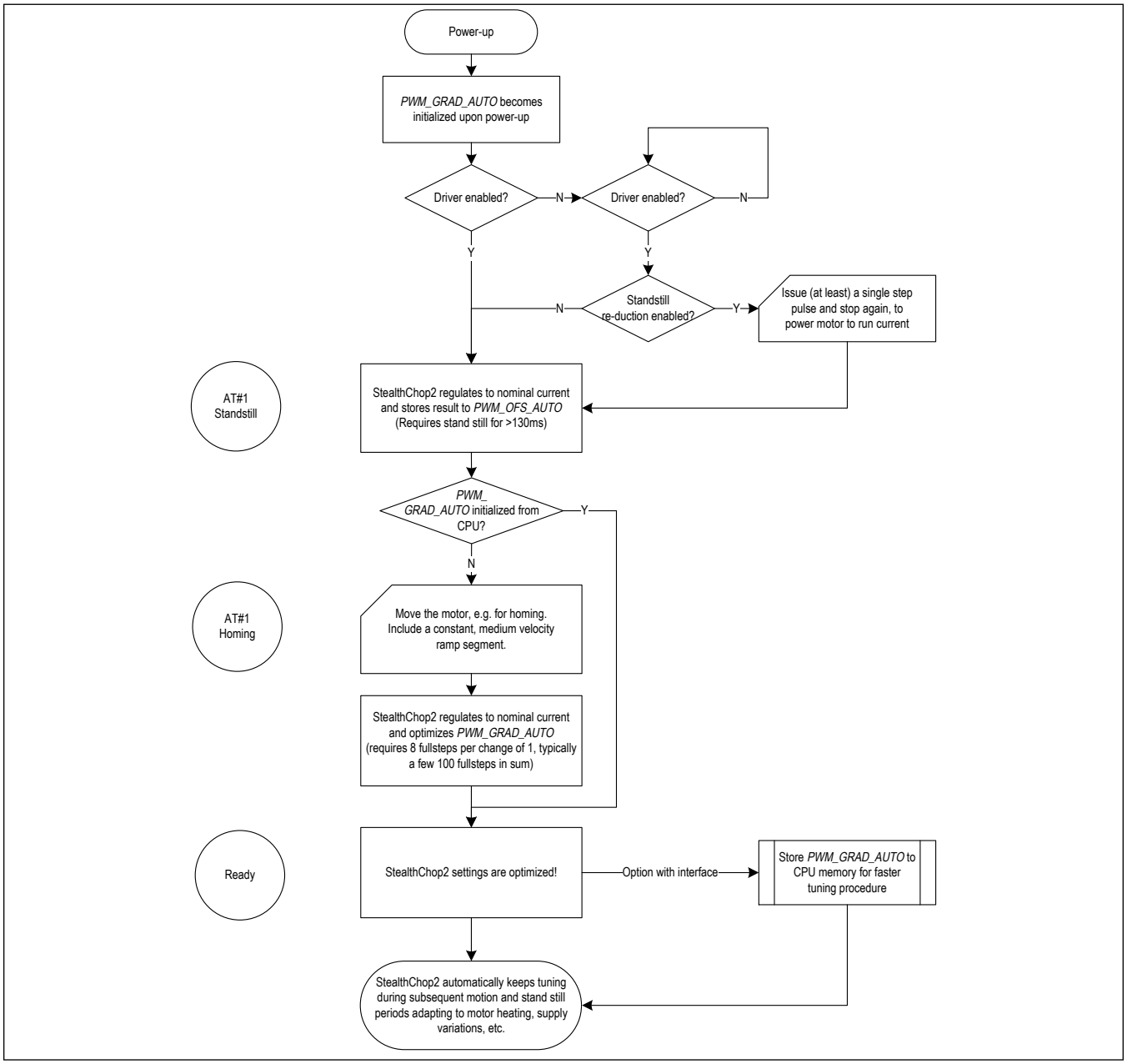


Figure 10. StealthChop2 Automatic Tuning Procedure

StealthChop2 Options

In order to match the motor current to a certain level, the effective PWM voltage becomes scaled depending on the actual motor velocity. Several additional factors influence the required voltage level to drive the motor at the target current: The motor resistance, its back EMF (e.g., 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 nor require any means of current measurement. This is perfect when motor type and supply voltage are well known. Therefore, we recommend the automatic mode, unless current regulation is not satisfying in the given operating conditions.

It is recommended to use application-specific initial tuning parameters, fitting the motor type and supply voltage. Additionally, operate in automatic tuning mode in order to respond to parameter change, e.g., 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 via 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 in order to adapt the frequency divider to the frequency of the clock source. A setting in the range of 20kHz–50kHz is good for most applications. It balances low current ripple and good higher velocity performance vs. dynamic power dissipation.

Table 11. 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}**
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* in order match the motor current to the target current. *PWM_REG* is the proportionality coefficient for this regulator. Basically, the proportionality coefficient should be as small as possible in order to get a stable and soft regulation behavior, but it must be large enough to allow the driver to quickly react to changes caused by variation of the motor target current (e.g., change of 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).

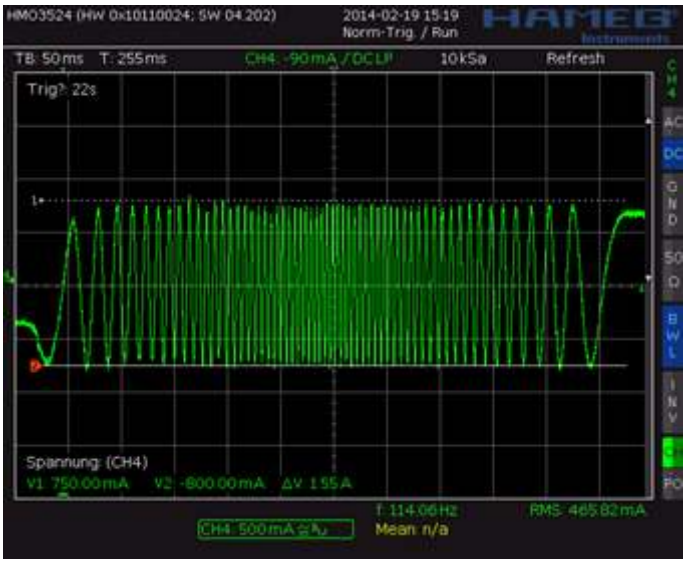


Figure 11. StealthChop2: Good Setting for PWM_REG

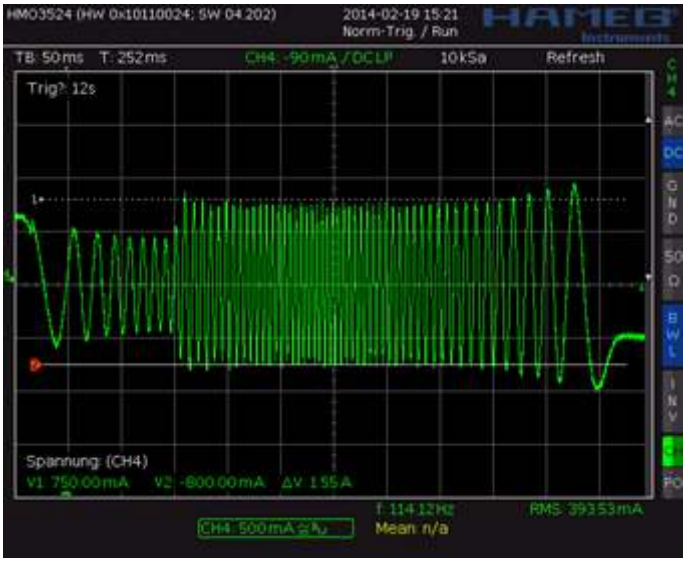


Figure 12. StealthChop2: Too Small Setting for PWM_REG during AT#2

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.

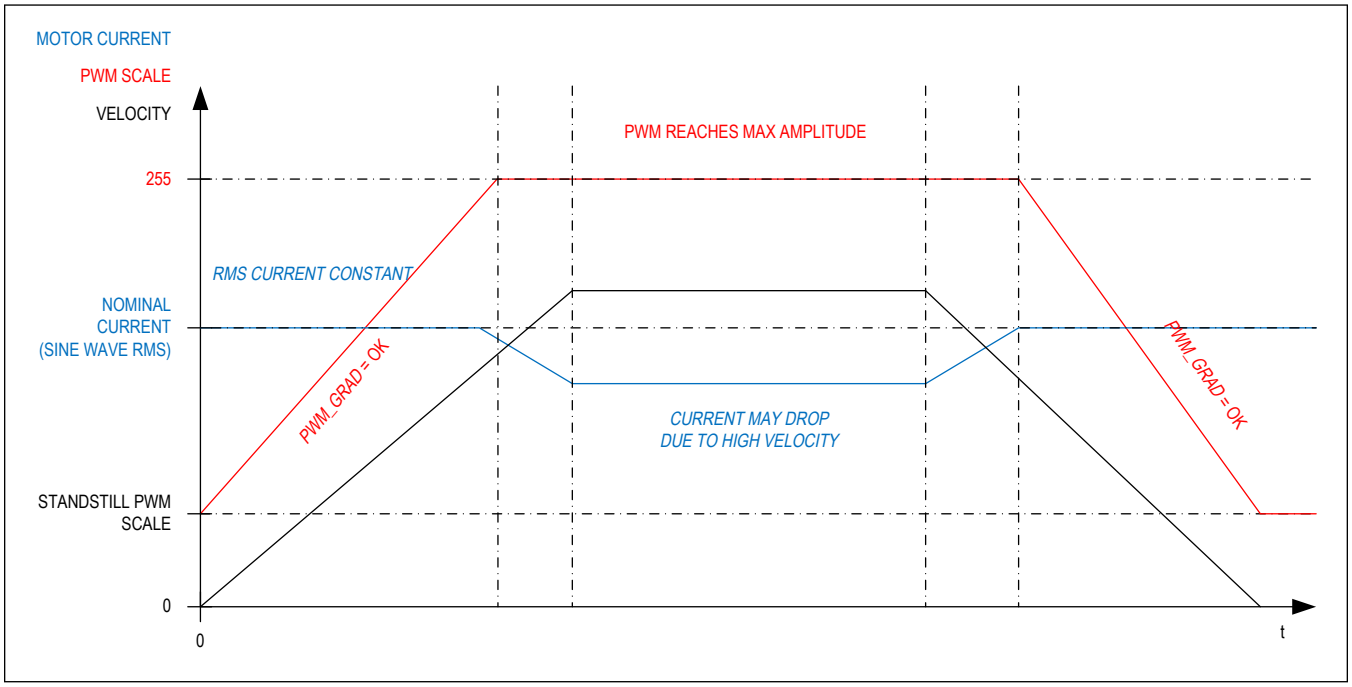


Figure 13. Successfully Determined PWM_GRAD(_AUTO) and PWM_OFS(_AUTO)

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 (e.g., for standstill power down) are automatically realized based on *PWM_OFS_AUTO* and *PWM_GRAD_AUTO* respectively based on *PWM_OFS* and *PWM_GRAD* with non-automatic current scaling. The freewheeling option allows going to zero motor current.

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

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

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

$I_{LowerLimit}$ can be treated as a thumb value for the minimum nominal *IRUN* motor current setting. In case the lower current limit is not sufficient to reach the desired setting, the driver retries with a lower chopper frequency in step AT#1, only.

f_{PWM} is the chopper frequency as determined by setting *PWM_FREQ*. In AT#1, the driver tries a lower, (roughly half frequency), in case it cannot reach the current. The frequency remains active in standstill, while currentscale *CS* = *IRUN.* With automatic standstill reduction, this is a short moment.

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

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

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

***Attention**:*

For automatic tuning, a lower coil current limit applies.

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

$$\text{PWM\_OFS} = \frac{374 \times R_{COIL} \times I_{COIL}}{V_M}$$

With V_M the motor supply voltage and I_{COIL} the target RMS current

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

$$U\text{PWM} = V_M \times \frac{\text{PWM\_SCALE}}{256} \times \frac{248}{256} \times \frac{1}{\sqrt{2}} = V_M \times \frac{\text{PWM\_SCALE}}{374}$$

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 TMC2240 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:

$$\text{PWM\_SCALE\_SUM} = \text{PWM\_OFS} + \text{PWM\_GRAD} \times 256 \times \frac{f_{STEP}}{f_{CLK}}$$

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

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

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

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, e.g., 51200 = 256 microsteps multiplied by 200 fullsteps for a 1.8° motor.

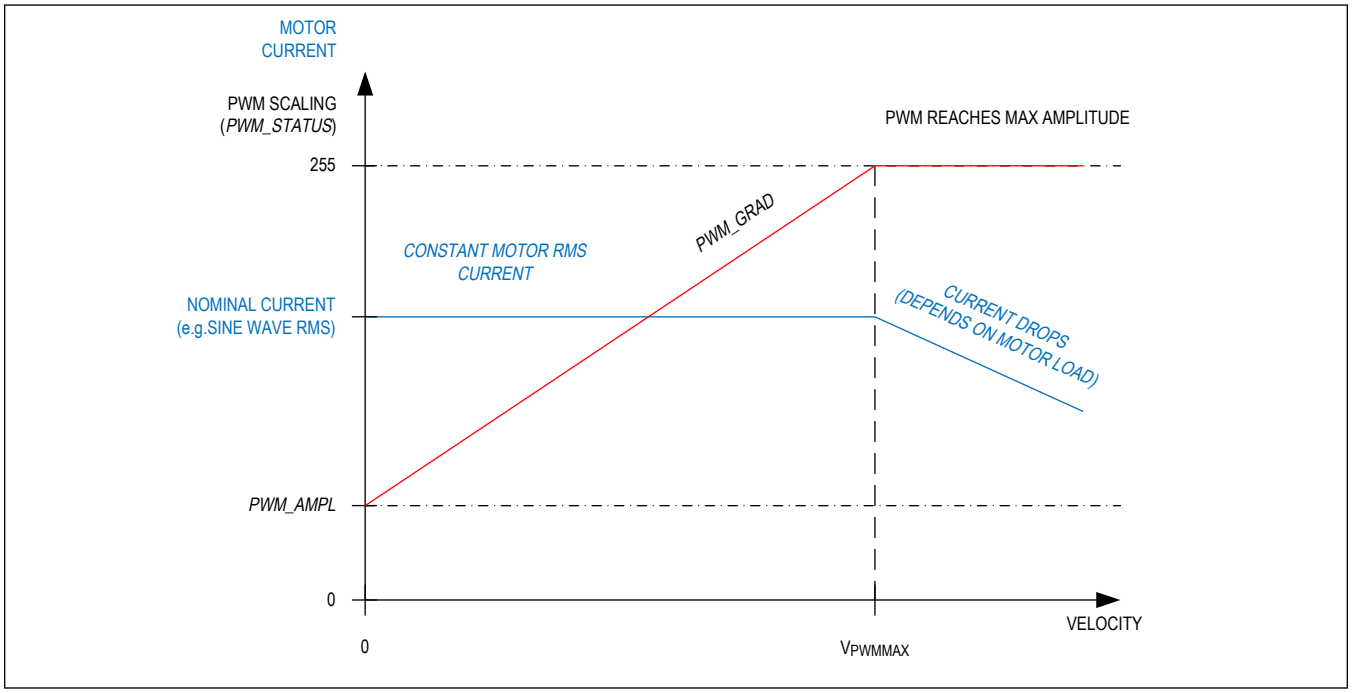


Figure 14. Velocity-Based PWM Scaling (pwm_autoscale = 0)

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 datasheets do not specify this value, as it can be deducted from motor torque and coil current rating. Within SI units, the numeric value of the back EMF constant C_{BEMF} has the same numeric value as the numeric value of the torque constant. For example, a motor with a torque constant of 1 Nm/A would have a C_{BEMF} of 1V/rad/s. Turning such a motor with 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 phase current for the specified holding torque.

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

The voltage is valid as RMS voltage per coil, thus the nominal current is multiplied by 2 in this formula, since the nominal current assumes a 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 TMC2240 allows combining StealthChop2 and SpreadCycle based on a velocity threshold. With this, StealthChop2 is only active at low velocities.

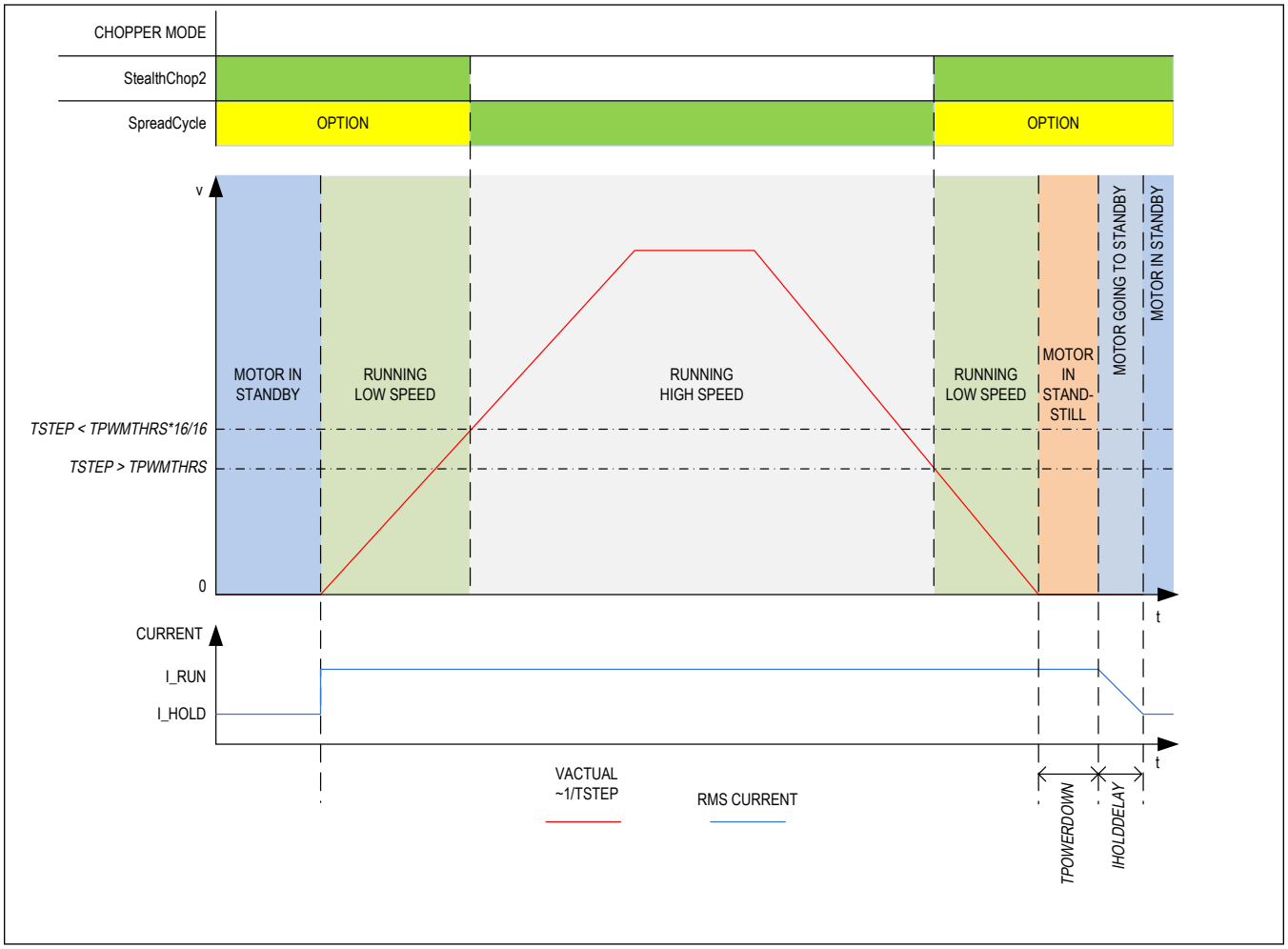


Figure 15. TPWMTHRS for Optional Switching to SpreadCycle

As a first step, both chopper principles should be parameterized and optimized individually. In a next step, a transfer velocity has to be fixed. For example, StealthChop2 operation is used for precise low speed positioning, while SpreadCycle shall be used for highly dynamic motion. *TPWMTHRS* determines the transition velocity. Read out *TSTEP* when moving at the desired velocity and program the resulting value to *TPWMTHRS*. Use a low transfer velocity to avoid a jerk at the switching point.

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 will occur with increased intensity. A high jerk may even produce a temporary overcurrent condition (depending on the motor coil resistance). At low velocities (e.g., 1 to a few 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 will apply 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 if you want to work with StealthChop2 only.

When enabling the StealthChop2 mode the first time using automatic current regulation, the motor must be at standstill

in order to allow a proper current regulation. When the drive switches to StealthChop2 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.

Tune the low side driver overcurrent detection to safely trigger upon motor stall, when using StealthChop2. This avoids high peak current draw from the power supply.

Open Load Flags

In StealthChop2 mode the status information is different compared to the cycle-by-cycle regulated SpreadCycle mode.

The flags OLA and OLB indicate that the current regulation is reaching the nominal current on both coils.

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

If desired, do an on-demand open load test using the SpreadCycle chopper as it delivers the safest result. With 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, e.g., due to a 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 in order to ensure a quick startup. With the freewheeling options, both freewheeling and passive braking can be realized. Passive braking is an effective eddy current motor braking, which consumes a minimum of energy because no active current is driven into the coils. However, passive braking will allow slow turning of the motor when a continuous torque is applied.

Operate the motor within your 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.

Parameters Controlling StealthChop2

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

Table 12. Parameters Controlling StealthChop2

PARAMETER	DESCRIPTION	SETTING	COMMENT
en_spread_ cycle	General disable for use of StealthChop2 (register GCONF).	1	Do not use StealthChop2
		0	StealthChop2 enabled
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. Entry the TSTEP reading (time between two microsteps) when operating at the desired threshold velocity.	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.	0 … 15	Upper four bits of 8 bit amplitude limit (Default = 12)
pwm_ autoscale	Enable automatic current scaling using current measurement. If off, use forward controlled velocity-based mode.	0	Forward controlled mode
		1	Automatic scaling with current regulator
pwm_ autograd	Enable automatic tuning of PWM_GRAD_AUTO	0	disable, use PWM_GRAD from register instead
		1	enable
PWM_FREQ	PWM frequency selection. Use the lowest setting giving good results. The frequency measured at each of the chopper outputs is half of the effective chopper frequency f_{PWM}.	0	$f_{PWM} = 2/1024\ f_{CLK}$
		1	$f_{PWM} = 2/683\ f_{CLK}$
		2	$f_{PWM} = 2/512\ f_{CLK}$
		3	$f_{PWM} = 2/410\ f_{CLK}$
PWM_REG	User defined PWM amplitude regulation loop P-coefficient. A higher value leads to a higher adaptation speed when pwm_autoscale = 1.	1 … 15	Results in 0.5 to 7.5 steps for PWM_SCALE_AUTO regulator per fullstep
PWM_OFS	User defined PWM amplitude (offset) for velocity-based scaling and initialization value for automatic tuning of PWM_OFFS_AUTO.	0 … 255	PWM_OFS = 0 disables linear current scaling based on current setting
PWM_GRAD	User defined PWM amplitude (gradient) for velocity-based scaling and initialization value for automatic tuning of PWM_GRAD_AUTO.	0 … 255	
FREEWHEEL	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.	0	Normal operation
		1	Freewheeling
		2	Coil short via LS drivers
		3	Coil short via HS drivers

Table 12. Parameters Controlling StealthChop2 (continued)

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.	-255 … 255	(read-only) Scaling value becomes frozen when operating in SpreadCycle
PWM_GRAD _AUTO PWM_OFS _AUTO	Allow monitoring of the automatic tuning and determination of initial values for PWM_OFS and PWM_GRAD.	0 … 255	(read-only)
TOFF	General enable for the motor driver, the actual value does not influence StealthChop2	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.	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.

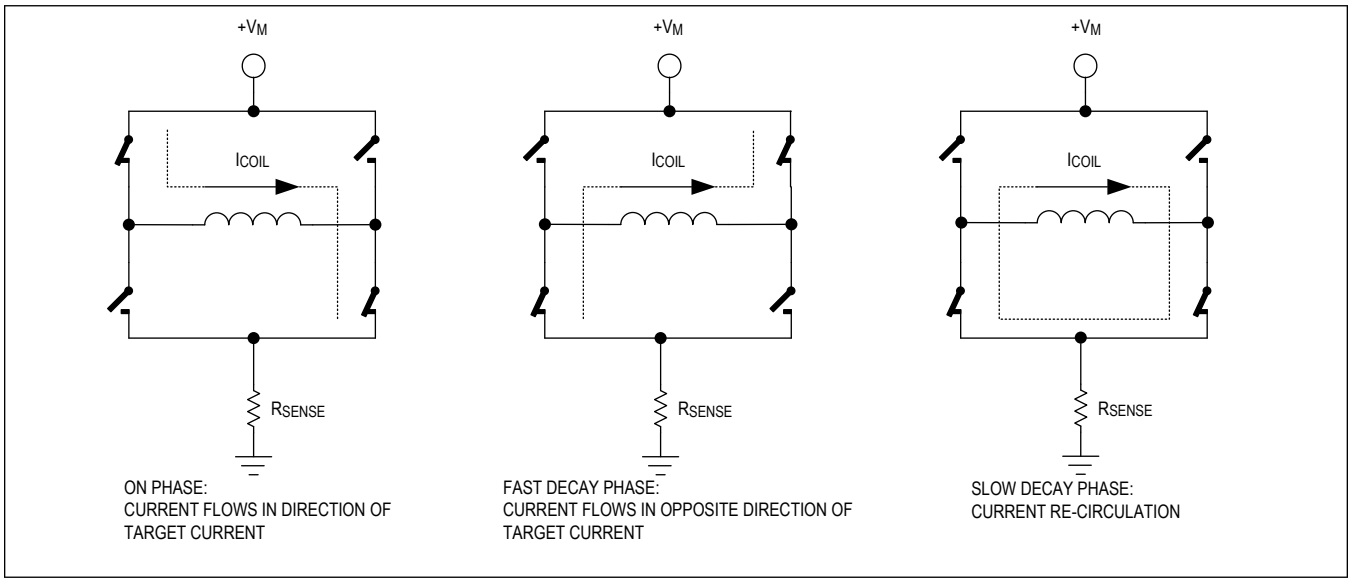


Figure 16. Typical Chopper Decay Phases

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 16kHz to 30kHz. 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 13. Parameters Controlling SpreadCycle and Classic Constant Off Time Chopper

PARAMETER	DESCRIPTION	SETTING	COMMENT
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.	0	chopper off
		1…15	off time setting N_{CLK} = 24 + 32 x *TOFF* (1 will work with minimum blank time of 24 clocks)
TBL	Selects the comparator *blank time*. This time needs to safely cover the switching event and the duration of the ringing. For most applications, a setting of 1 or 2 is good. For highly capacitive loads, e.g., when filter networks are used, a setting of 2 or 3 will be required.	0	16 t_{CLK}
		1	24 t_{CLK}
		2	36 t_{CLK}
		3	54 t_{CLK}
chm	Selection of the *chopper mode*	0	SpreadCycle
		1	classic const. off time

SpreadCycle Chopper

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

Each chopper cycle 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%–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 (Assumption: Two slow decay cycles make up for 50% of overall chopper cycle time.)
- t_{OFF} = 1 / 25kHz × 50 / 100 × 1 / 2 = 10 µ *s*
- 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 would require a setting of TOFF = 3 or 4
- With 16MHz clock this results in TOFF = 4.6, which would require a setting of TOFF = 4 or 5

Hint: Highest motor velocities sometimes benefit from setting TOFF to 1 or 2 and a short TBL of 1 or 0.

The hysteresis start setting forces the driver to introduce a minimum amount of current ripple into the motor coils. The current ripple must be higher than the current ripple which is caused by resistive losses in the motor in order to give best

microstepping results. This will allow the chopper to precisely regulate the current 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 will lead to a lower chopper frequency. The motor inductance limits the ability of the chopper to follow a changing motor current. Further the duration of the on phase and the fast decay must be longer than the blanking time, because the current comparator is disabled during blanking.

It is easiest to find the best setting by starting from a low hysteresis setting (e.g., *HSTRT* = 0, *HEND* = 0) and increasing *HSTRT*, until the motor runs smoothly at low velocity settings. This can best be checked when measuring the motor current with a current probe. Checking the sine wave shape near the zero transition will show a small ledge between both half waves in case the hysteresis setting is too small. At medium velocities (e.g., 100 fullsteps to 400 fullsteps per second), a too low hysteresis setting will lead to increased humming and vibration of the motor. A too high hysteresis setting will lead to reduced chopper frequency and increased chopper noise but will not yield any benefit for the wave shape.

As experiments show, the setting is quite independent of the motor because higher current motors typically also have a lower coil resistance. Therefore choosing a low to medium default value for the hysteresis (for example, effective hysteresis = 4) normally fits most applications. The setting can be optimized by experimenting with the motor: A too low setting will result in reduced microstep accuracy, while a too high setting will lead to more chopper noise and motor power dissipation. When the fast decay time becomes slightly longer than the blanking time, the setting is optimum. You can reduce the off-time setting if this is hard to reach.

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

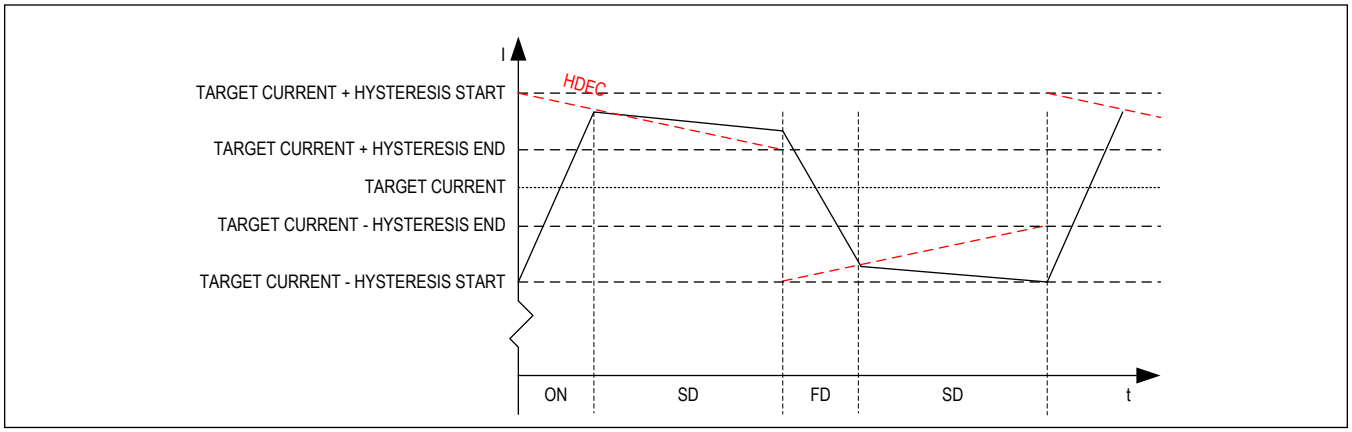


Figure 17. SpreadCycle Chopper Scheme Showing Coil Current during a Chopper Cycle

Table 14. SpreadCycle Mode Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
HSTRT	*Hysteresis start* setting. This value is an offset from the hysteresis end value *HEND*.	0…7	*HSTRT* = 1…8 This value adds to HEND.

Table 14. SpreadCycle Mode Parameters (continued)

HEND	*Hysteresis end* setting. Sets the hysteresis end value after a number of decrements. The sum *HSTRT* + *HEND* must be ≤16. At a current setting of max. 30 (amplitude reduced to 240), the sum is not limited.	0…2	-3…-1: negative HEND
		3	0: zero HEND
		4…15	1…12: positive HEND

Even at HSTRT = 0 and HEND = 0, the TMC2240 sets a minimum hysteresis via analog circuitry.

Example:

A hysteresis of 4 has been chosen. You might decide to not use hysteresis decrement. In this case set:

HEND = 6 (sets an effective end value of 6 - 3 = 3)

HSTRT = 0 (sets minimum hysteresis, e.g., 1: 3 + 1 = 4)

In order to take advantage of the variable hysteresis, we can set most of the value to the HSTRT, e.g., 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. Perfectly tuned, it also gives good results. In combination with RDSon current sensing without external sense resistors, this chopper mode can bring a benefit with regard to audible high-pitch chopper noise.

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

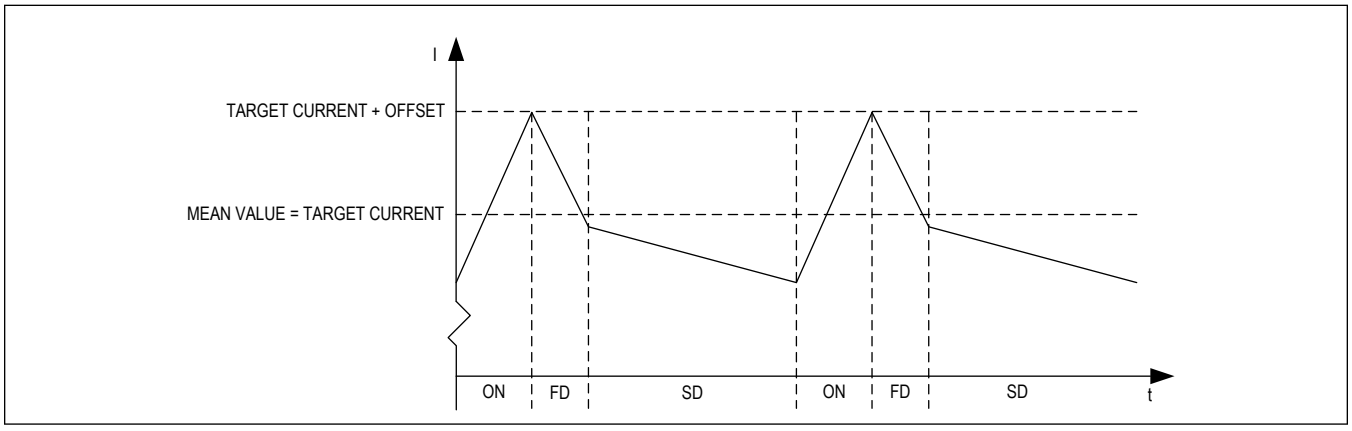


Figure 18. Classic Constant Off-Time Chopper with Offset Showing Coil Current

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 figures below). 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.

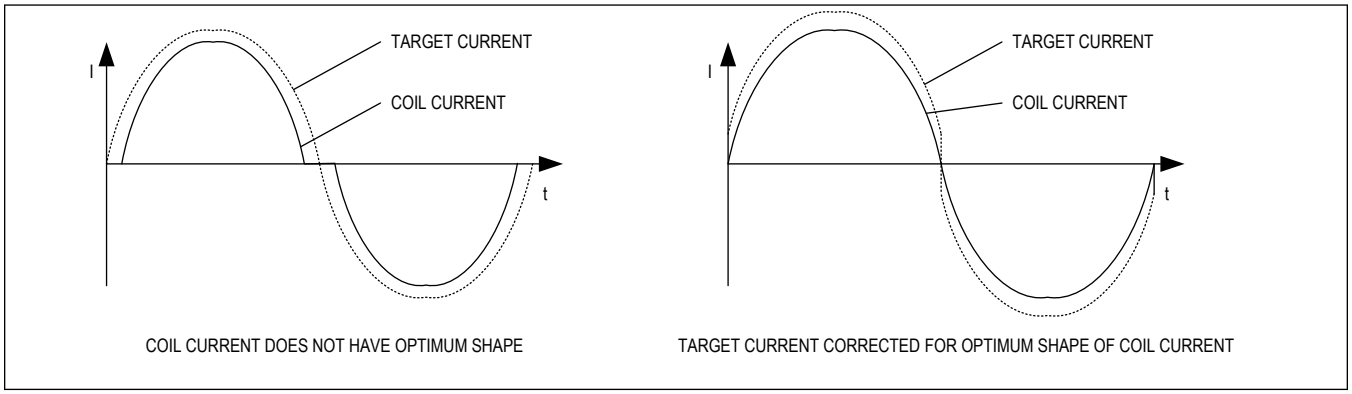


Figure 19. Zero Crossing with Classic Chopper and Correction Using Sine Wave Offset

Table 15. Parameters Controlling Constant Off-Time Chopper Mode

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

Integrated Current Sense

A non-dissipative Current Sensing is integrated. This feature eliminates the bulky external power resistors which are normally required for this function. The ICS results in a dramatic space and power saving compared with mainstream applications based on external sense resistor.

Setting the Motor Current

Table 16. 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.	0…31	scaling factor 1/32, 2/32, … 32/32
IHOLD	Identical to IRUN, but for motor in standstill.		

Table 16. Parameters Controlling the Motor Current (continued)

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, e.g., roughly one second at 16MHz.	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 jerk on the power supply current.	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 is selected with an external reference resistor and 2 bits in the DRV_CONF register.

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 equation below shows the full-scale current as a function of the R_{REF} shunt 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 17. IFS Full-Scale Range Settings (Example for R_{REF} = 12kΩ)

REGISTER CONFIG DRV_CONF bits 1..0	K_{IFS} (A x kΩ)	MAX. FS SETTING	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 standard 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 lower current is required.

Velocity-Based Mode Control

The TMC2240 allows the configuration of different chopper modes and modes of operation for optimum motor control. Depending on the motor load, the different modes can be optimized for lowest noise & high precision, highest dynamics,

or maximum torque at highest velocity. Some of the features like CoolStep or StallGuard2 are useful in a limited velocity range. A number of velocity thresholds allow combining the different modes of operation within an application requiring a wide velocity range.

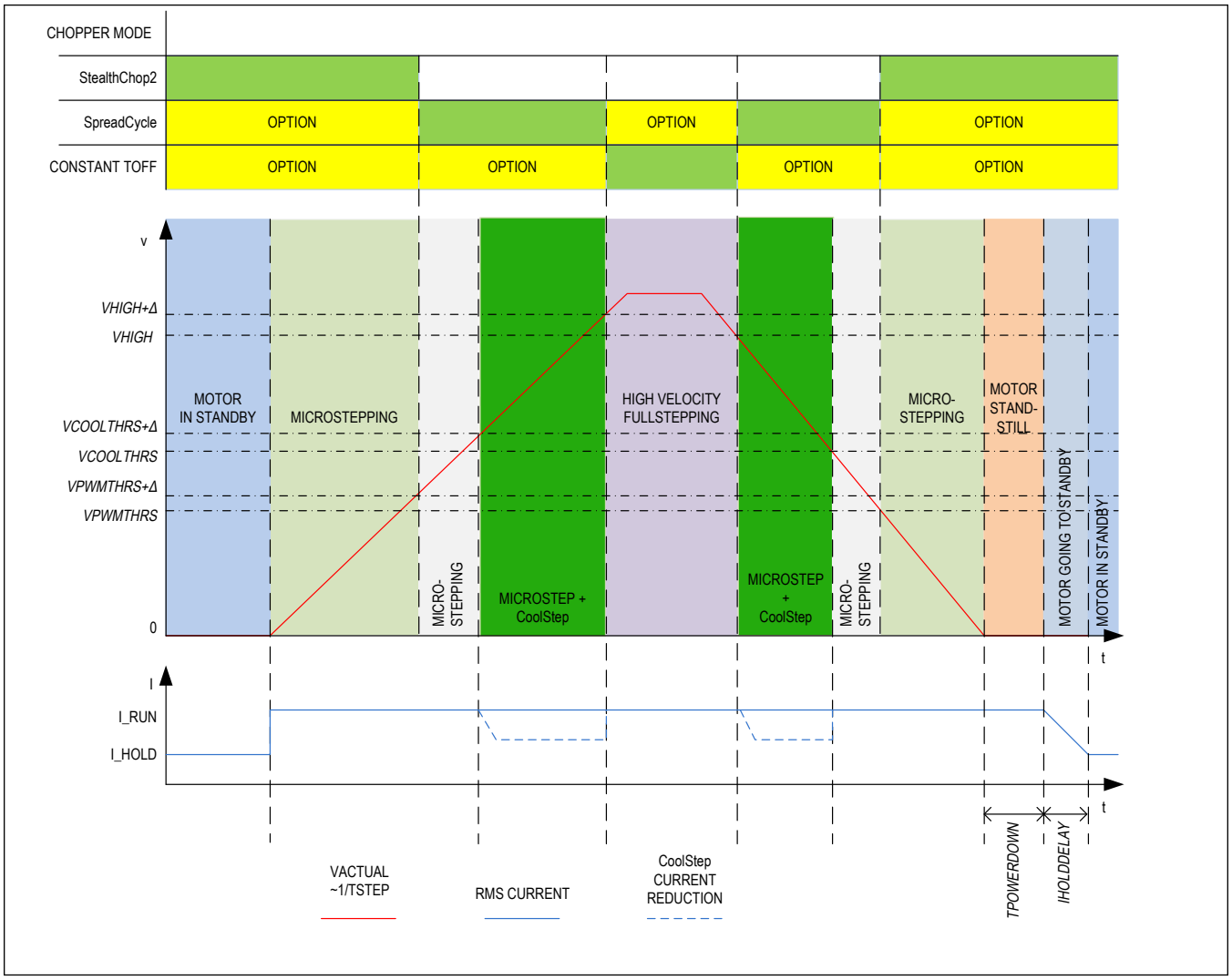


Figure 20. Choice of Velocity-Dependent Modes

The figure 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 microsteps. This way, the thresholds do not have to be adapted when the microstep resolution is changed. The thresholds represent the same motor velocity, independent of the microstep settings. *TSTEP* becomes compared to these threshold values. A hysteresis of 1/16 *TSTEP* resp. 1/32 *TSTEP* is applied to avoid continuous toggling of the comparison results when a jitter in the *TSTEP* measurement occurs. The upper switching velocity is higher by 1/16, resp. 1/32 of the value set as threshold. The motor current can be programmed to a run and a hold level, dependent on the standstill flag *stst*.

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

deactivation via software.

Table 18. Velocity-Based Mode Control Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
stst	Indicates motor standstill in each operation mode. Time is 2^20 clocks after the last step pulse.	0/1	Status bit, read only
TPOWER DOWN	This is the delay time after standstill (*stst*) of the motor to motor current power down. Time range is about 0 to 4 seconds. Setting 0 is no delay, 1 a one clock cycle delay. Further increment is in discrete steps of 2^18 clock cycles.	*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 standstill.	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	0… 1048575	Setting to control the upper velocity threshold for operation in StealthChop2
TCOOLTHRS	*TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: ● CoolStep is enabled, if configured ● StealthChop2 voltage PWM mode is disabled *TCOOLTHRS* ≥ *TSTEP* ● Stall output signal is enabled, if configured	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 fullstep stall detection.	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 x* 31/ 32) - 1 (upper velocity threshold)	0	Hysteresis is 1/16
		1	Hysteresis is 1/32
vhighfs	This bit enables switching to fullstep, when *VHIGH* is exceeded. Switching takes place only at 45° position. The fullstep target current uses the current value from the microstep table at the 45° position.	0	No switch to fullstep
		1	Fullstep at high velocities
vhighchm	This bit enables switching to *chm* = 1 and *fd* = 0, when *VHIGH* is exceeded. This way, a higher velocity can be achieved. Can be combined with *vhighfs* = 1. If set, the *TOFF* setting automatically becomes doubled during high velocity operation in order to avoid doubling of the chopper frequency.	0	No change of chopper mode
		1	Classic const. Toff chopper at high velocities
en_pwm_ mode	StealthChop2 voltage PWM enable flag (depending on velocity thresholds). Switch from off to on state while in standstill, only.	0	No StealthChop2
		1	StealthChop2 below VPWMTHRS

StallGuard2 Load Measurement

To fit different motor control schemes, the TMC2240 offers two types of StallGuard2 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. At maximum motor load,

the value goes to zero or near to zero. This corresponds to a load angle of 90° between the magnetic field of the coils and magnets in the rotor. This also is the most energy-efficient point of operation for the motor.

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

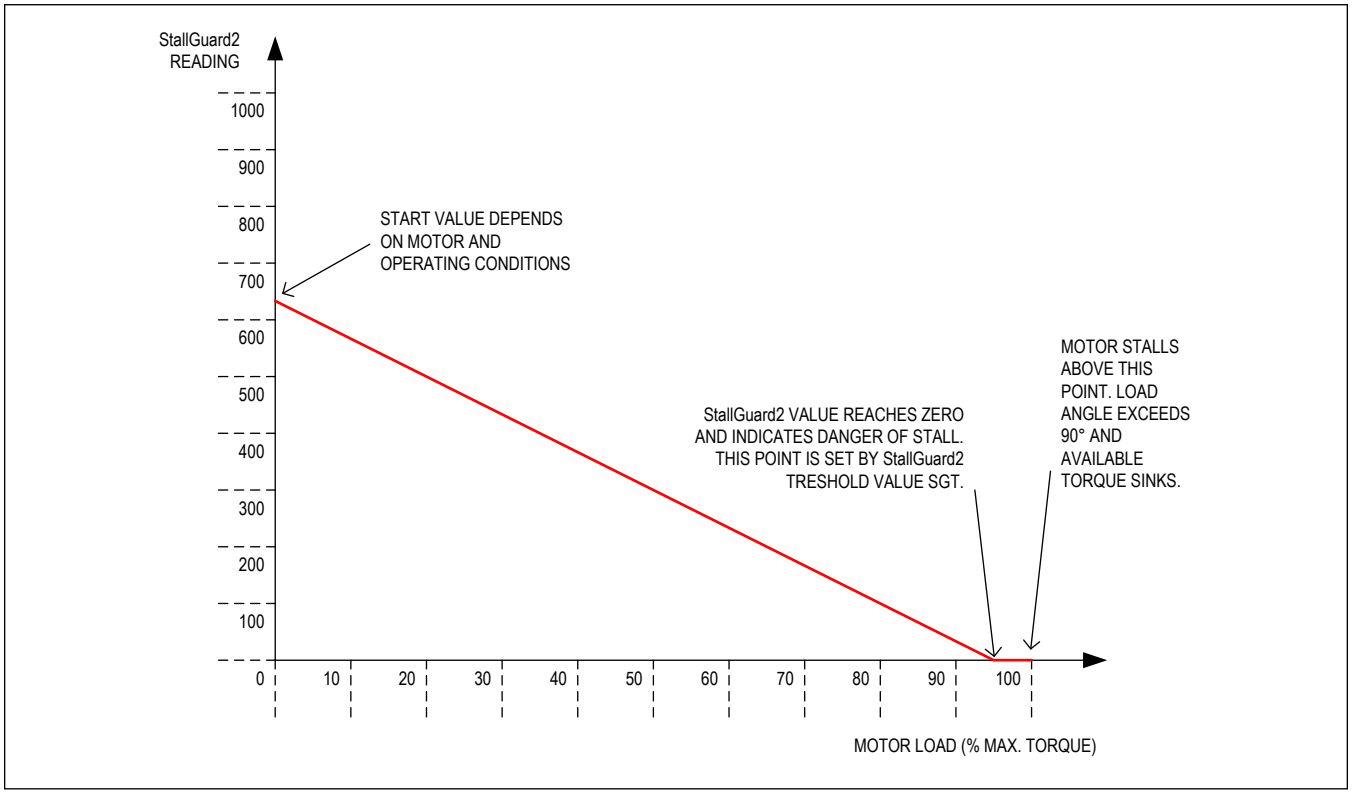


Figure 21. Function Principle of StallGuard2

Table 19. 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**

Table 19. StallGuard2-Related Parameters (continued)

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

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, e.g., some value within the range 0 to 100. The stall threshold should be a value safely within the operating limits, to allow for parameter stray. The response at an *SGT* setting at or near 0 gives some idea on the quality of the signal: Check the *SG_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 you set the *SGT* value 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 will be visible. After the step loss, the motor will vibrate and show 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 away from the hard stop to provide the distance required for the acceleration phase. After setting up SGT, start a motion into the direction of the hard stop and configure *diag0_stall* or *diag1_stall* to indicate the stall condition to the external controller using one of the diagnostic outputs. Once a stall is detected, the controller stops the motor. The stop condition also is indicated by the flag STALLGUARD in DRV_STATUS.

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 above will compensate for this. Other conditions will also lead to extreme settings of *SGT* and poor response of the measurement value *SG_RESULT* to the motor load.

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

StallGuard4 Load Measurement

StallGuard4 is developed for operation in conjunction with StealthChop2. It 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.

In order to use StallGuard4, check the sensitivity of the motor at border conditions.

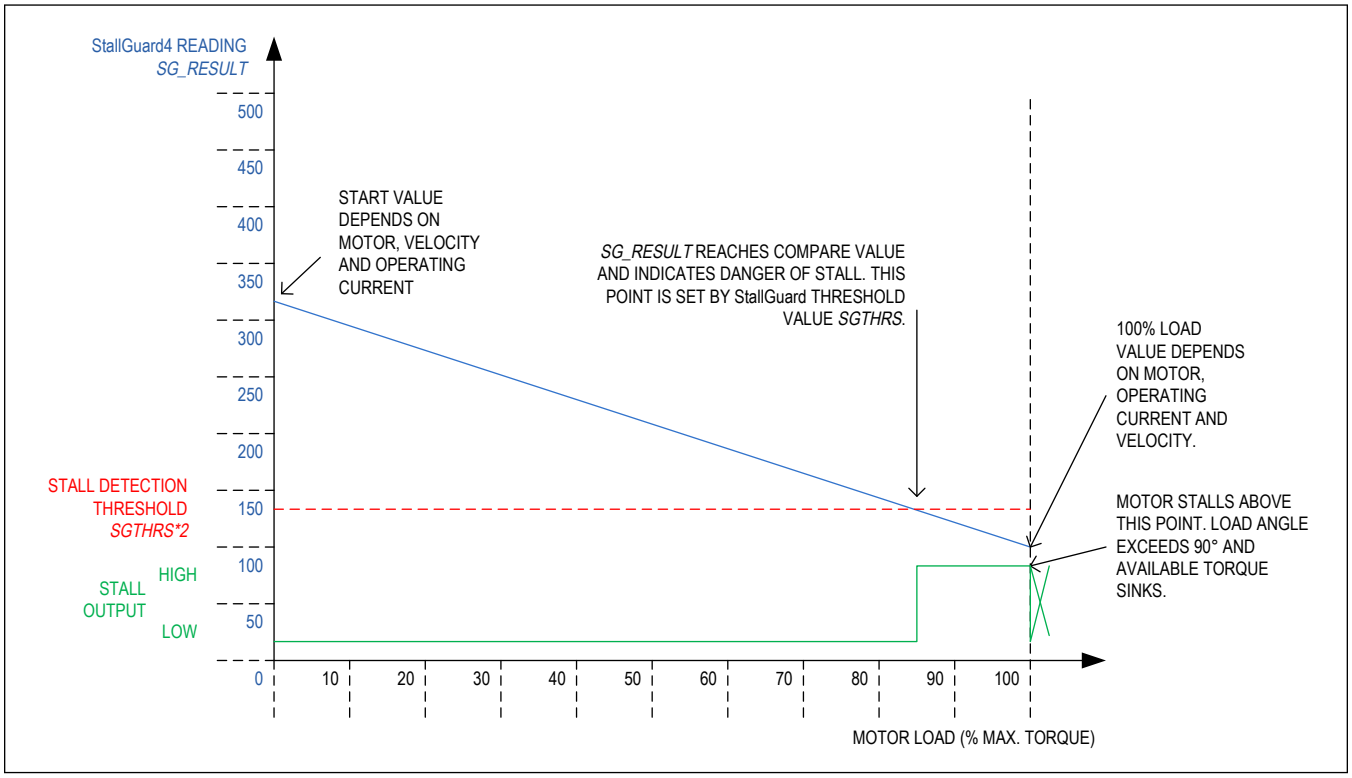


Figure 22. StallGuard4 Mode of Operation

Table 20. StallGuard4-Related Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
SG4THRS	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	The double of 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 20. 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

StallGuard4 vs. StallGuard2

StallGuard4 is optimized for operation with StealthChop2, and its predecessor StallGuard2 works with

SpreadCycle. 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.

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 your 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 via DIAG output (configure properly, 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 becomes stopped 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 (e.g., 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 become generated: *SG4_IND_0* upon falling 0-transition of the cosine wave (coil A); *SG4_IND_1* upon rising 0-transition of the co-sine 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 usage 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 needs to be determined the motor can drive without stalling. 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 will rule out certain effects which 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 will 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 leads to poor response. These velocities are typically characterized by the motor back EMF exceeding the supply voltage.

CoolStep Operation

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. However, the tuning has to be done for either one or the other. 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 21. CoolStep Critical Parameters

PARAMETER	DESCRIPTION	RANGE	COMMENT
SEMIN	4-bit unsigned integer that sets a *lower threshold*. If *SG_RESULT* goes below this threshold, CoolStep increases the current to both coils. The 4-bit *SEMIN* value is scaled by 32 to cover the lower half of the range of the 10-bit *SG_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, 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 figure below 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*, and CoolStep increases the current. When the load decreases, *SG_RESULT* rises above (*SEMIN* + *SEMAX* + 1) x 32, and the current is reduced.

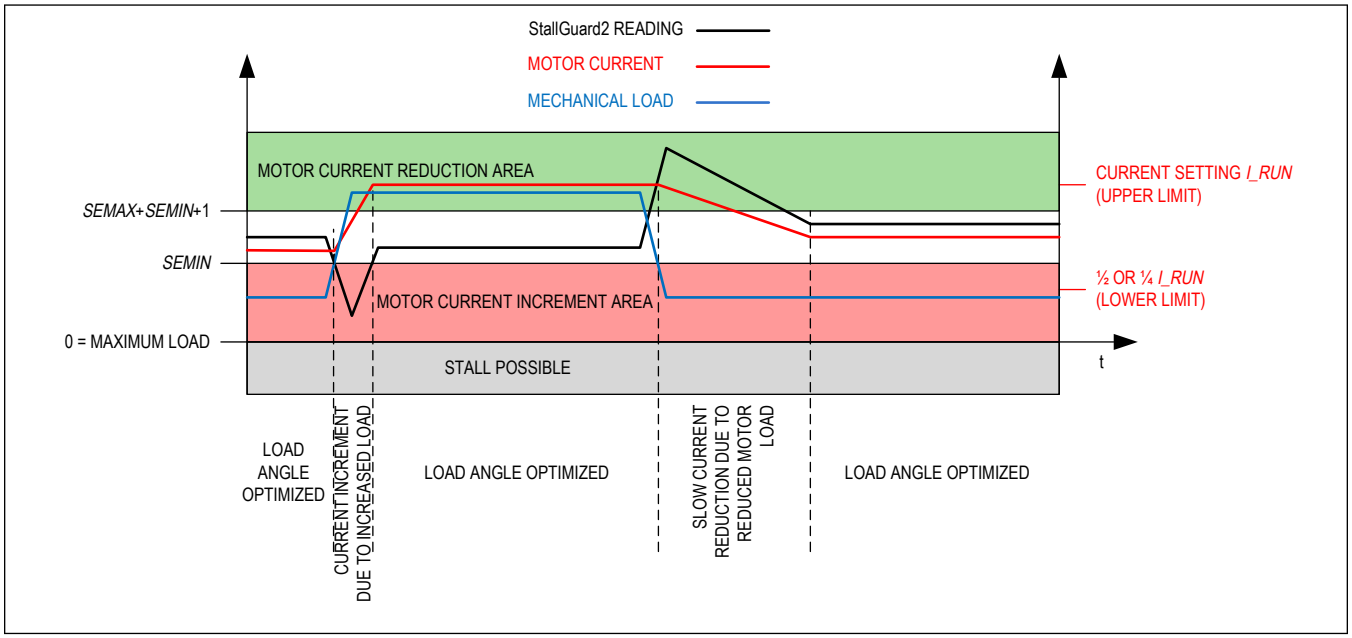


Figure 23. CoolStep Adapts Motor Current to the Load

Table 22. CoolStep Additional Parameters and Status Information

PARAMETER	DESCRIPTION	RANGE	COMMENT
SEUP	Sets the *current increment step*. The current becomes incremented for each measured StallGuard2 value below the lower threshold.	0…3	step width is 1, 2, 4, 8
SEDN	Sets the number of StallGuard2 readings above the upper threshold necessary for each *current decrement* of the motor current.	0…3	number of StallGuard2 measurements per decrement: 32, 8, 2, 1
SEIMIN	Sets the *lower motor current limit* for CoolStep operation by scaling the *IRUN* current setting.	0	0: 1/2 of IRUN
		1	1: 1/4 of IRUN
TCOOLTHRS	Lower velocity threshold for switching on CoolStep. Below this velocity CoolStep becomes disabled. Adapt to the lower limit of the velocity range where StallGuard2 gives a stable result. *Hint:* May be adapted to disable CoolStep during acceleration and deceleration phase by setting identical to *VMAX*.	1… 2^20-1	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 gives a stable result.	1… 2^20-1	Also controls additional functions like switching to fullstepping.
STATUS WORD	**DESCRIPTION**	**RANGE**	**COMMENT**
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 via *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.

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

Low Velocity and Standby Operation

As CoolStep is not able to measure the motor load in standstill and at very low RPM, a lower velocity threshold is provided in the driver. It should be set to an application specific default value. Below this threshold the normal current setting via *IRUN* respectively *IHOLD* is valid. An upper threshold is provided by the *VHIGH* setting. Both thresholds can be set as a result of the StallGuard2 tuning process.

Diagnostic Outputs

Operation with an external motion controller often requires quick reaction to certain states of the stepper motor driver. Therefore, the DIAG outputs supply a configurable set of different real time information complementing the STEP/DIR interface.

Both, the information available at DIAG0 and DIAG1 can be selected as well as the type of output (low active open drain – default setting, or high active push-pull). In order to determine a reset of the driver, DIAG0 always shows a power-on reset condition by pulling low during a reset condition. The figure below shows the available signals and control bits.

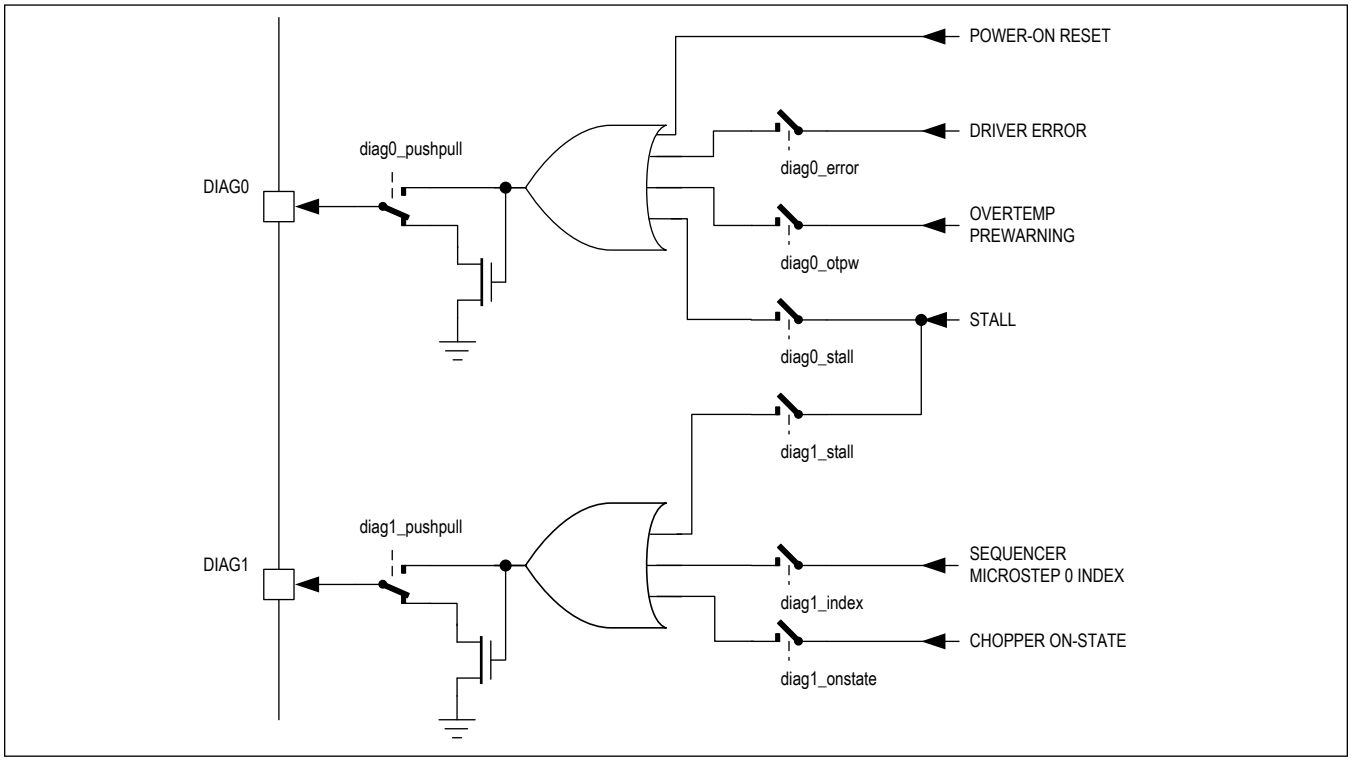


Figure 24. DIAG0 and DIAG1 Output Options

The stall output signal allows StallGuard to be handled by the external motion controller like a stop switch.

Depending on the chopper mode, it becomes activated whenever the StallGuard value *SG_RESULT* reaches zero, respectively when *SG4_RESULT* falls below *SG4_THRS*, and at the same time the velocity condition is fulfilled (*TSTEP* ≤ *TCOOLTHRS*).

Chopper on-state shows the on-state of both coil choppers (alternating) when working in SpreadCycle or constant off time in order to determine the duty cycle.

The index output signals the microstep counter zero position to allow the application to reference the drive to a certain current pattern.The duration of the index pulse corresponds to the duration of the microstep. When working without interpolation at less than 256 microsteps, the index time goes down to two CLK clock cycles. The index output signals the positive zero transition of the coil B microstep wave.

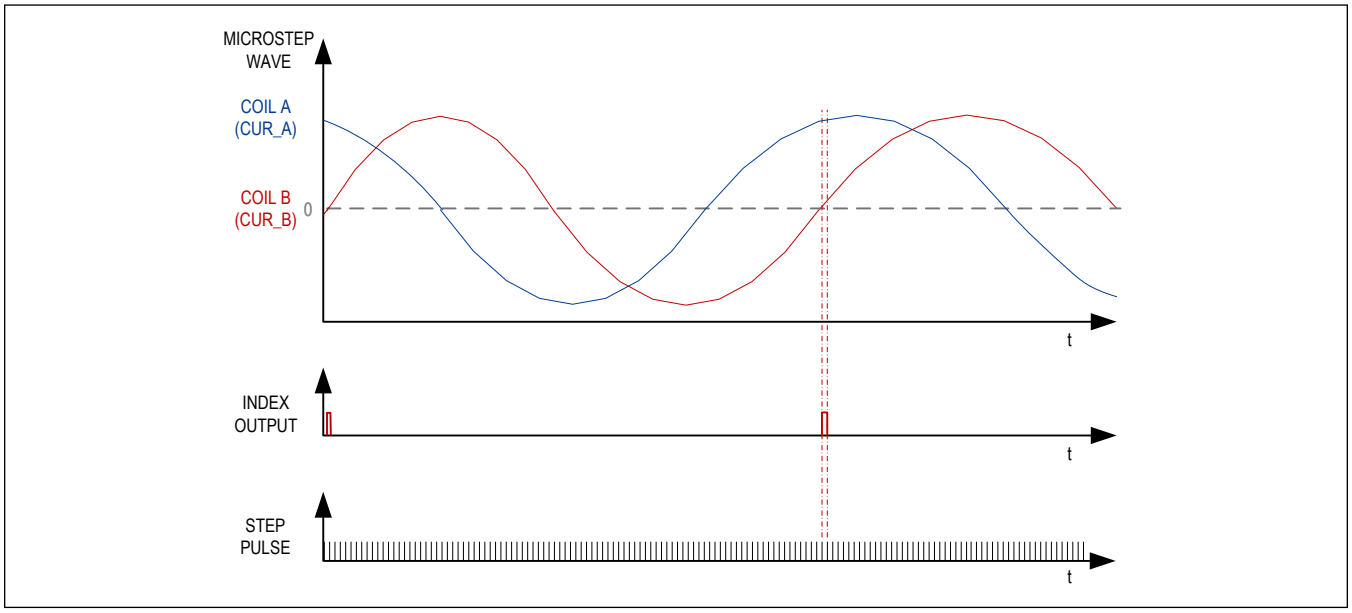


Figure 25. Index Signal at Positive Zero Transition of the Coil B Microstep Wave

Sine Wave Lookup Table

The TMC2240 provides a programmable look-up table for storing the microstep current wave. As a default, the table is pre-programmed with a sine wave, which is a good starting point for most stepper motors. Reprogramming the table to a motor specific wave allows drastically improved microstepping especially with low-cost motors. The user benefits are:

- Microstepping – extremely improved with low cost motors
- Motor – runs smooth and quiet
- Torque – reduced mechanical resonances yields improved torque
- Low frequency motor noise - reduced by adapting the sine & cosine wave shift for the actual motor's manufacturing tolerance

Microstep Table

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

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

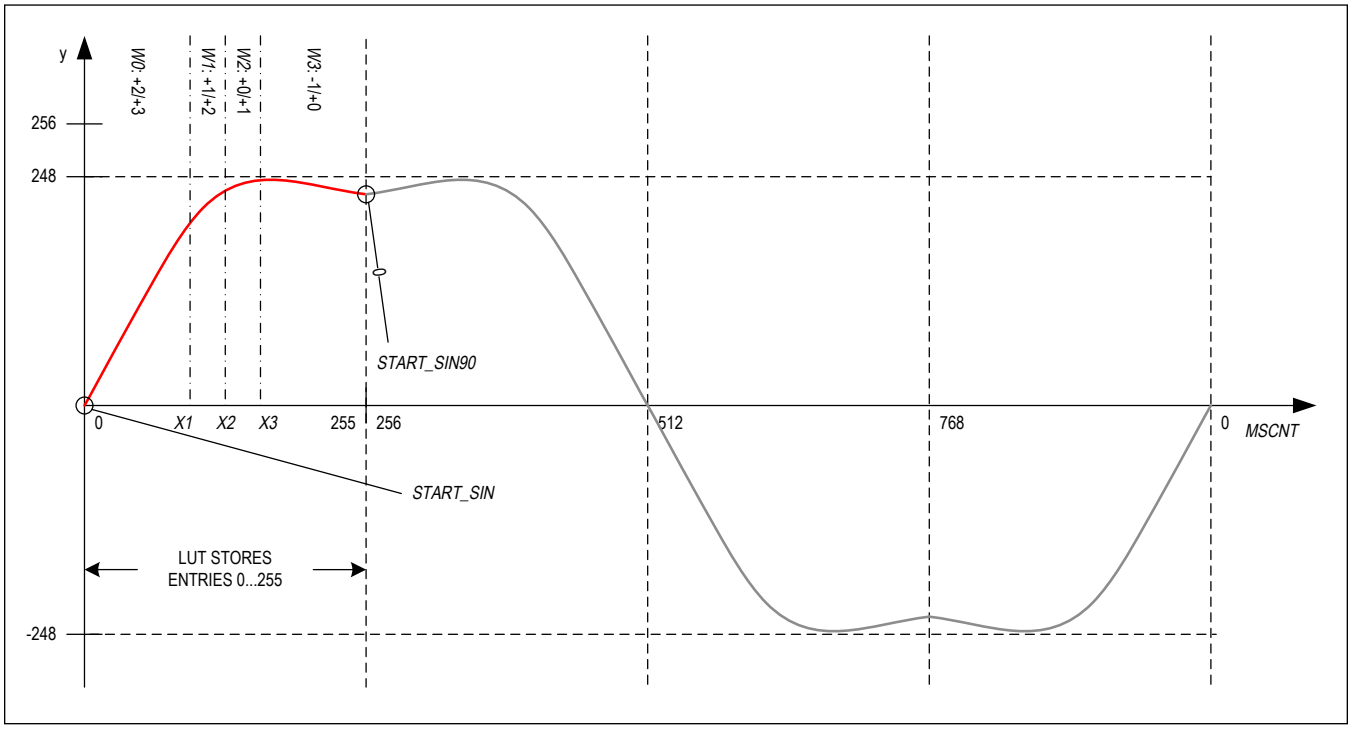


Figure 26. LUT Programming Example

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 needs to 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 will require moderate adaptations of a few, 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.

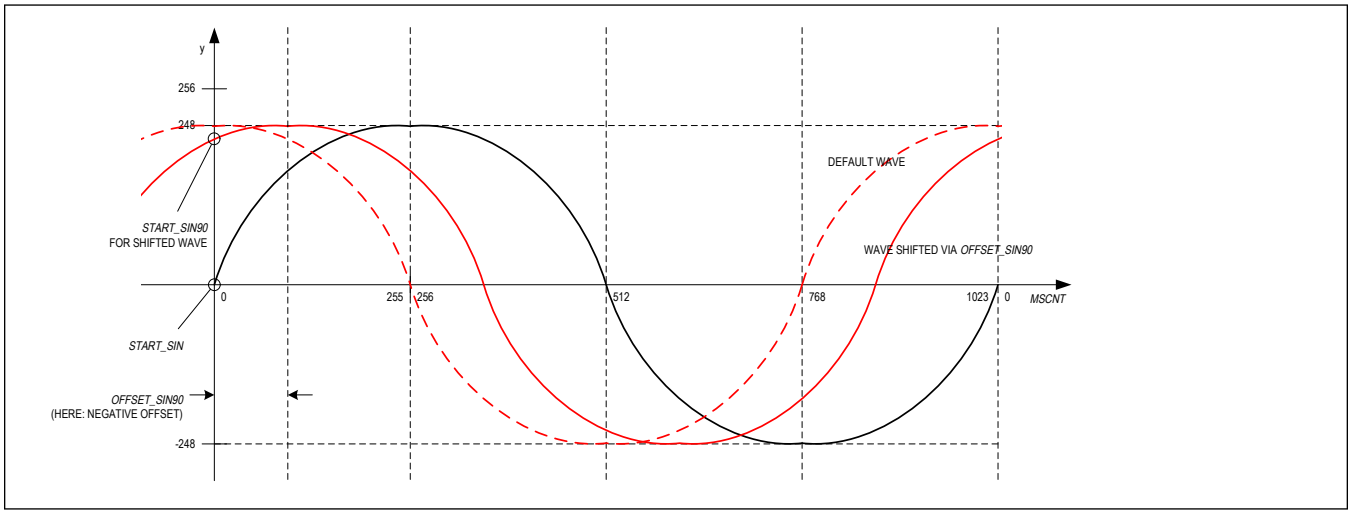


Figure 27. Shifting the Cosine Wave via OFFSET_SIN90

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 TMC2240 is equipped with an incremental encoder interface for ABN encoders. The encoder gives positions via 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.

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

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

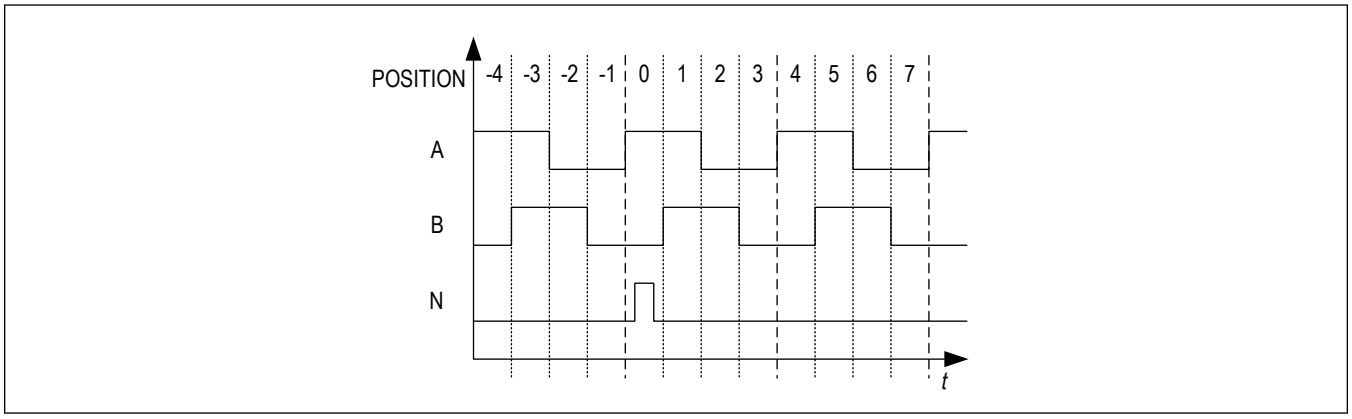


Figure 28. Outline of ABN Signals of an Incremental Encoder

The Encoder Constant *ENC_CONST*

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

Examples:

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

The Encoder Counter *X_ENC*

The encoder counter *X_ENC* holds the current encoder position ready for read out. Different modes concerning handling of the signals A, B, and N take into account active low and active high signals found with different types of encoders. For more details, see the Register Map.

The Register *ENC_STATUS*

The register *ENC_STATUS* holds the status concerning the event of an encoder clear upon an N channel signals. The

register *ENC_LATCH* stores the actual encoder position on an N signal event.

Checking for encoder latched event

Option 1: Check ENC_LATCH for change. It starts up with 0, and will show the encoder count where the N-event occurred, after starting motion for the first time. For consecutive rotations, it will show increased / decreased values and thus always changes.

Option 2: Check for the interrupt output active, and read the flag only following active interrupt output.

Please do not use the *ENC_STATUS* event flag for active, high-frequent polling, as in the event of a parallel read event and encoder N event, the flag will be cleared at the same moment, and will be missed.

Setting the Encoder to Match Motor Resolution

Encoder example settings for motor parameters: USC = 256 microsteps, 200 fullstep motor

Factor = FSC x USC/encoder resolution

Table 23. 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 = $51 \times 2^{16} + 0.2 \times 10000$

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 will stop abruptly and go to the power down state, as configured via *IHOLD*, *IHOLD_DELAY* and StealthChop2 standstill options.

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 <30µs are filtered out and will 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 need to be re-configured.

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).

Restart the Stepper Motor Without Position Loss

A self-locking drive allows switching off the motor completely without loss of position. Locking can result from mechanical friction and from the stepper motor cogging torque. Most stepper motors have a cogging torque in the range of a few percent of their nominal torque, which also will contribute to the motor locking in a certain position. Due to their construction, most motors lock at a fullstep position. A full step position is characterized by the position yielded with both coils at identical absolute current. With n-times microstepping, fullstep positions are reached each n steps. The first fullstep position is reached when exactly n/2 steps are done following a driver power-up. The internal

microstep counter shows 128, 384, 640, or 896 when a fullstep position is reached.

The motor will pull into the same step after power up, as long as the rotor position and electrical position differ by up to +-2 fullsteps, given that no external force pulls the motor into a certain direction. An offset of maximum one fullstep is safest.

When powering up the driver, all registers become reset to zero. This also affects the internal position counter. Thus, the position counter will restart from 0 after power up. With the enable pin fixed at “1”, the motor current will pull the motor to this (halfstep) position. With this, several options to keep track of the motor position result:

Table 24. Methods for Position Recovery

ENABLE PIN DRV_ENN	ACTIONS PRIOR TO POWER DOWN	ACTIONS AT POWER-UP
Fixed (to GND)	Keep track of the motor position by counting steps following initial power-up. Prior to power down, move to a position that can be divided by 4 x microstep resolution. At these positions, MSCNT is 0. Store the position.	MSCNT is cleared to 0 automatically. Start moving the motor as desired.
Controlled by CPU	Read out MSCNT and store it (together with the absolute motor position).	Apply a number of steps to restore MSCNT to the stored value prior to enabling the motor driver. number of step pulses = position modulo (4 x microstep resolution) Example: at 32 microstep setting, each step pulse increments MSCNT by 256/32 = 8. Calculate position modulo 128 to yield the required number of steps. Apply 10 steps with DIR = 0 increases MSCNT to a value of 80.

Protections and Driver Diagnostics

The TMC2240 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 TMC2240 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 TMC2240 contains an overvoltage comparator and a trigger output OV to control external switches in terms of excessive supply voltage increase.

Overcurrent Protection

An 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 is trying 3 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 for 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 *TOFF* = 0 in *CHOPCONF* and re-enable the bridges with *TOFF > 0.*

Thermal Protection and Shutdown

The TMC2240 has an internal thermal protection.

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

In addition, the TMC2240 supports ADC-based configurable thermal pre-warning 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 TMC2240 can go in thermal shutdown and the pre-warning 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 could be overheated, are avoided when enabling the short to GND protection. For many applications, the overtemperature pre-warning will indicate an abnormal operation situation and can be used to initiate user warning or power reduction measures like motor current reduction. The thermal shutdown is just an emergency measure and temperature rising to the shutdown level should be prevented by design.

Temperature Measurement

The TMC2240 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 pre-warning and overtemperature 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:

$$\text{ADC\_TEMP} = 7.7 \times \text{TEMP} + 2038$$

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

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 Pin OV

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 will not be sufficient to keep the supply within its limits.

To protect the driver as well as connected circuitry, the TMC2240 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 will chop 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 via register *ADC_VSUPPLY_AIN* as parameter *ADC_VSUPPLY.*

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.

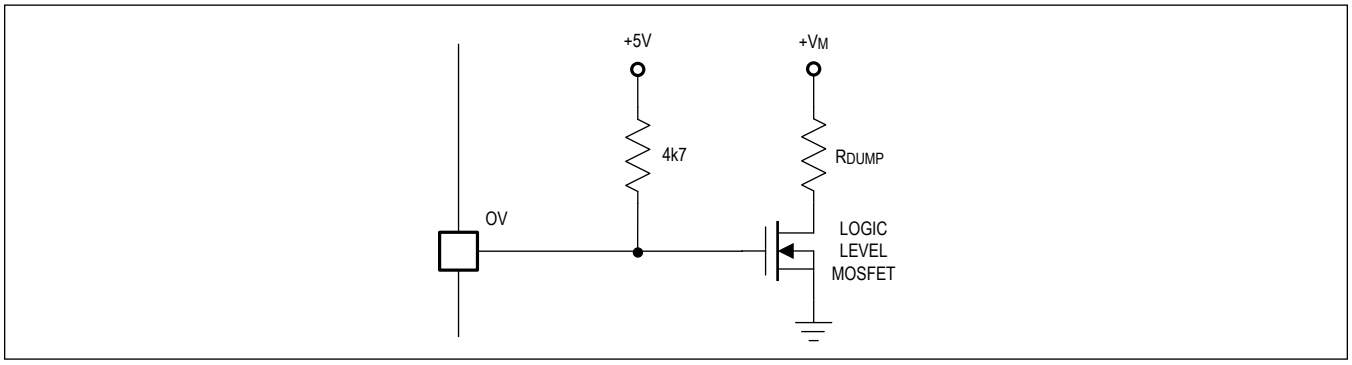


Figure 29. Brake Chopper Circuit Example

Short to GND Protection

The TMC2240 power stages are protected against a short circuit condition by an additional measurement of the current flowing through the high-side MOSFETs. This is important, as most short circuit conditions result from a motor cable insulation defect, e.g. when touching the conducting parts connected to the system ground. The short detection is protected against spurious triggering, e.g., by ESD discharges, by retrying three times before switching off the motor.

Once a short condition is safely detected, the corresponding driver bridge becomes switched off, and the *s2ga* or *s2gb* flag becomes set. In order to restart the motor, the user must intervene by disabling and re-enabling the driver. It should be noted, that the short to GND protection cannot protect the system and the power stages for all possible short events,

as a short event is rather undefined and a complex network of external components may be involved. Therefore, short-circuits should basically be avoided.

Open Load Diagnostics

Interrupted cables are a common cause for systems failing, e.g., when connectors are not firmly plugged. The TMC2240 detects open load conditions by checking, if it can reach the desired motor coil current. This way, also undervoltage conditions, high motor velocity settings or short and overtemperature conditions may cause triggering of the open load flag, and inform the user, that motor torque may suffer. In motor stand still, open load cannot be measured, as the coils might eventually have zero current.

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 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 TMC2240 features a UVLO protection for V_M, V_{CC_IO}, and the charge pump.

UVLO condition on V_M is triggered below 4.15V (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, e.g., due to a wrong capacitor value.

A V_M 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. The UVLO condition is also shown at the DIAG0 pin depending on the configured pin settings.

During a V_{CC_IO} UVLO, no communication with the IC is possible. DIAG0 pin will be active low (open drain).

ESD Protection

The chip has internal ESD protection on every pin.

The TMC2240 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_M Pin).

Anyhow, this is no protection against hot plugging of a motor.

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.

Using an External Clock

When an external clock is available, a frequency of 12MHz to 20MHz is recommended for optimum performance.

The duty cycle of the clock signal is uncritical, as long as minimum high or low input time for the pin is satisfied (see *Electrical Characteristics*).

Up to 20MHz can be used, when the clock duty cycle is 50%.

Make sure, that the clock source supplies clean CMOS output logic levels and steep slopes when using a high clock frequency.

The external clock input is enabled 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 prevent the driver from damage.

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 25. Overview of Register Map

REGISTER	DESCRIPTION
General Configuration Registers	These registers contain ● global configuration ● global status flags ● interface configuration ● and I/O signal configuration
Velocity Dependent Driver Feature Control Register Set	This register group offers registers for ● driver current control ● setting thresholds for CoolStep operation ● setting thresholds for different chopper modes
Direct Mode Registers	This register group offers registers used for the direct coil current control mode.
Encoder Register Set	The encoder register group 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 group offers registers for ● setting/reading out microstep table and counter ● chopper and driver configuration ● CoolStep and StallGuard configuration ● reading out StallGuard values and driver error flags

Register Map

TMC2240

ADDRESS	NAME	MSB							LSB
General Configuration Registers									
0x00	GCONF[23:16]				–	–	–	–	direct_mode
	GCONF[15:8]	stop_enable	small_hysteresis	diag1_pushpull	diag0_pushpull	–	diag1_onstate	diag1_index	diag1_stall
	GCONF[7:0]	diag0_stall	diag0_otpw	diag0_error	shaft	multistep_filt	en_pwm_mode	fast_standstill	–
0x01	GSTAT[7:0]						uv_cp	drv_err	reset
0x02	IFCNT[7:0]	IFCNT[7:0]							
0x03	SLAVECONF[15:8]					SENDDELAY[3:0]			
	SLAVECONF[7:0]	SLAVEADDR[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_ENN	ENCA	ENCB	DIR	STEP
0x0A	DRV_CONF[23:16]			–	–	–	–	–	–
	DRV_CONF[15:8]	–	–	–	–	–	–	–	–
	DRV_CONF[7:0]	–	–	SLOPE_CONTROL[1:0]		–	–	CURRENT_RANGE[1:0]	
0x0B	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[7:0]	TPOWERDOWN[7:0]							
0x12	TSTEP[23:16]					TSTEP[19:16]			
	TSTEP[15:8]	TSTEP[15:8]							
	TSTEP[7:0]	TSTEP[7:0]							
0x13	TPWMTHRS[23:16]					TPWMTHRS[19:16]			
	TPWMTHRS[15:8]	TPWMTHRS[15:8]							
	TPWMTHRS[7:0]	TPWMTHRS[7:0]							
0x14	TCOOLTHRS[23:16]					TCOOLTHRS[19:16]			
	TCOOLTHRS[15:8]	TCOOLTHRS[15:8]							
	TCOOLTHRS[7:0]	TCOOLTHRS[7:0]							
0x15	THIGH[23:16]					THIGH[19:16]			
	THIGH[15:8]	THIGH[15:8]							
	THIGH[7:0]	THIGH[7:0]							

ADDRESS	NAME	MSB							LSB
Direct Mode Register									
0x2D	DIRECT_MODE[31:24]	–	–	–	–	–	–	–	DIRECT_COIL_B[8]
	DIRECT_MODE[23:16]	DIRECT_COIL_B[7:0]							
	DIRECT_MODE[15:8]	–	–	–	–	–	–	–	DIRECT_COIL_A[8]
	DIRECT_MODE[7:0]	DIRECT_COIL_A[7:0]							
Encoder Registers									
0x38	ENCMODE[15:8]						enc_sel_decimal	–	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[7:0]							–	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]							
ADC Registers									
0x50	ADC_VSUPPLY_AIN[31:24]				ADC_AIN[12:8]				
	ADC_VSUPPLY_AIN[23:16]	ADC_AIN[7:0]							
	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]							

ADDRESS	NAME	MSB							LSB
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]							
	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[15:8]							MSCNT[9:8]	

ADDRESS	NAME	MSB							LSB
	MSCNT[7:0]	MSCNT[7:0]							
0x6B	MSCURACT[23:16]							CUR_A[1:0]	
	MSCURACT[15:8]	–	–	–	–	–	–	–	CUR_B[8]
	MSCURACT[7:0]	CUR_B[7:0]							
0x6C	CHOPCONF[31:24]	diss2vs	diss2g	dedge	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]			
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]							
	PWMCONF[7:0]	PWM_OFS[7:0]							
0x71	PWM_SCALE[23:16]								PWM_SCALE_AUTO[0]
	PWM_SCALE[15:8]	–	–	–	–	–	–	PWM_SCALE_SUM[9:8]	
	PWM_SCALE[7:0]	PWM_SCALE_SUM[7:0]							
0x72	PWM_AUTO[15:8]	–	–	–	–	–	–	–	–
	PWM_AUTO[7:0]	PWM_OFS_AUTO[7:0]							
0x74	SG4_THRS[7:0]	SG4_THRS[7:0]							
0x75	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				20	19	18	17	16
Field				–	–	–	–	direct_mode
Reset				–	–	–	–	0x0
Access Type				–	–	–	–	Write, Read

BIT	15	14	13	12	11	10	9	8
Field	stop_enable	small_hysteresis	diag1_pushpull	diag0_pushpull	–	diag1_onstate	diag1_index	diag1_stall
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	diag0_stall	diag0_otpw	diag0_error	shaft	multistep_filt	en_pwm_mode	fast_standstill	–
Reset	0x0	0x0	0x0	0x0	0x1	0x0	0x0	–
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	–

BITFIELD	BITS	DESCRIPTION	DECODE
direct_mode	16	Enable direct motor phase current control via serial interface.	0x0: Normal operation 0x1: Motor coil currents and polarity directly programmed via serial interface: Register *DIRECT_MODE* (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 will work 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_pushpull	13	DIAG1 output type configuration.	0x0: DIAG1 is open collector output (active low) 0x1: Enable DIAG1 push pull output (active high)
diag0_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_onstate	10	DIAG1 output configuration.	0x0: Disable DIAG1active on chopper on. 0x1: *diag1_onstate* Enable DIAG1 active when chopper is on (for the coil which is in the second half of the fullstep)

BITFIELD	BITS	DESCRIPTION	DECODE
diag1_index	9	DIAG1 output configuration.	0x0: Disable DIAG1 active on index position. 0x1: *diag1_index* Enable DIAG1 active on index position (microstep look up table position 0)
diag1_stall	8	DIAG1 output configuration.	0x0: *diag1_stall* Motor stall not indicated at DIAG1 0x1: *diag1_stall* Enable DIAG1 active on motor stall (set *TCOOLTHRS* before using this feature)
diag0_stall	7	DIAG0 output configuration.	0x0: *diag0_stall* Motor stall not indicated at DIAG0 0x1: *diag0_stall* Enable DIAG0 active on motor stall (set *TCOOLTHRS* before using this feature)
diag0_otpw	6	DIAG0 output configuration.	0x0: Disable DIAG0 active on driver over temperature prewarning 0x1: Enable DIAG0 active on driver over temperature prewarning (*otpw*)
diag0_error	5	DIAG0 output configuration. DIAG0 always shows the reset-status, i.e. is active low during reset condition.	0x0: Disable DIAG0 active on driver errors. 0x1: Enable DIAG0 active on driver errors: Over temperature (*ot*), short to GND (*s2g*), undervoltage chargepump (*uv_cp*)
shaft	4	Change motor direction / direction sign	0x0: Default motor direction 0x1: Inverse motor direction
multistep_filt	3	Enable step input filtering for StealthChop2	0x0: no StealthChop2 0x1: StealthChop2 voltage PWM mode enabled (depending on velocity thresholds). Switch from off to on state while in stand-still and at IHOLD= nominal IRUN current, only.
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 stand-still 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		2	1	0
Field		uv_cp	drv_err	reset
Reset		0x0	0x0	0x1
Access Type		Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read

BITFIELD	BITS	DESCRIPTION	DECODE
vm_uvlo	4	1: VM undervoltage has occured since last reset.	

BITFIELD	BITS	DESCRIPTION	DECODE
register_reset	3		0x0: normal operation 0x1: Indicates that the registermap has been reset. All registers have been cleared to reset values.
uv_cp	2	Charge pump undervoltage condition flag	0x0: normal operation 0x1: Indicates an undervoltage on the charge pump. The driver is disabled during undervoltage. This flag is latched for information.
drv_err	1	Driver error flag	0x0: normal operation 0x1: Indicates, that the driver has been 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	0x0: normal operation 0x1: Indicates that the IC has been 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	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.

SLAVECONF (0x3)

BIT					11	10	9	8
Field					SENDDELAY[3:0]			
Reset					0x0			
Access Type					Write, Read			
BIT	7	6	5	4	3	2	1	0
Field	SLAVEADDR[7:0]							
Reset	0x0							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
SENDDELAY	11:8	SWUART Slave Configuration	0x0: 8 bit times (not allowed with multiple slaves) 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
SLAVEADDR	7:0	*SLAVEADDR*: These eight bits set the address of unit 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: 0-254 (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	DIR	STEP
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 via 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
DIR	1	
STEP	0	

DRV_CONF (0xA)

BIT			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	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 in order to adapt a drive to a certain motor type. This value should be chosen before tuning other settings, because it also influences chopper hysteresis. This value is just intended for finetuning 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)

Test Reg

BIT					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 * 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:* Choose sense resistors in a way, that normal IRUN is 16 to 31 for best microstep performance.
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 stand still.

TPOWERDOWN (0x11)

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		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 stand still. All TSTEP related thresholds use a hysteresis of 1/16 of the compare value to compensate for jitter in the clock or the step frequency. The flag *small_hysteresis* modifies the hysteresis to a smaller value of 1/32. (*Txxx 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.

TPWMTHRS (0x13)

BIT					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

TCOOLTHRS (0x14)

BIT					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 stall output signal become enabled when exceeding this velocity. It becomes disabled again once the velocity falls below this threshold. *TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: ● CoolStep is enabled, if configured *TCOOLTHRS* ≥ *TSTEP* ● Stall output signal (DIAG0/1) is enabled, if configured

THIGH (0x15)

BIT					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 2-3 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 becomes switched over to fullstep mode stall detection.

DIRECT_MODE (0x2D)

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

BITFIELD	BITS	DESCRIPTION
DIRECT_COIL_B	24:16	When direct mode in GCONF is selected: Signed coil B current

BITFIELD	BITS	DESCRIPTION
DIRECT_COIL_A	8:0	When direct mode in GCONF is selected: Signed coil A current

ENCMODE (0x38)

BIT						10	9	8
Field						enc_sel_decimal	–	clr_enc_x
Reset						0x0	–	0x0
Access Type						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
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
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		1	0
Field		–	n_event
Reset		–	0x0
Access Type		–	Write 1 to Clear, Read

BITFIELD	BITS	DESCRIPTION	DECODE
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

ADC_VSUPPLY_AIN (0x50)

BIT				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 on ADC_AIN pin in integer
ADC_VSUPPLY	12:0	Actual Value of voltage on VS (filtered with low pass filter), update rate: each 2048 clocks

ADC_TEMP (0x51)

BIT				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

OTW_OV_VTH (0x52)

BIT				28	27	26	25	24
Field				OVERTEMPPREWARNING_VTH[12:8]				
Reset				0d2962				
Access Type				Write, Read				
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	OVERTEMPPREWARNING_VTH[7:0]							
Reset	0d2962							
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 Overtemperatureprewarning will be triggered (Reset: 0xB92 equals 120°C)
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							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_B*. *CUR_A* uses an offset of 256 (2 phase motor). *Hint:* Move to a position where *MSCNT* is zero before re-initializing *MSLUTSTART* or *MSLUT* and *MSLUTSEL*.

MSCURACT (0x6B)

BIT							17	16
Field							CUR_A[1:0]	
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 (co-sine 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	dedge	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		–			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
dedge	29	enable double edge step pulses	0x0: disabled 0x1: Enable step impulse at each step edge to reduce step frequency requirement.
intpol	28	interpolation to 256 microsteps	0x0: no interpolation 0x1: The actual microstep resolution (*MRES*) becomes extrapolated to 256 microsteps for smoothest motor operation (useful for STEP/DIR operation, only)
MRES	27:24	Micro step resolution selection 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]	0x0: Native 256 microstep setting. 0x1: 128 0x2: 64 0x3: 32 0x4: 16 0x5: 8 0x6: 4 0x7: 2 0x8: FULLSTEP 0x9: unused

BITFIELD	BITS	DESCRIPTION	DECODE
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 *TOFF* setting automatically becomes doubled during high velocity operation in order 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.	0x0: 16 clocks 0x1: 24 clocks 0x2: 36 clocks 0x3: 54 clocks
chm	14	chopper mode	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 *with chm*=1: *disfdcc*=1 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*	

BITFIELD	BITS	DESCRIPTION	DECODE
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: H*STRT* 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	*TOFF* 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								
								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 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*) 0x1: 1/4 of current setting (*IRUN*)
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	

BITFIELD	BITS	DESCRIPTION	DECODE
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	

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 stand still 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.

BITFIELD	BITS	DESCRIPTION	DECODE
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 (*TOFF*=0) or by the 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 (*TOFF*=0) or by the ENN input.
otpw	26	overtemperature pre-warning flag	0x0: normal operation 0x1: Overtemperature pre-warning threshold is exceeded. The overtemperature pre-warning 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 fullstep stall (in fullstep 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
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 (*TOFF*=0) or by the ENN input.

BITFIELD	BITS	DESCRIPTION	DECODE
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 (*TOFF*=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 stand still 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 in order to detect a stall. *SG_RESULT* is used as a base for CoolStep operation, by comparing it to a programmable upper and a lower limit. It is not applicable in StealthChop2 mode. StallGuard2 works best with microstep operation. 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 a reading the chopper on-time. As the motor heats up, its coil resistance rises and the chopper on-time increases. For StallGuard4 specifics, please refer *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.	
FREEWHEEL	21:20	Allows different standstill modes Stand still option when motor current setting is zero (*I_HOLD*=0). %00: Normal operation %01: Freewheeling %10: Coil shorted using LS drivers %11: Coil shorted using HS drivers	

BITFIELD	BITS	DESCRIPTION	DECODE
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 in order to a) detect standstill b) wait > 128 chopper cycles at *IRUN* and c) regulate *PWM_OFS_AUTO* so that -1 < *PWM_SCALE_AUTO* < 1 2. Motor running and 1.5 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}	
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.*	

BITFIELD	BITS	DESCRIPTION	DECODE
PWM_OFS	7:0	User defined PWM amplitude offset (0-255) related to full motor current (*CS_ACTUAL*=31) in stand still. (*Reset default=30*) Use *PWM_OFS* as initial value for automatic scaling to speed up the automatic tuning process. To do this, set *PWM_OFS* to the determined, application specific value, with *pwm_autoscale*=0. Only afterwards, set *pwm_autoscale*=1. Enable StealthChop2 when finished. *PWM_OFS* = 0 will disable scaling down motor current below a motor specific lower measurement threshold. This setting should only be used under certain conditions, i.e. when the power supply voltage can vary up and down by a factor of two or more. It prevents the motor going out of regulation, but it also prevents power down below the regulation limit. *PWM_OFS* > 0 allows automatic scaling to low PWM duty cycles even below the lower regulation threshold. This allows low (standstill) current settings based on the actual (hold) current scale (register *IHOLD_IRUN*).	

PWM_SCALE (0x71)

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

BIT								16
Field								PWM_SCALE_AUTO[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	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	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 via 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 the double of this threshold. A stall is signaled with *SG_RESULT ≤ SG4_THRS x 2*

SG4_RESULT (0x75)

BIT							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 will 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

BITFIELD	BITS	DESCRIPTION
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 capacitors for filtering the power supply. The capacitors need to cope with the current ripple cause by chopper operation. A minimum capacity of 100µF near the driver is recommended for best performance. Current ripple in the supply capacitors also depends on the power supply internal resistance and cable length. VCC_IO must be supplied from an external source, e.g., a low drop 3.3V regulator.

Place all filter capacitors as close as possible to the related IC pins. Use a solid common GND for all GND connections. Connect VDD1V8 filtering capacitor directly to VDD1V8 pin. Low ESR electrolytic capacitors are recommended for VS filtering.

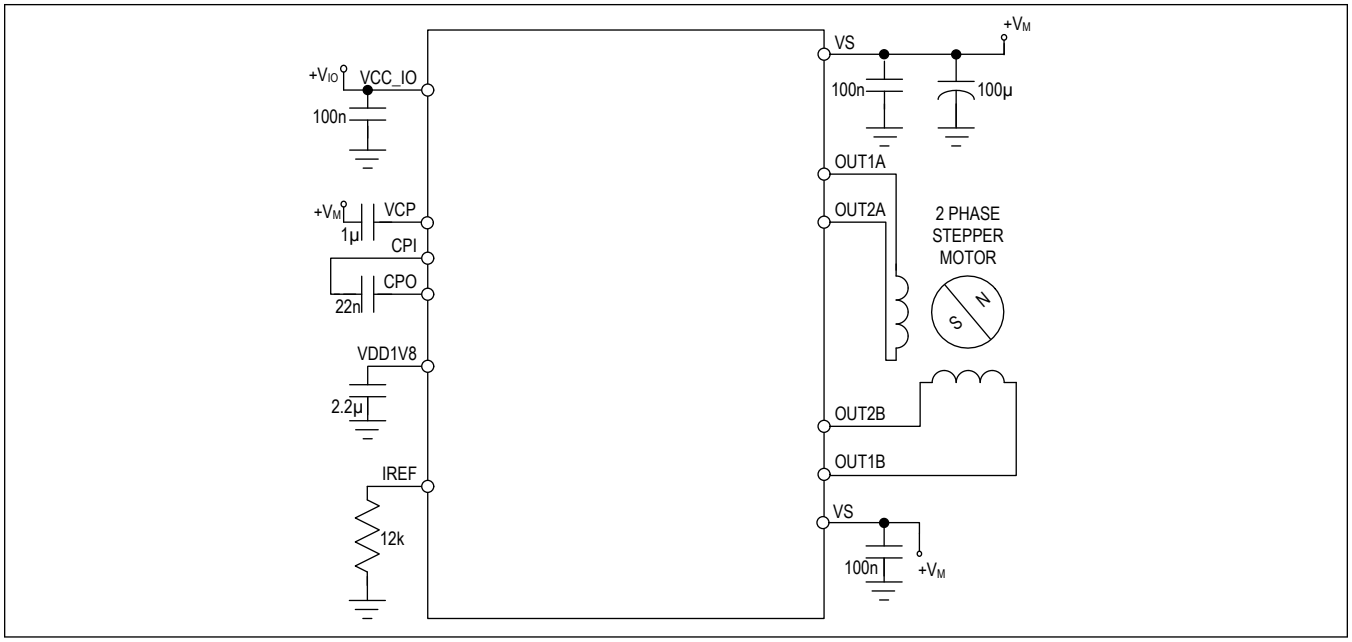


Figure 30. Standard Application Circuit

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 will heat up the PCB cooling infrastructure also, if operated at an increased duty cycle. This in turn leads to a further increase of driver temperature. An increase of temperature by about 100°C increases MOSFET resistance by roughly 50%. This is a typical behavior of MOSFET switches. Therefore, under high duty cycle, high load conditions, thermal characteristics have to be carefully taken into account, especially when increased environment temperatures are to be supported. Refer the thermal characteristics and the layout examples as well.

As a thumb rule, thermal properties of the PCB design may become critical at or above 1.5A RMS motor current for increased periods of time. Keep in mind 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.

Typical Application Circuits (continued)

Driver Protection and EME Circuitry

Some applications have to cope with ESD events caused by motor operation or external influence. Despite ESD circuitry within the driver chips, ESD events occurring during operation can cause a reset or even a destruction of the motor driver, depending on their energy. Especially plastic housings and belt drive systems tend to cause ESD events 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 will bring more benefit concerning ESD suppression, but cause additional current flow in each chopper cycle, and thus increase driver power dissipation, especially at high supply voltages. The values shown are example values – they might be varied between 100pF and 1nF. The capacitors also dampen high frequency noise injected from digital parts of the application PCB circuitry and thus reduce electromagnetic emission.

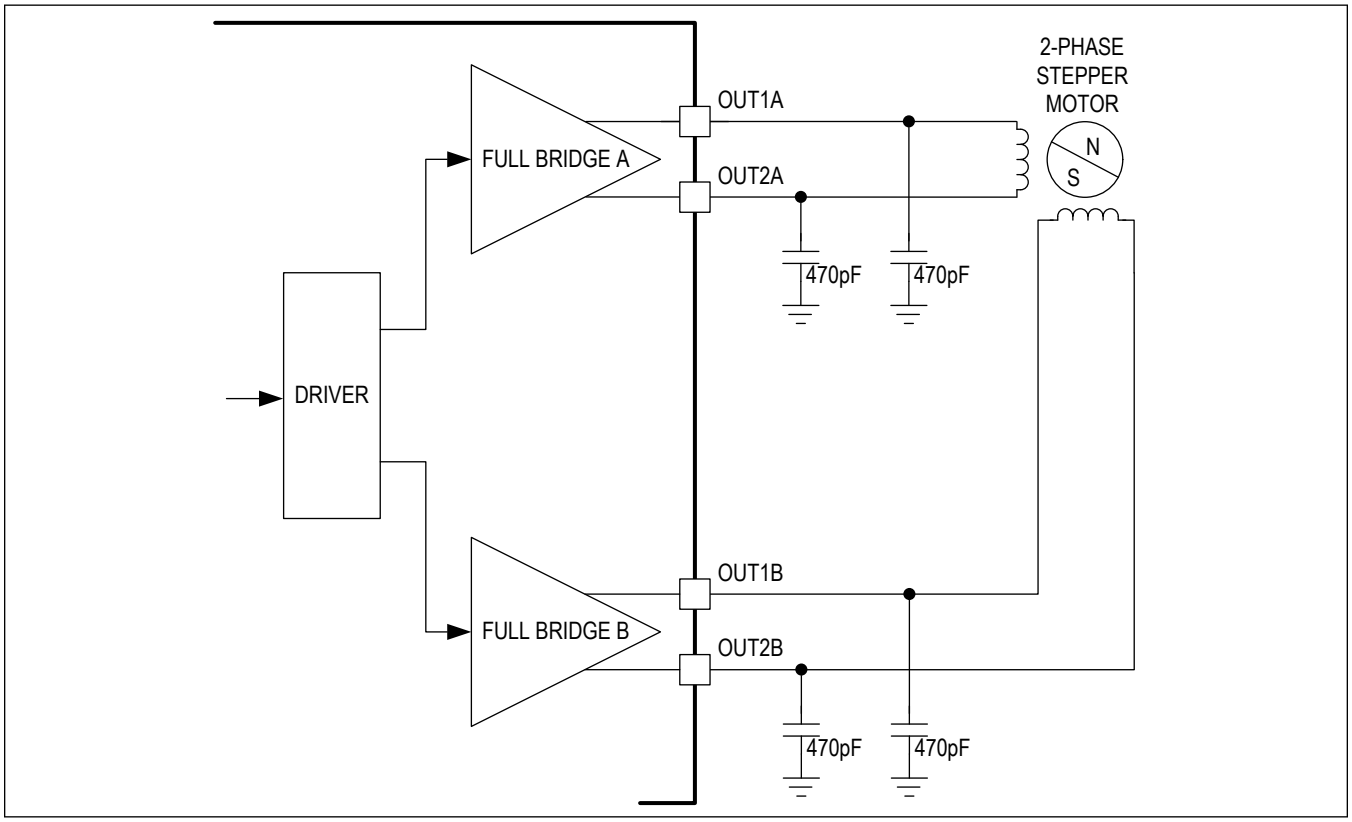


Figure 31. Simple ESD Enhancement

A more elaborate scheme uses LC filters to de-couple 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 ESD voltage. Fit the varistors to the supply voltage rating. The SMD inductivities will conduct full motor coil current and need to be selected accordingly.

Typical Application Circuits (continued)

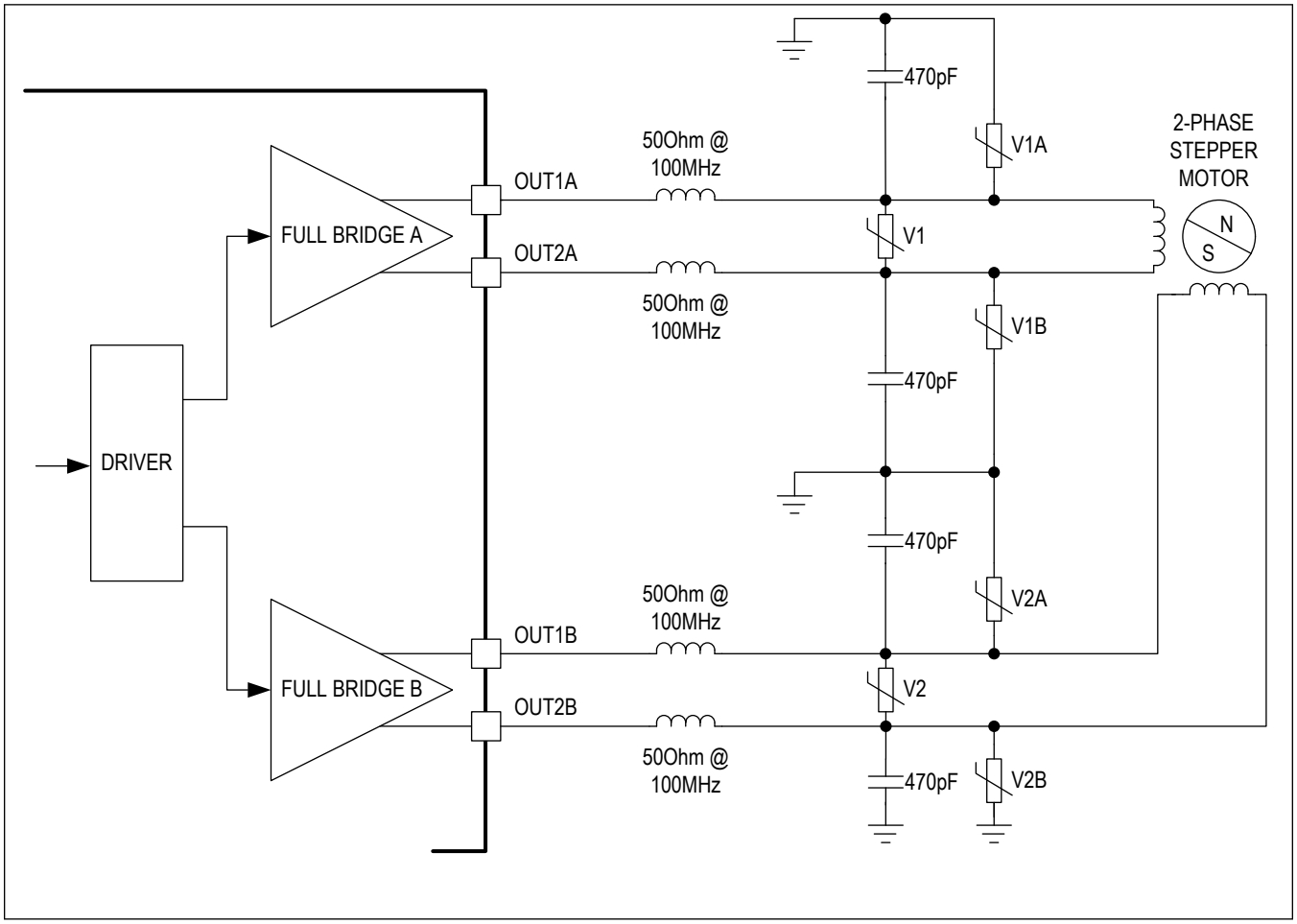


Figure 32. Elaborate Motor Output Protection

Ordering Information

PART NUMBER	TEMPERATURE RANGE	PIN-PACKAGE
TMC2240ATJ+	-40°C to +125°C	32 TQFN - 5mm x 5mm
TMC2240ATJ+T	-40°C to +125°C	32 TQFN - 5mm x 5mm
TMC2240AUU+*	-40°C to +125°C	38 TSSOP-EP 4.4mm x 9.7mm
TMC2240AUU+T*	-40°C to +125°C	38 TSSOP-EP 4.4mm x 9.7mm

** Future product—contact factory for availability.*

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

T Denotes tape-and-reel.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	07/22	Release for Market Intro	—