65V/4.2A, Fully-Integrated, Smart, Three-Phase Stepper Motor Driver

TMC3241

General Description

The TMC3241 is a miniaturized, smart, high-power, three-phase stepper motor controller and driver IC with SPI and extensive diagnostic capabilities. Highest integration, high energy efficiency, and a small form factor enable miniaturized, scalable, and cost-effective motor drive solutions perfectly suited for Nema 17, Nema 23, and even Nema 24 steppers, or for three-phase PMSM motors in open loop control.

The TMC3241 allows operation using step and direction input as well as through its integrated motion controller. It combines a smart 1024 sine wave step motor driver for up to 171 microsteps, and three integrated 65V, 4.2A RMS, 6A peak H-bridges plus non-dissipative integrated current sensing (ICS). ICS eliminates bulky external sense resistors, resulting in space and power savings.

The integrated power MOSFETs' low impedance of ~40mΩ ensures high efficiency and low heat generation. Abundant diagnostics include temperature and voltage ADC, and a full set of protection functions like short protection and thermal shutdown.

It comes in a compact, near chip-scale, thermally-optimized, 30-pin, 6mm × 6mm, FC2QFN package.

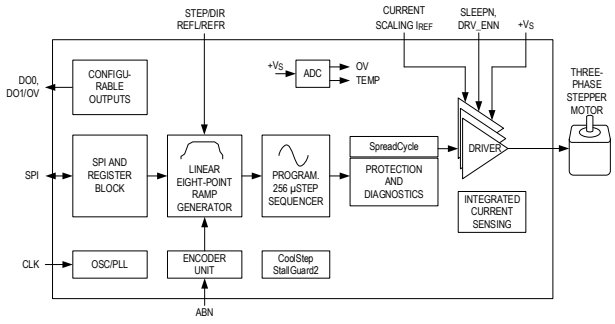
Key Applications

- Three-Phase Stepper and Open Loop BLDC/PMSM
- Optical Instrumentation
- Lab Automation, IVD
- Robotics and Factory Automation
- 3D Printers, Label Printers
- Medical File Printer
- Camera Gimbals, Surveillance Cameras
- POS
- Pumps and Valve Control
- Stage Lighting, Antenna Positioning

Benefits and Features

- Voltage Range: 4.5V to 65V DC
- Low $R_{DS(ON)}$ (HS + LS): 0.08Ω Typical (T_A = 25°C)
- Current Ratings per Half-Bridge (Typical at 25°C):
  - I_{RMS} = 4.2A_{RMS} (6A Full-Scale Sine Peak)
  - I_{MAX} = 9.0A (Bridge Peak Current)
- STEP/DIR interface with MicroPlyer interpolation
- Eight-Point Ramp Generator Option
- SPI for Configuration, Control, and Diagnostics
- Fully-Integrated Lossless Current Sensing
- (Periodic) Position Compare Function
- Incremental Encoder
- Highest Resolution of 1024 Sine Wave Steps
- Programmable Sine Wave Table to Match Motor
- SpreadCycle Highly Dynamic Motor Chopper
- StallGuard2 Sensorless Load Detection
- CoolStep Current Control
- Chip Temperature and Supply Voltage Measurement
- Full Protection and Diagnostics
- Overvoltage Protection Output

Simplified Block Diagram



19-102175; Rev 0; 10/26

© 2026 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.

TABLE OF CONTENTS

General Description 1
Key Applications 1
Benefits and Features 1
Simplified Block Diagram 1
Absolute Maximum Ratings 7
Electrical Characteristics 7
Package Information 12
FC2QFN30, 6mm × 6mm 12
Package Marking Specification 12
Pin Descriptions 12
Pin Configurations 14
Functional Diagrams 15
Detailed Description 16
Principles of Operation 16
Full Featured Motion Controller and Driver 16
Key Concepts 17
Control Interface 17
Step and Direction Interface 17
Automatic Standstill Power Down 17
Mechanical Load Sensing with StallGuard2 18
Load Adaptive Current Control with CoolStep 18
Real Time On-Chip Scope Interface 18
Encoder Interface 18
Serial Peripheral Interface (SPI) 18
SPI Datagram Structure 18
Selection of Write/Read (WRITE_notREAD) 19
SPI Status Bits Transferred with Each Datagram Read Back 19
Data Alignment 20
SPI Signals 20
SPI Timing 20
Step and Direction Interface 20
Timing 21
Changing Resolution 21
MicroPlyer Step Interpolator and Standstill Detection 22
SpreadCycle 22
SpreadCycle Chopper 24
Direct Mode 25

StallGuard2 Load Measurement 25
Tuning StallGuard2 Threshold SGT 26
Variable Velocity Limits *TCOOLTHRS* and *THIGH* 27
Small Motors with High Torque Ripple and Resonance 28
Temperature Dependence of Motor Coil Resistance 28
Accuracy and Reproducibility of StallGuard2 Measurement 28
StallGuard2 Update Rate and Filter 28
Detecting a Motor Stall 28
Homing with StallGuard2 29
Limits of the StallGuard2 Operation 29
CoolStep Load Adaptive Current Scaling 29
Setting Up for CoolStep 29
Tuning CoolStep 31
Response Time 31
Low Velocity and Standby Operation 31
Integrated Current Sense 31
Setting the Full-Scale Current Range 31
Slope Control 33
Velocity-Based Mode Control 33
Ramp Generator 35
Enabling the internal Motion Controller 35
Real-World Unit Conversion 35
Motion Profiles 36
Ramp Mode 36
Eight-Point Ramp 38
Start and Stop Velocity 38
Velocity Mode 41
Early Ramp Termination 41
Application Example: Joystick Control 42
Reference Switches 42
Implementing a Homing Procedure: 43
Homing with a Third Switch 43
Virtual Reference Switches 44
Ramp Generator Response Time 44
External STEP/DIR Driver 44
Position Compare Functions 45
ABN Incremental Encoder Interface 45
N Signal 45

The Encoder Counter X_ENC 46
The Register ENC_STATUS 46
The Encoder Constant ENC_CONST 46
Setting the Encoder to Match Motor Resolution 46
Sine Wave Lookup Table 47
Microstep Table 47
Matching the Phase Shift to the Motor 48
Reset, Disable/Stop, and Power Down 48
Emergency Stop 48
External Reset and Sleep Mode 48
Diagnostics and Protections 49
Diagnostic Outputs 49
Overcurrent Protection 50
Thermal Protection and Shutdown 50
Temperature Measurement 51
Chip Temperature Measurement 51
Overvoltage Protection and OV Output 51
Short Protection (Short-to-GND and Short-to-V_S) 52
Open-Load Diagnostics 52
Undervoltage Lockout Protection 52
ESD Protection 53
Clock Oscillator and Clock Input 53
Internal PLL Block 53
Using the Internal Clock 55
Using an External Clock 55
Quick Configuration Guides 56
PLL Initialization Guide 56
Current Setting Guide 57
SpreadCycle Configuration 58
CoolStep in Combination with SpreadCycle 59
Moving the Motor Using the Motion Controller 60
Initialization and Configuration Example 63
Design for Sustainability 63
Register Map 64
GCR Register Overview 64
Ordering Information 109
Revision History 110

LIST OF FIGURES

Figure 1. Package Marking (Top View) 12
Figure 2. TMC3241 Pinout with Bump Side Down 14
Figure 3. Block Diagram 15
Figure 4. Block Diagram with Typical External Components 16
Figure 5. Automatic Motor Current Control at Standstill and Ramp-Up 18
Figure 6. SPI Timing Diagram 20
Figure 7. STEP and DIR Signal Timing 21
Figure 8. STEP and DIR Signal Input Filter Structure 21
Figure 9. MicroPlyer Microstep Interpolation with Rising STEP Frequency (Example: 16 to 256) 22
Figure 10. Typical Chopper Decay Phases (Sense Resistor is Only Symbolic for Current Measurement in Low-Side Path) 23
Figure 11. SpreadCycle Chopper Scheme Showing the Coil Current During a Chopper Cycle 25
Figure 12. Function Principle of StallGuard2 26
Figure 13. Example: Optimum SGT Setting and StallGuard2 Reading with an Example Motor 28
Figure 14. CoolStep Adapts Motor Current to the Load 30
Figure 15. Choice of Velocity-Dependent Modes 34
Figure 16. Ramp Generator Velocity Trace Showing Second Move into Negative Direction 37
Figure 17. Illustration of Optimized Motor Torque Usage with the Ramp Generator 38
Figure 18. Eight-Point Ramp with VMAX Not Reached Due to Too Low Distance 39
Figure 19. V2 Not Reached and No AMAX and DMAX Phase Due to Low Distance 39
Figure 20. V1 Not Reached Due to Low Distance 40
Figure 21. TVMAX Not Kept Due to Low Distance 40
Figure 22. Eight-Point Ramp Examples with On-the-Fly Target Position Change 41
Figure 23. Using Reference Switches (Example) 42
Figure 24. Virtual Stop Switches and Limit Visualization 44
Figure 25. Outline of ABN Signals of an Incremental Encoder 46
Figure 26. LUT Programming Example 48
Figure 27. Schematic of DO0 and DO1 Outputs 50
Figure 28. Brake Chopper Circuit Example 52
Figure 29. Block Diagram of PLL 53
Figure 30. Quick Configuration Guide for Starting the PLL 56
Figure 31. Quick Configuration Guide for Current Setting 57
Figure 32. Quick Configuration Guide for SpreadCycle 58
Figure 33. Quick Configuration Guide for CoolStep with SpreadCycle 59
Figure 34. Quick Configuration Guide for Moving a Motor in Velocity Mode 60
Figure 35. Quick Configuration Guide for Moving a Motor to a Target Position 61
Figure 36. Quick Configuration Guide for Motion Ramp Parameter Settings 62

LIST OF TABLES

Table 1. Packaging Marking Description 12
Table 2. TMC3241 Feature Overview 17
Table 3. SPI Datagram Structure 19
Table 4. SPI Read/Write Example Flow 19
Table 5. SPI_STATUS – Status Flags Transmitted with SPI Access in Bits 39 to 32 19
Table 6. Parameters Controlling SpreadCycle 23
Table 7. StallGuard2-Related Parameters 26
Table 8. Procedure to Tune StallGuard 27
Table 9. CoolStep Critical Parameters 29
Table 10. CoolStep Additional Parameters and Status Information 30
Table 11. Full-Scale Current Register Configuration for R_{REF} = 12kΩ 32
Table 12. Parameters Controlling the Motor Current 32
Table 13. Velocity-Based Mode Control Parameters 34
Table 14. Ramp Generator Parameters vs. Units 36
Table 15. Implementing a Homing Procedure 43
Table 16. Homing with a Third Switch Procedure 43
Table 17. X_COMPARE_REPEAT Options and Periodic Pulse Behavior 45
Table 18. Encoder Example Settings for a 300 Fullstep Motor with 171 Microsteps 47
Table 19. PLL Control Register PLL Parameters 54

Absolute Maximum Ratings

V_S to GND ..... -0.3 V to 70 V
V_{DD} to GND ..... -0.3 V to Min. (2.2, V_S + 0.3)V
OUTU, OUTV, OUTW ..... -0.3 V to V_S + 0.3 V
V_{CP} to GND ..... V_S - 0.3 V to Min. (74, V_S + 6)V
CPO to GND ..... V_S - 0.3 V to V_{CP} + 0.3V
CPI to GND ..... -0.3 V to V_S + 0.3V
IREF to GND ..... -0.3 V to Min. (2.2, V_{DD} + 0.3)V
V_{CC_IO} to GND ..... -0.3 V to 5.5 V
Logic Input/Output Voltage to GND .... -0.3 V to V_{CC_IO} + 0.3 V
SLEEPN to GND ..... -0.3V to 5.5V
Operating Temperature Range ..... -40°C to 125°C
Junction Temperature ..... +160°C
Storage Temperature Range ..... -40°C to 125°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.

Electrical Characteristics

(V_S = 4.5V to 65V, R_{REF} = 12kΩ, typical values assume T_A = 25°C and V_S = 48V, limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
POWER SUPPLY							
Supply Voltage Range	V_S			4.5		65	V
Sleep Mode Current Consumption	I_{VS}	T_J = 25°C	V(SLEEPN) = 0			10	µA
Driver Off Current Consumption	I_{VS}	V(SLEEPN) = 1, V_{CC_IO} = 0V				1.5	mA
Quiescent Current Consumption	I_{VS}	V(SLEEPN) = 1				1.7	mA
1.8V Regulator Output Voltage	V_{VDD}	V_S = 4.5V, I_{LOAD} = 20mA			1.8		V
V_{DD} Current Limit	$IV18_{LIM}$			70			mA
Charge Pump Voltage	V_{CP}				V_S + 2.7		V
Logic I/O Supply Voltage Range	V_{CC_IO}			2.3		5.5	V
V_{CC_IO} UVLO Rising Threshold	UVLOVCCR			2.0	2.1	2.2	V
V_{CC_IO} UVLO Falling Threshold	UVLOVCCF			1.9	2.0	2.1	V
V_{CC_IO} UVLO Hysteresis	UVLOVCCH				100		mV
Sleep Mode Current Consumption	I_{VCC_IO}	V(SLEEPN) = 0			0	5	µA
Quiescent Current Consumption	I_{VCC_IO}	After LDO switchover. 100mA is the current limit of the internal LDO.				100	mA
LOGIC LEVEL INPUTS-OUTPUTS							

(V_S = 4.5V to 65V, R_{REF} = 12kΩ, typical values assume T_A = 25°C and V_S = 48V, limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Input Voltage Level (High)	V_{IH}			0.7 × V_{CC_IO}			V
Input Voltage Level (Low)	V_{IL}					0.3 × V_{CC_IO}	V
Input Hysteresis	V_{HYS}				0.1 × V_{CC_IO}		V
Internal Pullup/Pulldown Resistance	R_{PULL}	To GND or to V_{CC_IO}		60	100	140	KΩ
Input Leakage	IN_{Leak}	Inputs without pullup/pulldown resistance		-1		1	µA
Open-Drain 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) = 5V		-1		1	µA
SLEEPN Voltage Level High	VIH_{SLEEPN}			0.9			V
SLEEPN Voltage Level Low	VIL_{SLEEPN}					0.6	V
SLEEPN Pulldown Input Resistance	RPD_{SLEEPN}			0.85	1.5		MΩ
OUTPUT SPECIFICATIONS							
Output ON-Resistance Low Side	$R_{DS(ON)}$ (LS)	Full-scale bits = 11			0.036	0.070	Ω
		Full-scale bits = 10			0.047	0.095	
		Full-scale bits = 01			0.070	0.140	
		Full-scale bits = 00			0.138	0.280	
Output ON-Resistance High Side	$R_{DS(ON)}$ (HS)				0.040	0.080	Ω
Output Leakage	I_{LEAK}	Driver OFF	T = +125°C	-80		80	µA
			T = +25°C	-5		5	
Dead Time	t_{DEAD}				100		ns
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 = 11		10.0			A
		Full-scale bits = 10		7.5			

(V_S = 4.5V to 65V, R_{REF} = 12kΩ, typical values assume T_A = 25°C and V_S = 48V, limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
		Full-scale bits = 01	5.5			
		Full-scale bits = 00	2.8			
Overcurrent Protection Blanking Time	TOCP		0.9	1.5	2.5	µs
UVLO Threshold on V_S	UVLO	V_S rising	3.85	4	4.15	V
UVLO Threshold on V_S Hysteresis	UVLOHYS			110		mV
Thermal Protection Threshold Temperature	TSD	For temperature rising		165		°C
Thermal Protection Temperature Hysteresis				20		°C
CURRENT REGULATION						
I_{REF} Pin Resistor	R_{REF}			12		kΩ
I_{REF} Output Voltage	V_{REF}		0.882	0.9	0.918	V
Full-Scale Current Constant	KIFS	IFS = 1.5A		18		A × kΩ
Full-Scale Current Constant	KIFS	IFS = 3A		36		A × kΩ
Full-Scale Current Constant	KIFS	IFS = 4.5A		54		A × kΩ
		IFS = 6A		72		
Current Trip Regulation Accuracy	DITRIP1	ITRIG from 30% to 100% FS	-5		5	%
	DITRIP2	ITRIG from 15% to 30% FS	-10		10	
	DITRIP3	ITRIG from 10% to 15% FS	-15		15	
Current Trip Regulation Matching				0.7		%
FUNCTIONAL TIMINGS						
SLEEP Time	t_{SLEEP}	SLEEPN = 0 to OUTxx three state			50	µs
Wake-Up Time from Sleep	t_{WAKE}	SLEEPN = 1 to normal operation		2		ms
Enable Time	t_{ENA}	Time from DRV_ENN pin falling edge to driver on			1	µs
Disable Time	t_{DIS}	Time from DRV_ENN pin rising edge to driver off			1.4	µs
CLOCK OSCILLATOR AND INPUT						
Internal Oscillator Frequency	f_{CLKOSC}	Including variation over temperature	15.3	16	16.7	MHz

(V_S = 4.5V to 65V, R_{REF} = 12kΩ, typical values assume T_A = 25°C and V_S = 48V, limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
External Clock Frequency	f_{CLK}	n = 1 … 32		1	n × 1	32	MHz
External Clock High/Low Time	t_{CLKL}			15			ns
External Clock Timeout Detection in Cycles of Internal fCLKOSC		Checked for lower/upper limit.			32		cycles
Allowed Tolerance for External Clock (no clk_1M0_tmo)	f_{CLKTMO}	Related to setting of CLOCK DIVIDER.		-1		1	%
Internal PLL frequency	f_{PLL}	f_{PLL} = 5 × fCLK16			80		MHz
Internal PLL Startup Time (Until Commit Goes Low)	$t_{PLLSTRT}$					170	µs
Internal 16MHz Clock for Internal Controller Logic and Interface Logic	f_{CLK16}	When internal oscillator is used as clock source.	CLK_SYS_SEL = 0	15.3	16	16.7	MHz
Internal 16MHz Clock for Internal Controller Logic and Interface Logic	f_{CLK16}	When external clock and PLL are used as clock source.	CLK_SYS_SEL = 1	15.84	16	16.16	MHz
SPI TIMINGS							
SCK Valid before or after Change of CSN	t_{CC}			TSCLK			ns
CSN High Time	t_{CSH}			4 × 1/fCLK16			ns
SCK Low Time	t_{CL}			20			ns
SCK High Time	t_{CH}			20			ns
SCK Frequency	f_{SCK}					8	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			34	50	ns
		V_{CC_IO} = 2.3V			55	85	
SDI, SCK, and CSN Filter Delay Time	t_{FILT}	Rising and falling edge			10		ns
					10		
STEP/DIR TIMINGS							

(V_S = 4.5V to 65V, R_{REF} = 12kΩ, typical values assume T_A = 25°C and V_S = 48V, limits are 100% tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Step Frequency	f_{STEP}	Maximum microstep resolution	Dedge = 1			$f_{CLK16}/8$	MHz
			Dedge = 0			$f_{CLK16}/4$	
Fullstep Frequency	f_{PS}					$f_{CLK16}/512$	MHz
STEP High Time	t_{SH}			100			ns
STEP Low Time	t_{SL}			100			ns
DIR to STEP Setup Time	t_{SU}			100			ns
DIR to STEP Hold Time	t_H			20			ns
ENCODER TIMING							
Encoder Counting Frequency	f_{CNT}				<10	16	MHz
A/B/N Input Low Time	t_{ABNL}			160			ns
A/B/N Input High Time	t_{ABNH}			160			ns
A/B/N Spike Filtering Time	$t_{FILTABN}$	Rising and falling edge			190		ns
ADC/SUPPLY/TEMPERATURE							
ADC Resolution					9		Bit
ADC_VSUPPLY Resolution	V_{ADCFS}				140.9		mV
Driver Temperature Accuracy	T_{driver}				±10		°C
Supply Voltage Measurement Accuracy		V_S ≥ 15V		-5.0		5.0	%
		V_S < 15V		-10		10	
ADC Sample Rate	$F_{SAMPLE,ADC}$				2.45		kHz
DAC OUTPUT							
DAC Resolution		V_{CC_IO} > 2.9V			8		Bit
Output Voltage Range		V_{CC_IO} > 2.9V		0.2		2.7	V
Output Current		V_{CC_IO} > 2.9V				5	mA
DAC Conversion Frequency		V_{CC_IO} > 2.9V				100	kHz

Package Information

FC2QFN30, 6mm × 6mm

Package Code	F306A6F+1F
Outline Number	21-100708
Land Pattern Number	90-100245
Thermal Resistance, Four-Layer Board, Ambient of 25°C, 4W Power Dissipation:	
Junction to Ambient (θ_{JA})	23 C/W
Junction to Case Thermal Resistance (θ_{JC})	2.6 C/W
Ψ_{JT}	1.9 C/W

Package Marking Specification

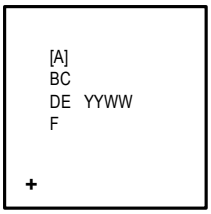


Figure 1. Package Marking (Top View)

Table 1. Packaging Marking Description

SYMBOL	DESCRIPTION
[A]	ADI Logo
BC	B – Part number prefix (3 digits, typically “TMC”) C – Part number (4 digits)
DE	D – Test Flow E – Part number suffix (2 digits)
YYWW	YY – manufacturing year WW – manufacturing work week
F	F – Assembly and Fab Code (7 digits)
+	Lead free indicator near pin 1

Pin Descriptions

PIN	NAME	FUNCTION	Type
3, 18	GND	Ground. Connect to ground plane.	GND
1, 5, 16, 20	V_S	Motor supply voltage. Provide filtering capacity near pin with shortest loop to GND plane/exposed pad.	Supply
26	V_{DD}	Output of internal 1.8V regulator. Attach 2.2µF or larger ceramic capacitor to AGND near to pin for best performance.	Supply
6	V_{CP}	Charge pump voltage. Tie to V_S using 1.0µF, 10V capacitor.	Output
7	CPO	Charge pump capacitor output.	Output
8	CPI	Charge pump capacitor input. Tie to CPO using 22nF, 100V capacitor.	Output

9	REFR/DIR	Direction Input in S/D Mode; Optional right reference input for internal ramp generator.	DI (pd)
10	REFL/STEP	Step Input in S/D Mode; Optional left reference input for internal ramp generator.	DI (pd)
11	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.	DI
12	CSN	SPI chip select input (negative active)	DI (pd)
13	SCK	SPI serial clock input	DI (pd)
14	SDI	SPI data input	DI (pd)
15	SDO	SPI data output (tristate)	DIO (pd)
27	I_{REF}	Analog reference current for current scaling. Provide external resistor to GND.	AI
23	ENCB	Encoder B-channel input	DI
24	ENCA	Encoder A-channel input	DI
28	ENCN/OVN	Encoder N-channel input/overvoltage output (negative active)	DI/DO
30	DRV_ENN	Enable input. The power stage is switched off (all motor outputs floating) when this pin is driven to a high level.	DI (pu)
17	OUTW	Motor coil W output	OUTPUT
2	OUTV	Motor coil V output	OUTPUT
4	OUTU	Motor coil U output	OUTPUT
21	SLEEPN	Low active power down input/reset input. Apply a continuous low level to bring the device to sleep mode. Once the IC returns from sleep mode/reset, it must be reconfigured before being used again. The latest register contents before going into sleep mode are not stored. While reconfiguring the IC, it is advised to still hold the bridge drivers disabled with DRV_ENN. Do not use during high motor velocity! If not used, connect to V_{CC_IO}. Internal pulldown	DI
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
22	DO0	Configurable output/diagnostics output 0. Interrupt or STEP/DIR output from internal motion controller for external driver. Use external pullup resistor in open-drain mode. In system reset state, this pin is actively pulled low to indicate reset condition to external controller. Configurable on-chip scope interface (OSCI) analog output for application tuning.	DO/AO
25	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.	Supply
29	DO1	Configurable output/diagnostics output 1. Interrupt or STEP/DIR output from internal motion controller for external driver. Use external pullup resistor in open-drain mode. Configurable on-chip scope interface (OSCI) analog output for application tuning.	DO/AO

Pin Configurations

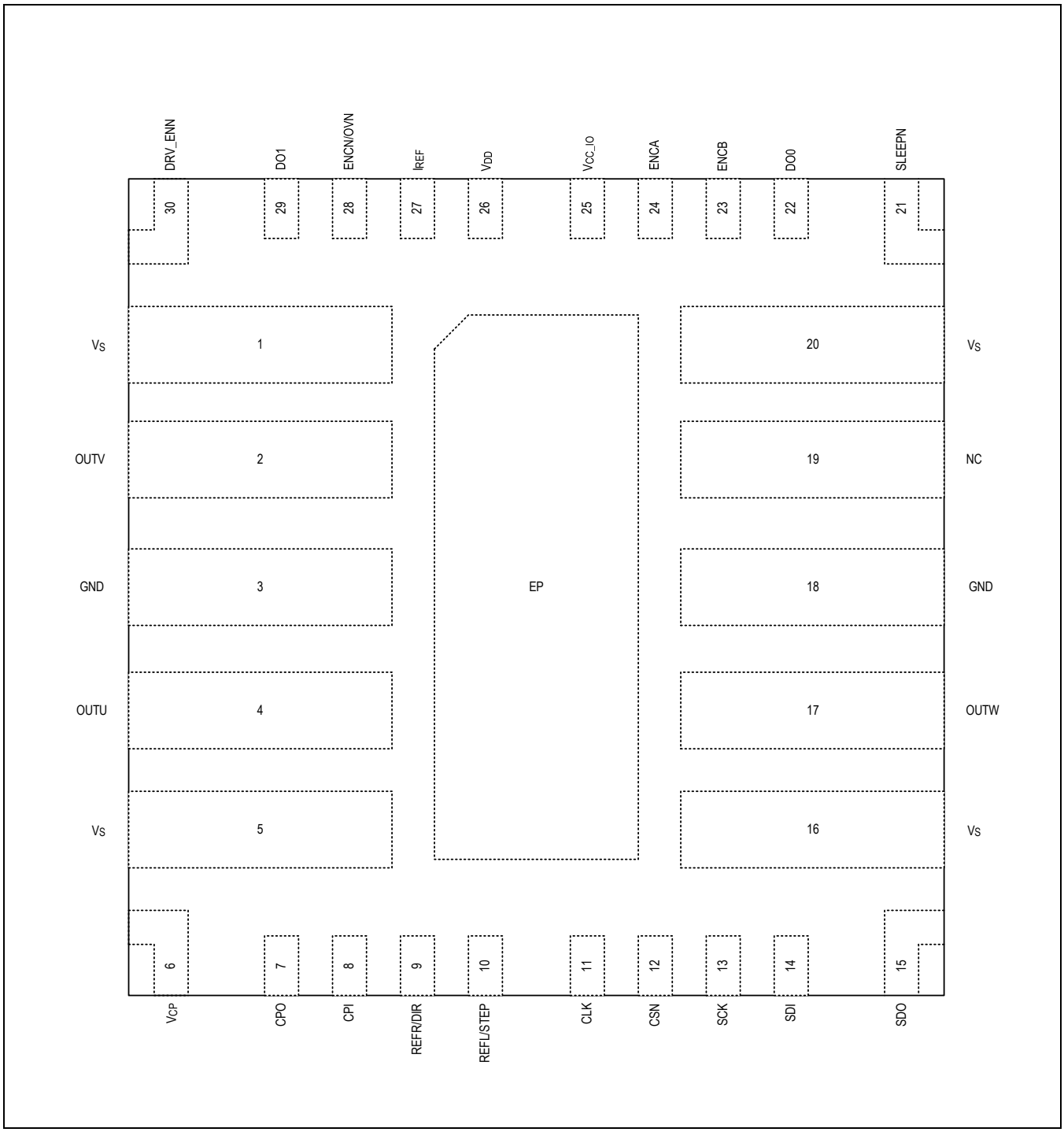


Figure 2. TMC3241 Pinout with Bump Side Down

Functional Diagrams

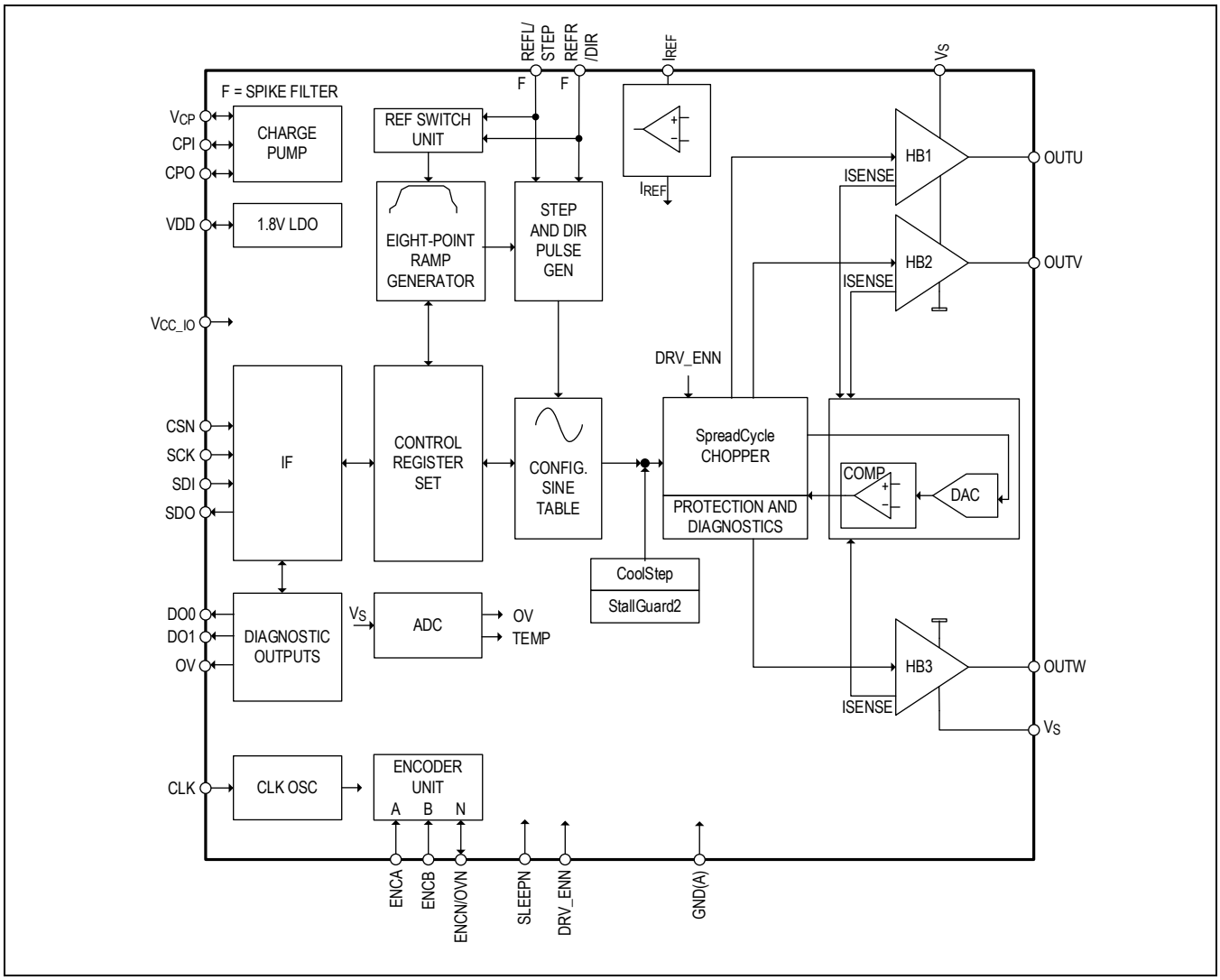


Figure 3. Block Diagram

Detailed Description

Principles of Operation

Full Featured Motion Controller and Driver

The TMC3241 is an intelligent stepper motor motion controller and driver chip – a cDriver. This smart power conversion component directly drives a three-phase stepper motor and interfaces to a CPU for configuration, control, and diagnostic feedback. All the functionality and logic to control and drive a stepper motor is integrated. No software is required to control the motor. Only configuration and target positions or velocities must be provided. The TMC3241 offers numerous unique functions, which are enabled by the system-on-chip integration of the driver and controller.

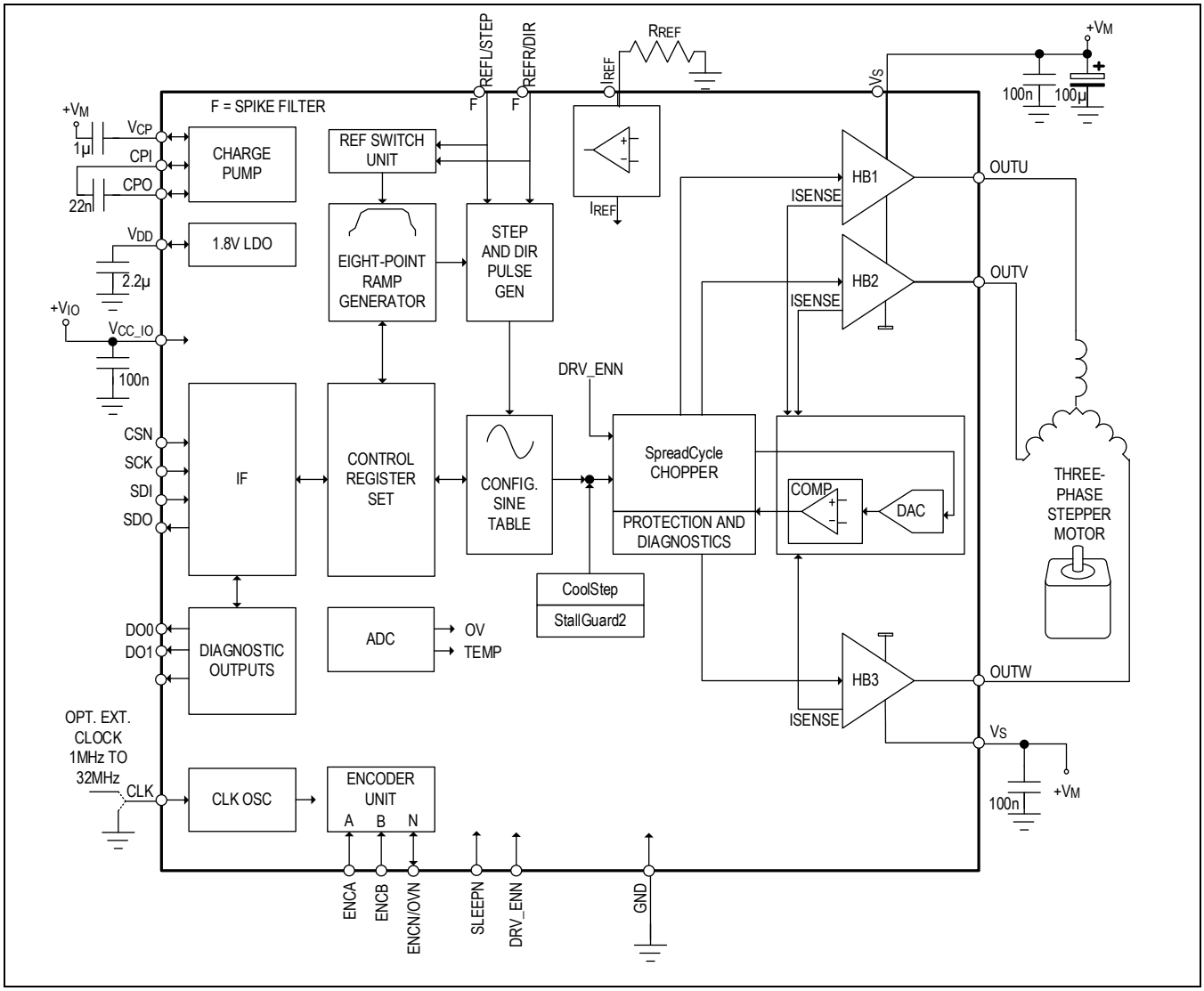


Figure 4. Block Diagram with Typical External Components

Key Concepts

The TMC3241 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.

Table 2. TMC3241 Feature Overview

FEATURE	DESCRIPTION
SpreadCycle	High-precision cycle-by-cycle current control for highest dynamic movements.
StallGuard2	Sensorless stall detection and mechanical load measurement for SpreadCycle.
CoolStep	Uses StallGuard2 to adapt the motor current for improved energy efficiency and lower heating of motor and driver.

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

Control Interface

The TMC3241 supports a serial peripheral interface (SPI). The SPI is a bit-serial interface synchronous to a bus clock. For every bit sent from the bus controller to the bus peripheral, another bit is sent simultaneously from the peripheral back to the controller. Communication between an SPI controller (example, an MCU) and the peripheral always consists of sending one 40-bit command word and receiving one 40-bit status word.

Step and Direction Interface

The step and direction interface is used for controlling the motor movement (position, speed, acceleration, and direction) from an external motion controller. Active edges on the STEP input can be rising edges or both rising and falling edges, as controlled by the 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 microstep or a number of 2, 4, 8, 16, …256 microsteps depending on the configuration. A step impulse with a low state on DIR increases the microstep counter and a high state on DIR decreases the counter by an amount controlled by the microstep resolution. An internal table translates the counter value into the target current values to control the motor current.

The sine wave table uses 1024 entries for an electrical field rotation. Therefore, each six motor fullsteps are mapped to 1024 microsteps. This means, a native motor fullstep is divided into 1024/6 = 170.67 microsteps. Thus, a 300 fullstep three-phase stepper offers 51200 microsteps per rotation, or an eight pole-pair permanent magnet synchronous motor (PMSM) offers 8192 fullsteps per rotation.

Automatic Standstill Power Down

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

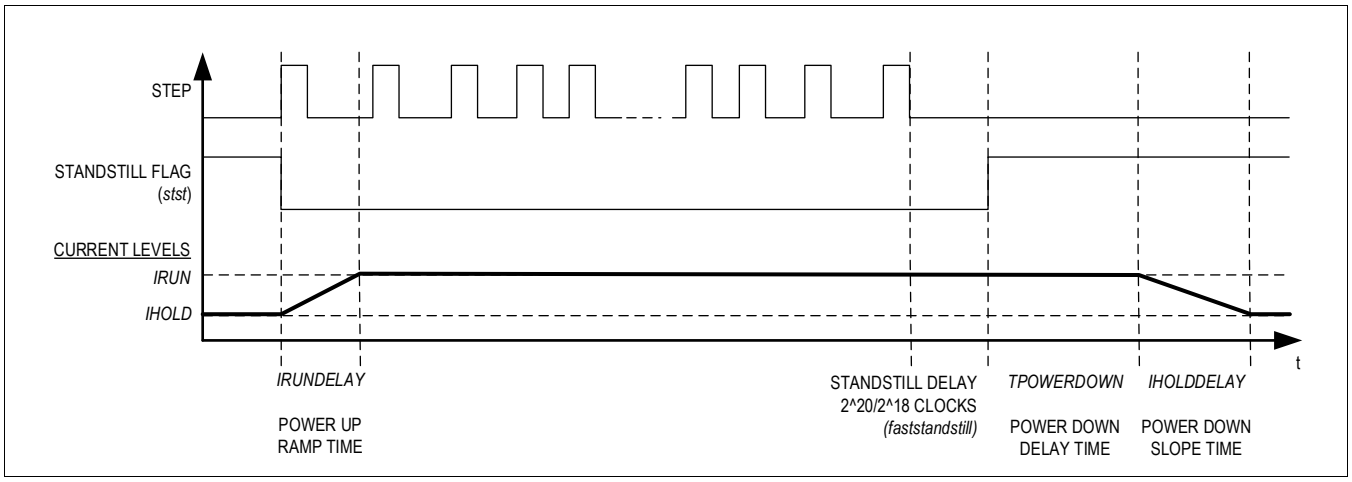


Figure 5. Automatic Motor Current Control at Standstill and Ramp-Up

Mechanical Load Sensing with StallGuard2

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 that 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.

Load Adaptive Current Control with CoolStep

CoolStep drives the motor using an optimized current level. It is based on the StallGuard2 load measurement information to adjust the motor current to the minimum amount required in the actual load situation.

CoolStep allows saving a lot of energy, especially for motors that see varying loads and operate at a high duty cycle. Taking into account that a stepper motor application must work with a torque reserve of 30% to 50%, even a constant load application allows saving lots of energy, because the driver automatically enables torque reserve, when required. The reduction in power dissipation further keeps the system cooler and increases life time, and allows savings in the power supply and cooling infrastructure. Keep in mind that half motor current means a quarter of the power dissipation in the motor coils. This power dissipation makes up for most of the stepper motor losses!

Real Time On-Chip Scope Interface

One novel function of the TMC3241 is its real time on-chip scope interface (RT-OSCI). Two internal DACs can be mapped to the diagnostic pins, and these output a range of internal registers for tuning support, debugging, and monitoring.

Encoder Interface

The TMC3241 provides an encoder interface for external incremental encoders. The encoder can be used for the homing of the motion controller (alternatively to reference switches). A programmable prescaler allows the adaptation of the encoder resolution to the motor resolution. A 32-bit encoder counter is provided.

Serial Peripheral Interface (SPI)

SPI Datagram Structure

The TMC3241 uses 40-bit SPI datagrams for communication with a microcontroller. Microcontrollers equipped with hardware SPI are typically able to communicate using integer multiples of 8 bits. The CSN line must stay active (= low) for the complete duration of the datagram transmission.

Each datagram 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 with 32 data bits even if it uses less than 32 data bits. For simplification, each register is specified by a one byte address:

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

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

Table 3. 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)

Read and write are 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. The bit named W is a WRITE_notREAD control bit. 0x80 must be added to the address for a write access. The SPI always delivers data back to the controller, independent of the W bit. The data transferred back is the data read from the address, which is transmitted with the previous datagram if the previous access is a read access. If the previous access is a write access, the data read back mirrors the previously received write data. 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. The following read or write access delivers back the data read from the address transmitted in the preceding read cycle.

A read access request datagram uses dummy write data. Read data is transferred back to the controller with the subsequent read or write access. Hence, multiple registers can be read in a pipelined fashion. Whenever data is read from or written to the TMC3241, 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 must be set to 0x21 in the access preceding the read access. For a write access to the register (VMAX), the address byte must be set to 0x80 + 0x27 = 0xA7. For read access, the data bit might have any value (-). So, set these to 0.

Table 4. SPI Read/Write Example Flow

ACTION	DATA SENT TO *{Variable not found in the variables list}*	DATA RECEIVED FROM *{Variable not found in the variables list}*
Read XACTUAL	0x2100000000	0xSS and unused data*
Read XACTUAL	0x2100000000	0xSS and XACTUAL
Write VMAX = 0x00ABCDEF	0xA700ABCDEF	0xSS and XACTUAL
Write VMAX = 0x00123456	0xA700123456	0xSS00ABCDEF

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

SPI Status Bits Transferred with Each Datagram Read Back

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

Table 5. SPI_STATUS – Status Flags Transmitted with SPI Access in Bits 39 to 32

BIT	NAME	COMMENT
7	*status_stop_r*	RAMP_STAT[1] – 1: Signals stop right switch status (motion controller only).
6	*status_stop_l*	RAMP_STAT[0] – 1: Signals stop left switch status (motion controller only).
5	*position_reached*	RAMP_STAT[9] – 1: Signals target position reached (motion controller only).
4	*velocity_reached*	RAMP_STAT[8] – 1: Signals target velocity reached (motion controller only).

3	*standstill*	DRV_STATUS[31] – 1: Signals motor standstill.
2	*sg*	DRV_STATUS[24] – 1: Signals StallGuard flag active.
1	*driver_error*	GSTAT[1] – 1: Signals driver error (clear through register *GSTAT*).
0	*reset_flag*	GSTAT[0] – 1: Signals a reset occurred (clear through register *GSTAT*).

Data Alignment

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

SPI Signals

The SPI bus on the TMC3241 has four signals:

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

The SPI peripheral is enabled for an SPI transaction by a low on-the-chip select input CSN. Bit transfer is synchronous to the bus clock SCK, with the peripheral latching the data from SDI on the rising edge of SCK and driving data to SDO following the falling edge. The most significant bit is sent first. A minimum of 40 SCK clock cycles is required for a bus transaction with the TMC3241.

If more than 40 clocks are driven, the additional bits shifted into SDI are shifted out on SDO after a 40-clock delay through an internal shift register. This can be used for daisy chaining multiple chips.

The CSN must be low during the whole bus transaction. When CSN goes high, the contents of the internal shift register are latched into the internal control register and recognized as a command from the SPI controller to the SPI peripheral. If more than 40 bits are sent, only the last 40 bits received before the rising edge of CSN are recognized as the command. The driver only accepts SPI transfers of a minimum of 40 bits and longer by a multiple of 8 bits.

SPI Timing

The SPI max frequency is at 8MHz. 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. *Figure 6* shows the timing parameters of an SPI bus transaction. The **Electrical Characteristics** table shows the timing values.

The SPI uses SPI MODE 3. Using any other mode does not result in reliable data transfer.

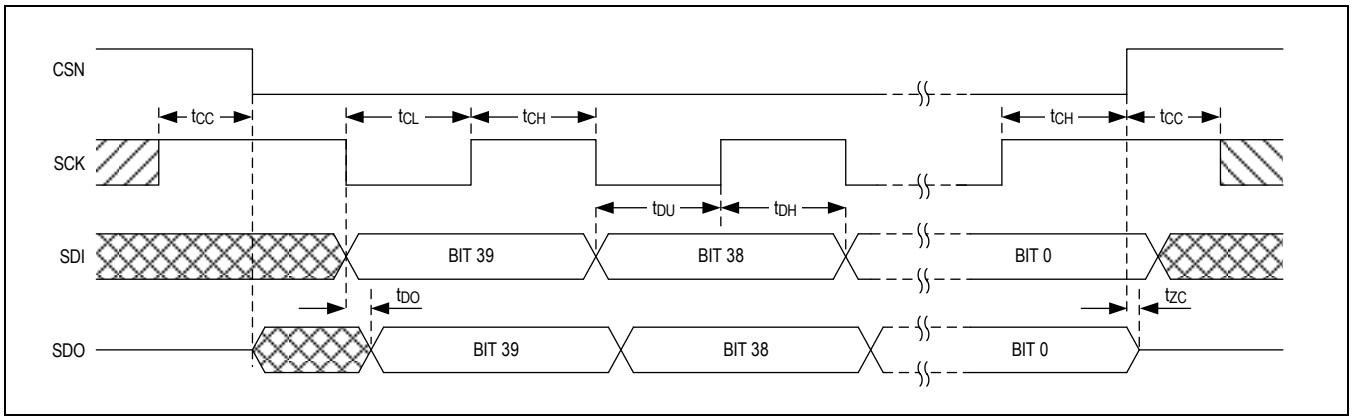


Figure 6. SPI Timing Diagram

Step and 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

Figure 7 shows the timing parameters for the STEP and DIR signals. When the *dedge* mode bit in the *CHOPCONF* register is set, both edges of the STEP input signal 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 approximately 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.

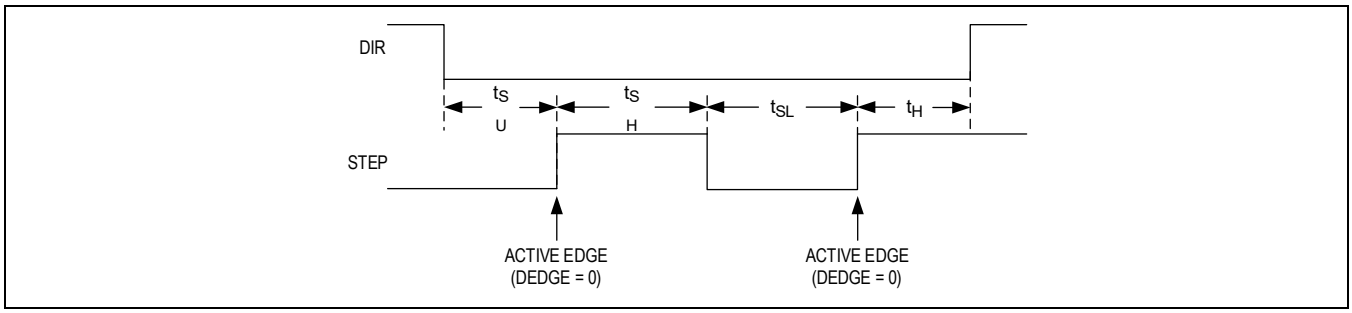


Figure 7. STEP and DIR Signal Timing

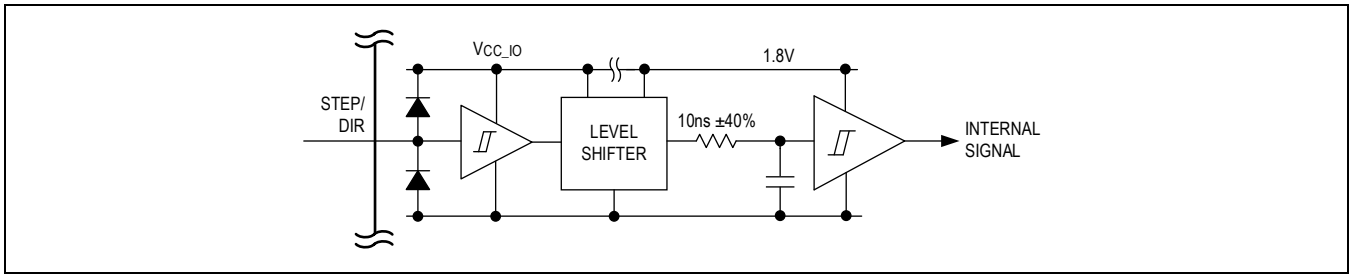


Figure 8. STEP and DIR Signal Input Filter Structure

Changing Resolution

A reduced microstep resolution allows limiting the step frequency for the STEP and 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 six 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, and 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 per STEP pulse. The sequencer has a special provision to allow seamless switching among 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 lower resolutions, because any failure in the step sequence leads to an asymmetrical run when comparing a motor running clockwise and counterclockwise. MicroPlyer smoothens the motion by taking all the intermediate table steps, once a certain minimum velocity is reached. Therefore, the more coarse discrete positions with reduced resolution only determine possible motor positions, once the motor is in standstill.

Examples:
Quarter Resolution: Cycles through the microstep table taking four steps per step pulse, thus, effectively reaching four times the motor velocity for the same frequency step generation. The step counter goes to positions, -6, -2, 2, 6, 10, … in the table, with positions -2 and 2 being symmetrical to the zero transition. Effectively, this results in 64 steps per quarter sine wave, 256 steps per sine wave, or 256/6 = 42.67 microsteps per motor fullstep.

MicroPlyer Step Interpolator and Standstill Detection

For each active edge on STEP, MicroPlyer produces microsteps at full 256x table resolution (171 effective microsteps between each two fullsteps of the three-phase motor). It interpolates the time between the two-step impulses at the step input based on the last step interval. This way, from two microsteps up to 256 microsteps are driven for a single-step pulse. Take into account that the native resolution of the three-phase motor is six fullsteps per 1024 microsteps, and therefore, the effective microstep resolution per fullstep of the motor is not an integer number.

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

The step rate for the interpolated two 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 two microsteps corresponds to 2^{20} (roughly one million system clock cycles) for an even distribution of 256 microsteps. At a 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 set, which indicates a standstill event. At this 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 the holding current IHOLD.

Note: 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!

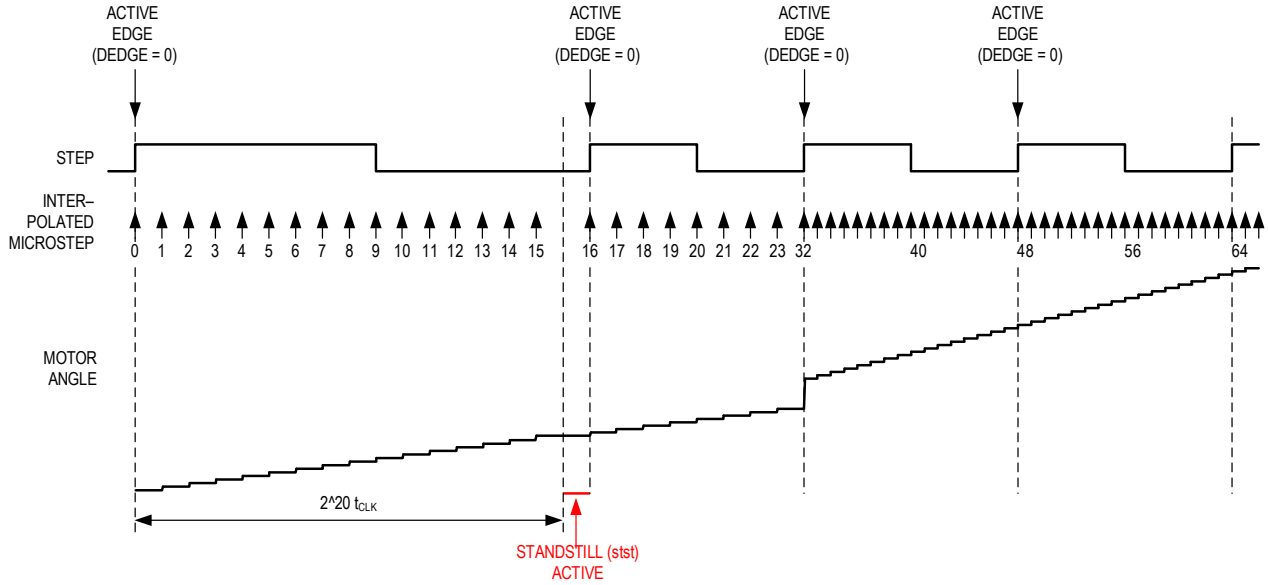


Figure 9. MicroPlyer Microstep Interpolation with Rising STEP Frequency (Example: 16 to 256)

In *Figure 9*, 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 does not generate all 16 microsteps. So, there is a small jump in the motor angle between the first and second cycles at the higher rate.

SpreadCycle

SpreadCycle is a cycle-by-cycle current control. Therefore, it can react extremely fast to changes in the motor velocity or motor load. The chopper regulates the current in the three coils by switching each coil in one of three different states. *Figure 10* shows the different phases of a chopper cycle for one coil, which is seen by two half-bridges. This assumes a triangle connection of the coils while a star connection virtually shows the same behavior. In the on-phase, the current is actively driven into the coils by connecting them to the power supply in the direction of the target current. A fast decay phase reverses the polarity of the coil voltage to actively reduce the current. The slow decay phase shorts the coil to let the current recirculate.

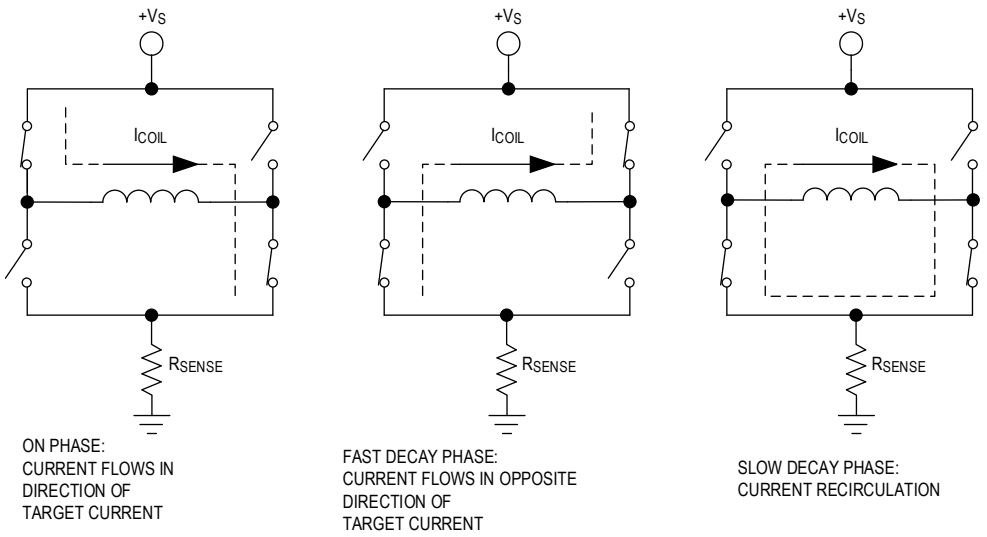


Figure 10. Typical Chopper Decay Phases (Sense Resistor is Only Symbolic for Current Measurement in Low-Side Path)

Although the current can be regulated using only the on-phases and fast decay phases, inserting 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 the coil current during phases when the current flows through exactly one low-side 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 the parasitic capacitance. During this time, typically one or two microseconds, the current cannot be measured and must be masked. This is accomplished by a blanking time to block these spikes.

The chopper frequency is an important parameter for a chopped motor driver. A too low frequency might generate audible noise. A higher frequency reduces the 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.

Table 6. Parameters Controlling SpreadCycle

PARAMETER	DESCRIPTION	SETTING	COMMENT
TOFF	Sets the slow decay time (off time). This setting also limits the maximum chopper frequency. For most applications, an off-time is not required. Here, a dummy value other than 0 must be programmed to this register to enable the driver and the *nosd* flag is set. Setting this parameter to zero completely disables all the driver transistors and the motor can freewheel. Default = 0	0	Chopper off
		1…15	Off-time setting $N_{CLK} = 24 + 32 \times TOFF$ (1 works with a minimum blank time of 28 clocks).
TBL	Selects the comparator blank time. This time must 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, for example, when filter networks are used, a setting of 2 or 3 is required. Default = 2	0	20 t_{CLK}
		1	28 t_{CLK}
		2	40 t_{CLK}
		3	58 t_{CLK}
HYST	The hysteresis setting is the main control for the chopper and determines the chopper frequency. A higher setting introduces more current ripple, and thus, reduces frequency. A too low setting results	0 … 63	Hysteresis for the chopper.

	in the coil current only loosely following the target current, and thus, reduced microstep performance, especially in the current zero crossing. A too high setting can cause audible chopper noise.		
nosd	Selection of the *TOFF* insertion.	0	Use *TOFF* setting for additional SD phases.
		1	No slow decay phase
CSYNC	This bit switches on the chopper synchronization. If enabled, the chopper engine is reset with each motor fullstep, to avoid a beat occurring between the full step sequence and chopper clock.	0	Disable
		1	Sync. enable

SpreadCycle Chopper

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

Each chopper cycle comprises an on-phase, a fast decay phase, and a slow decay phase. Additional slow decay phases can be removed by setting *CHOPCONF.nosd.* The hysteresis determines the chopper frequency by forcing the driver to introduce some amount of current ripple into the motor coils. The motor inductivity determines the ability to follow a changing motor current. The duration of the on-phase and fast decay phase must cover at least the blank time, because the current comparator is disabled during this time. This is satisfied by choosing a positive value for the hysteresis and can be estimated by the following calculation:

$$f_{CHOP} = \frac{V_M}{2 \times \frac{2}{3} L_{COIL} \times I_{HYST}}$$

where, f_{CHOP} is the resulting chopper frequency. I_{COIL} is the peak motor coil current at the maximum motor current setting CS, and motor coil resistance R_{COIL} motor coil inductivity L_{COIL}.

The current hysteresis I_{HYST} results from the *CHOPCONF.HYST* setting as follows:

$$I_{HYST} = CHOPCONF.HYST \times \frac{I_{COIL}}{2 \times 248} \times \frac{256}{CS + 1}$$

The calculated chopper frequency should preferably lie between 18kHz and 60kHz. If a too high chopper frequency results, increase hysteresis, or add a slow decay phase with CHOPCONF.TOFF duration.

The setting can also be determined by experimenting with the motor: a too low hysteresis setting results in reduced microstep accuracy, while a too high setting leads to more chopper noise and motor power dissipation. The correct setting can be determined best by rotating the motor slowly, and increasing hysteresis setting, until the motion of the motor is very smooth (feel with fingers or add a long pointer to the axis, example, laser pointer). Or, measure the motor currents with a current probe or an oscilloscope, and check the waves for a pure sine wave. A further increment of the hysteresis setting lowers the chopper frequency and might at some point generate audible chopper noise. For high inductivity motors, audible noise might occur at an optimum setting. Increase the supply voltage, or choose a motor with a different, higher current winding. (For further reading, refer to the ADI-Trinamic application note *AN001*).

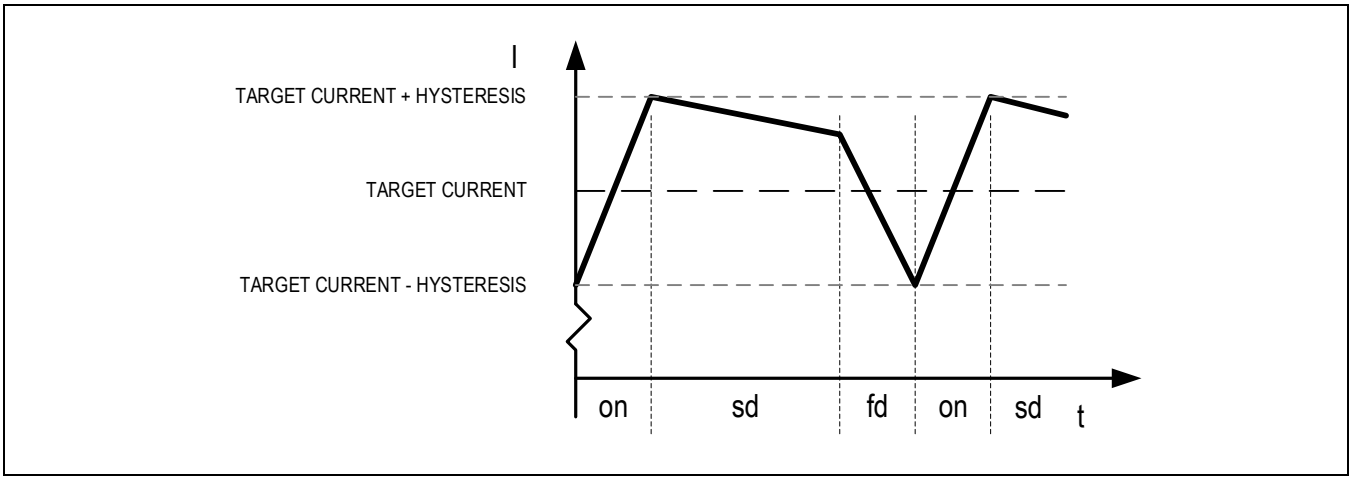


Figure 11. SpreadCycle Chopper Scheme Showing the Coil Current During a Chopper Cycle

Direct Mode

The TMC3241 offers a special direct motor-phase current control mode called the direct mode. This mode allows to individually command the dedicated phase current values per motor phase using SPI.

Setting the bit *GCONF.direct_mode* switches into the direct mode.

Register XTARGET (0x2D) specifies the signed coil U current I_U (bits 24...16) and coil V current I_V (bits 8...0). Coil W current results from the following equation:

$$I_W = -(I_V + I_U)$$

In this mode, the current is scaled by the I_{HOLD} setting.

StallGuard2 Load Measurement

To complement the SpreadCycle operation with a sensorless feedback system, the TMC3241 offers a special three-phase variant of StallGuard2. StallGuard2 works in the SpreadCycle mode.

StallGuard2 provides a measurement of the load reserve of the motor. It can be used for stall detection as well as other uses at loads below that 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. Its absolute value (offset) depends on the operating conditions and settings. As the load on the motor increases, the StallGuard value (*SG_RESULT*) decreases. Tuning is required to properly detect stalls. Set the StallGuard threshold (*SGTHRS*) such that *SG_RESULT* reaches 0 (or near to 0) when the motor is overloaded/stalls.

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

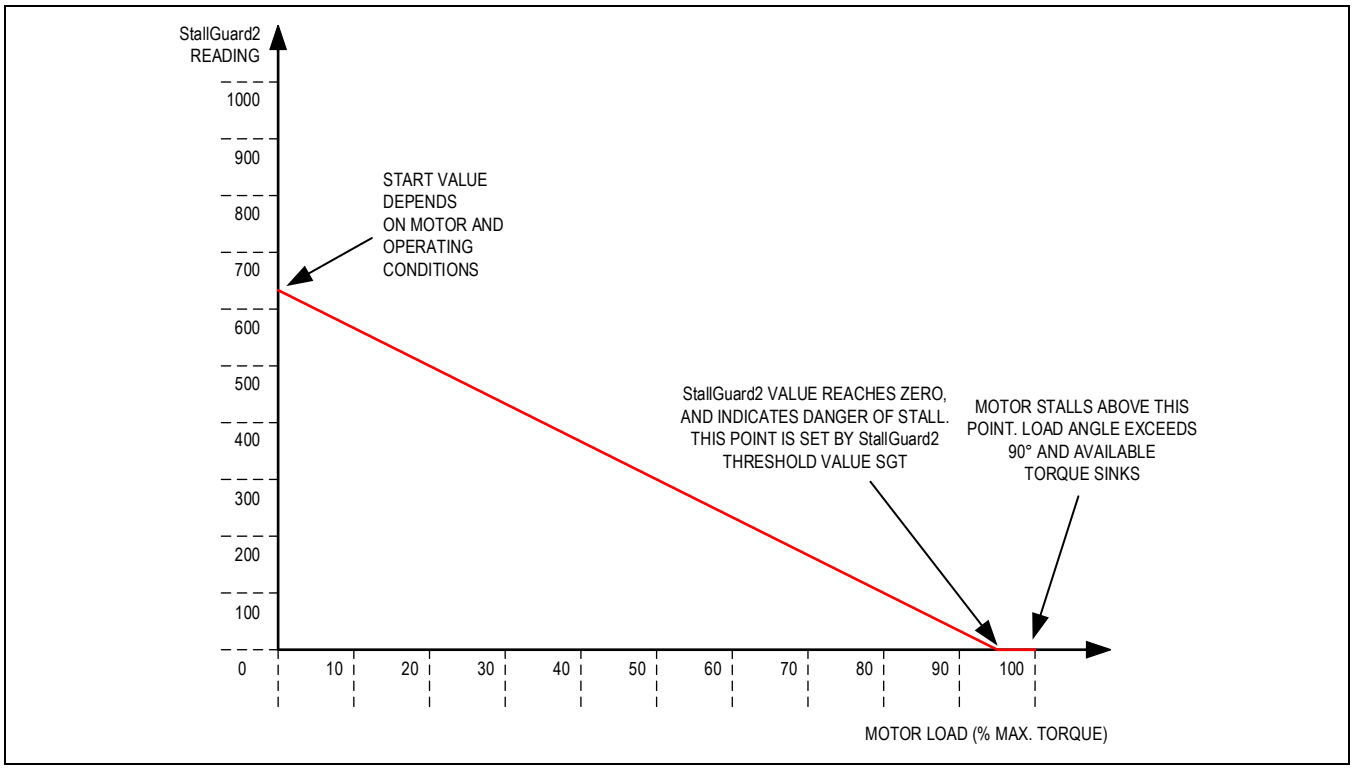


Figure 12. Function Principle of StallGuard2

Table 7. 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 (six fullsteps).	0	Standard mode (best for stall detection with hard stall).
		1	Filtered mode
sg_speed	Selects a longer filtering time for less noise at low motor velocities.	0	Standard mode
		1	Low velocity mode
STATUS WORD	**DESCRIPTION**	**RANGE**	**COMMENT**
SG_RESULT	This is the StallGuard2 result. A higher reading indicates less mechanical load. A lower reading indicates a higher load, and thus, a higher load angle. Tune the *SGT* setting to show a *SG_RESULT* reading of roughly 0 to 100 at the maximum load before motor stall.	0…1023	0: Highest load Low value: High load High value: Less load

Tuning StallGuard2 Threshold SGT

The StallGuard2 value *SG_RESULT* is affected by motor-specific characteristics and application-specific demands on load, velocity, supply voltage, and current level. Therefore, the easiest way to tune the StallGuard2 threshold *SGT* for a specific motor type and operating conditions is interactive tuning in the actual application.

Table 8. Procedure to Tune StallGuard

NUMBER	TASK	DETAIL
1	Operate Motor	Operate the motor at the normal operation velocity, supply voltage, and current setting for the application, and monitor *SG_RESULT*. Ideally, the velocity and current setting for the homing should be separately tuned, if different from normal operation, and selected, as desired, for the homing of the drive.
2	Apply Load and Optimize *SGT*	Apply slowly increasing mechanical load to the motor. Now, try the best setting for *sg_speed* and *sfilt* to have low noise on the signal, but high amplitude of *SG_RESULT* over the applicable load range. If the motor stalls before *SG_RESULT* reaches zero, decrease *SGT*. If *SG_RESULT* reaches zero before the motor stalls, increase *SGT*. A good *SGT* starting value is zero. *SGT* is signed. So, it can have negative or positive values.
3	Enable Stop-on-Stall	Now, enable *sg_stop* and make sure the motor is stopped safely whenever it is stalled. Disable *sfilt* for a quick response to the overload. Increase *SGT* if the motor is stopped before a real stall occurs. Restart the motor by disabling *sg_stop* or by clearing *event_stop_sg* in the *RAMP_STAT* register (write to clear). StallGuard sensitivity in the homing process can be controlled by both the motor current and *SGT* setting.
4	Test and Optimize Settings	The optimum setting is reached when *SG_RESULT* is between 0 and roughly 100 at increasing load shortly before the motor stalls, and *SG_RESULT* increases by 100 or more without load. *SGT*, in most cases, can be tuned for a certain motion velocity or a velocity range. Make sure the setting works reliably in a certain range (example, 80% to 120% of the desired velocity) and also under extreme motor conditions (lowest and highest applicable motor temperature when in the homing condition).

The upper velocity for the stall detection with this setting is determined by the velocity, where the motor back-EMF approaches the supply voltage and the motor current starts dropping when further increasing the velocity. Above this velocity, StallGuard2 delivers less signal.

The power supply voltage affects *SG_RESULT*. So, tighter regulation results in more accurate values. *SG_RESULT* measurement has a high resolution, and there are a few ways to enhance its accuracy, as described in the following sections.

Variable Velocity Limits *TCOOLTHRS* and *THIGH*

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

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

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

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

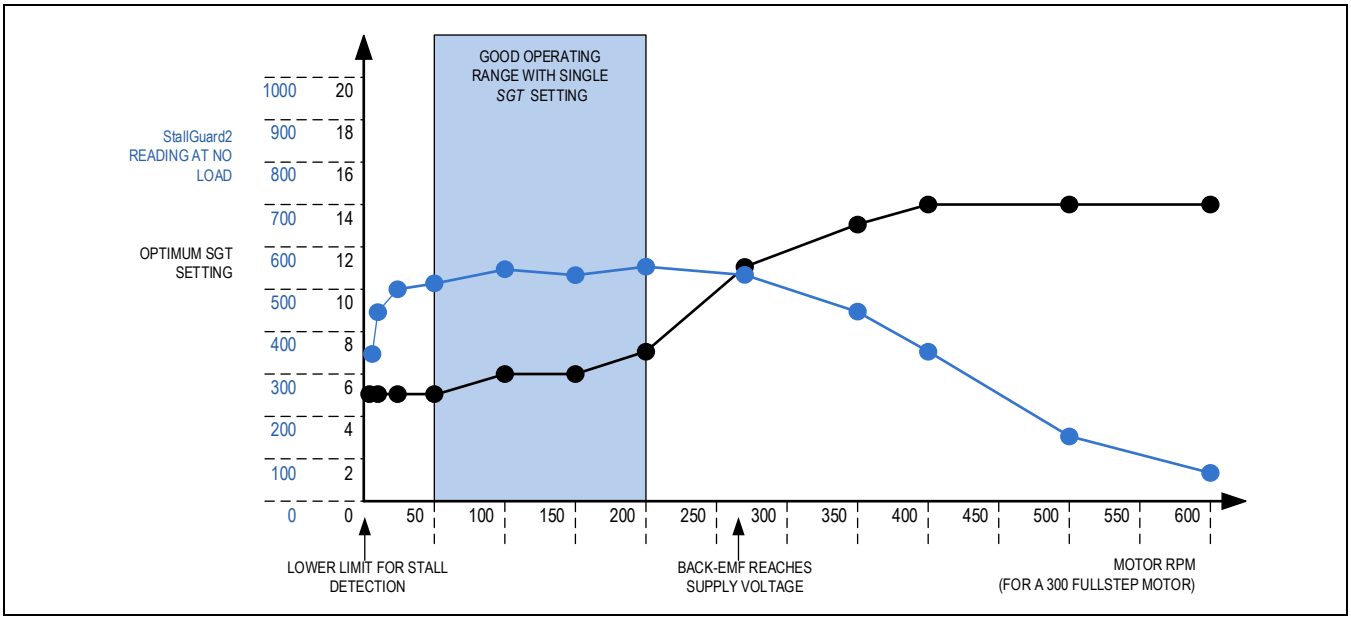


Figure 13. Example: Optimum SGT Setting and StallGuard2 Reading with an Example Motor

Small Motors with High Torque Ripple and Resonance

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

Temperature Dependence of Motor Coil Resistance

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

Accuracy and Reproducibility of StallGuard2 Measurement

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

$$\text{StallGuard2 measurement error} = +/- \max(1, |SGT|)$$

StallGuard2 Update Rate and Filter

The StallGuard2 value is updated with each two fullsteps of the motor. This is enough to safely detect a stall, as stalling of the motor always means the loss of six fullsteps. In a practical application, especially when using CoolStep, a more precise measurement might be more important than an update for each two fullsteps, taking into account that mechanical load never changes instantaneously from one step to the next. Therefore, a filtering function is available: the SFILT bit enables the filtering of the motor load measurement over three measurements. The filter should always be enabled when a precise measurement is desired. It compensates for anisotropies in the construction of the motor, example, due to the misalignment of the magnet poles. The bit should be cleared only if a very fast response to the increasing load is required.

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 must be determined, which the motor can drive without stalling. At the same time, monitor the *SG_RESULT* value at this load, for example, some value within the range 0 to 100. The stall threshold should be a value safely within the operating limits to allow for parameter stray. The response at an *SGT* setting at or near 0 gives some idea on the quality of the signal: check the *SG_RESULT* value without load and with maximum load. These should show a difference of at least 100 or a few 100, which is large compared to the offset. When setting *SGT* in a way that a reading of 0 occurs at maximum motor load, a motor stall can be automatically detected to issue a motor stop. In the moment of the step resulting in a step loss, the lowest reading is visible. After the step loss, the motor vibrates and shows a higher *SG_RESULT* reading.

Homing with StallGuard2

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

Limits of the StallGuard2 Operation

StallGuard2 does not operate reliably at extreme motor velocities: very low motor velocities (for many motors, less than 1 RPS) generate a low back-EMF and make the measurement instable and dependent on environment conditions (temperature, and more). Other conditions also lead to the extreme settings of *SGT* and poor response of the measurement value *SG_RESULT* to the motor load.

Operation at 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.

CoolStep Load Adaptive Current Scaling

CoolStep is an automatic smart energy optimization for stepper motors based on the motor mechanical load. In applications with varying load, it can dramatically increase motor efficiency. CoolStep requires StallGuard2 to be tuned before use. Like StallGuard2, CoolStep is limited to operation at a narrow target velocity range, as a single tuning covers a single velocity operating point.

Setting Up for CoolStep

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

Table 9. CoolStep Critical Parameters

PARAMETER	DESCRIPTION	RANGE	COMMENT
SEMIN	4-bit unsigned integer that sets a lower threshold. If *SG_RESULT* goes below this threshold (indicating a large load), CoolStep increases the current to the 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* × 32. 1 is a good starting value for operation.
SEMAX	4-bit unsigned integer that controls an upper threshold. If *SG_RESULT* is sampled equal to or above this threshold enough times (indicating a light load), CoolStep decreases the current to the coils. The upper threshold is (*SEMIN* + *SEMAX* + 1) × 32.	0…15	Threshold is (*SEMIN* + *SEMAX* + 1) × 32. Applications often work best with a setting of 0 or 1 (small hysteresis).

Figure 14 shows the operating regions of CoolStep:

- The black line represents the *SG_RESULT* StallGuard measurement.
- The blue line represents the mechanical load applied to the motor.
- The red line represents the current scaling *CS_ACTUAL*, as determined by CoolStep.

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

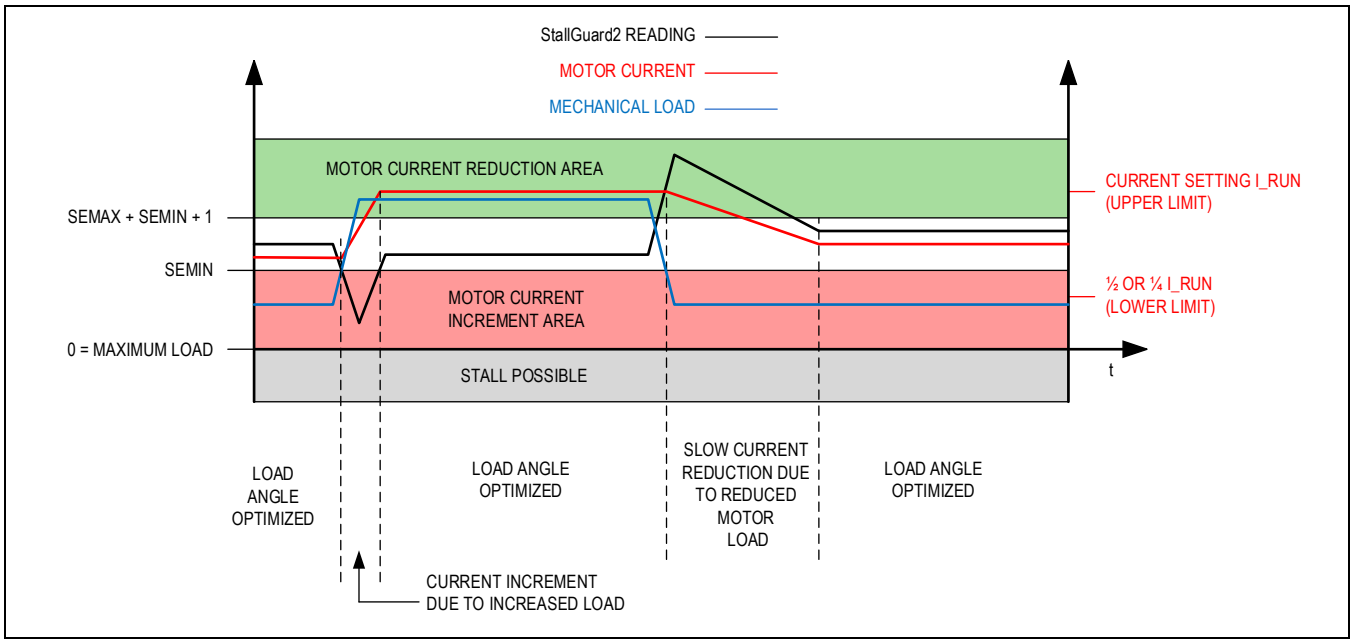


Figure 14. CoolStep Adapts Motor Current to the Load

Table 10. CoolStep Additional Parameters and Status Information

PARAMETER	DESCRIPTION	RANGE	COMMENT
SEUP	Sets the current increment step. The motor current is incremented by this setting whenever a new StallGuard2 value is measured that lies below the lower threshold as set by *SEMIN*.	0…3	Step up width of *CS_ACTUAL* is 8, 16, 32, 64.
SEDN	Controls the current reduction slope for StallGuard2 readings above the upper threshold.	0…7	Reduction of *CS_ACTUAL* per StallGuard2 reading: 1/8, ¼, ½, 1, 2, 4, 8, 16.
SEIMIN	Sets the lower motor current limit for CoolStep operation by scaling down the *IRUN* current setting to ½ or ¼.	0	0: 1/2 of *IRUN* (standard setting).
		1	1: 1/4 of *IRUN* (usable in limited cases only).
TCOOLTHRS	Lower velocity threshold for switching on CoolStep. Below this velocity, CoolStep is disabled. Adapt to the lower limit of the velocity range where StallGuard2 gives a stable result. **Hint:** May be adapted to disable CoolStep during the acceleration and deceleration phases by setting the threshold velocity just slightly below *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 is disabled. Adapt to the velocity range where StallGuard2, respectively, CoolStep gives a good 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…255	Check to monitor CoolStep activity.

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°.

The current increment speed is specified in *SEUP* and the current decrement speed is specified in *SEDN*. These can be tuned separately because these 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 three. **Advice:** the most common and most beneficial use is to adapt CoolStep for operation at the typical system target operation velocity and to set the velocity thresholds accordingly. As acceleration and decelerations normally shall be quick, these require the full motor current, while these have only a small contribution to the overall power consumption due to their short duration.

Low Velocity and Standby Operation

CoolStep is not able to measure the motor load in standstill and at very low RPM. Therefore, a lower velocity threshold is provided by *TCOOLTHRS*. It should be set to an application-specific default value. Below this threshold, the normal current setting using *IRUN*, respectively, *IHOLD* is valid. An upper threshold is available through the *VHIGH* setting. The velocity limits *VHIGH* and *VCOOLTHRS* are determined by the settings *THIGH* and *TCOOLTHRS*.

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

Integrated Current Sense

Non-dissipative current sensing is integrated in the TMC3241 (ICS). This feature eliminates the bulky external power resistors, which are normally required with external current sensing. The ICS results in a dramatic space and power saving compared with mainstream applications based on the external sense resistor. For optimum performance, the ICS individually measures $R_{DS(ON)}$ for each of the power MOSFETs, taking into account individual MOSFET temperature to yield the best results.

Setting the Full-Scale Current Range

The maximum full-scale current I_{FS} is defined by the external reference resistor R_{REF}. A standard low-power resistor with 1% accuracy is sufficient. Typically, R_{REF} has a value of 12kΩ. A missing resistor is signaled through the flag *IOIN.ext_ref_det* remaining at 0.

A set of registers allows the scaling of the full-scale current for a specific application and motor. A first adaptation to the desired motor current range is required using the global settings *CURRENT_RANGE* and *CURRENT_RANGE_SCALE*. Select these settings to cover the maximum current desired for the motor operation. Typically, this is the nominal motor current specified in the motor data sheet, or up to 150% of it, to allow short time torque increase. Within the selected range, fine adjust the motor current to fit the individual motor, application, and its operating situation using *IRUN* and *IHOLD*. The automatic load-dependent current adaptation by CoolStep uses *IRUN* as maximum value and automatically scales down to a fraction of it to match the actual mechanical load. Take care that the motor current also scales with the reference current. Therefore, the reference resistor should be selected for a tight matching.

I_{FS} is calculated using the following equation:

$$I_{FS} = CRS \times \frac{KIFS}{R_{REF}}$$

Table 11 shows the available maximum full-scale ranges with a typical R_{REF} of 12kΩ. The given currents include the upper limit of the chopper hysteresis. It is not recommended to work with different values for R_{REF}.

Table 11. Full-Scale Current Register Configuration for R_{REF} = 12kΩ

DRV_CONF REGISTER BITS 1...0 (CURRENT_RANGE)	KIFS [A x kΩ]	DRV_CONF REGISTER BITS 3…2 (CURRENT_RANGE_ SCALE)	CURRENT RANGE SCALE (CRS)	MAX. FULL-SCALE SETTING I_{FS}		TYPICAL $R_{DS(ON)}$ (LS) (Ω)
				[A] PEAK	[A] RMS	
11	72	11	100%	6.000	4.24	0.036
10	54	11	100%	4.500	3.18	0.047
01	36	11	100%	3.000	2.12	0.070
00 (default)	18	11	100%	1.500	1.06	0.138
		10	75%	1.125	0.795	
		01	50%	0.750	0.530	
		00 (default)	25%	0.375	0.265	

A precise calculation for SpreadCycle operation is given in the following equation. It considers the sine wave amplitude.

$$RMSCurrent_{Spread} = 4.24A \times \frac{248}{256} \times \frac{CURRENT_RANGE + 1}{4} \times \frac{CURRENT_RANGE_SCALE + 1}{4} \times \frac{IRUN + 1}{250}$$

where, 248/256 is the amplitude of the sine wave table.

The parameters given in *Table 12* allow to adapt the current scaling, as well as the current ramp up and down.

Table 12. Parameters Controlling the Motor Current

PARAMETER	DESCRIPTION	SETTING	COMMENT
CURRENT_RANGE	Choose before the operation of the motor as fitting to the motor's maximum current and keep this value fixed within the application. The peak current setting this way is adapted by changing the on-resistance of the low-side MOSFETs to a lower value for higher motor currents.	0…3	Selects the current range of the driver to 1.5A, 3A, 4.5A, or 6A peak.
CURRENT_ RANGE_SCALE	Choose before the operation of the motor as fitting to the motor's maximum current and keep this value fixed within the application.	0…3	Scales the actual current range to 25%, 50%, 75%, or 100%. Apply values 0 to 2 scale down *CURRENT_RANGE* 0 for tiny motors.
IRUN	Current scale when motor is running. Scales coil current values as taken from the internal sine wave table. For high precision motor operation and best performance of CoolStep, work with a current scaling factor in the range 128 to 255 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. Take care to use a reduced current setting in case a high hysteresis setting is required for the motor: maximum current setting is limited by maximum setting of 255, respectively, 262-HYST/2, whichever is lower. Do not work with too small settings (poor performance). Default = 240	0…255	Scaling factor 1/256…256/256
IHOLD	Identical to *IRUN*, but for motor in standstill. The lower limit for *IHOLD* is specific to the application, and *IHOLD* may even be set to 0, if no holding torque is required. Default = 64		

IHOLDDELAY	Allows smooth current reduction from run current to hold current. *IHOLDDELAY* controls the number of clock cycles for the motor power down after *TZEROWAIT* in increments of 2^{11} clocks: 0 = instant power down, 1…255: current reduction delay per current step in multiples of 2^{11} clocks. **Example:** When using IRUN = 240 and IHOLD = 40, 200 current steps are required for hold current reduction. An *IHOLDDELAY* setting of 4, thus, results in a power down time of 4 × 200 × 2^{11} clock cycles, example, 100ms at 16MHz. Default = 7	0	Instant power down to IHOLD.
		1…255	1 × 2^{11} … 255 × 2^{11} clocks per current decrement.
IRUNDELAY	Controls the number of clock cycles for motor power-up after start is detected. Allows smooth current increment upon the 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 ramping time helps to reduce acoustic noise and avoids an overshoot on the power supply current. **Example:** When using IRUN = 240 and IHOLD = 40, 200 current steps are required for ramping up to run current. An *IRUNDELAY* setting of 4, thus, results in a power up time of 4 × 200 × 64 clock cycles, example, 3.2ms at 16MHz. Default = 4	0	Instant power up to IRUN.
		1...15	Delay per current increment step in multiples of *IRUNDELAY* × 64 clocks.
TPOWERDOWN	Sets the delay time after standstill (stst) detection to motor current power down. Time range is about 0 seconds to 4.2 seconds. A certain minimum value helps to measure the motor coil resistance in applications where major heat-up is expected. Default = 10	0	Fastest power down
		1…255	Delay in multiples of 2^{18} clocks.
fast_standstill	Controls the step pulse timeout until a standstill condition is detected. The default setting of 0 corresponds to a minimum velocity setting of 16. The fast setting is recommended in conjunction with the quick *TPOWERDOWN* timing. It corresponds to a velocity of 64.	0	2^{20} clocks
		1	2^{18} clocks

Slope Control

The TMC3241 supports a programmable slope control for its power stage outputs. It allows shaping the output slopes in four steps using the *SLOPE_CONTROL* setting. While a fast slope is best for the low power dissipation of the driver, a reduced slope causes less electromagnetic emission, and reduces the ringing of supply voltages and coupling to the cables adjacent to the motor cables. The slowest settings with <400V/µs should be used in low operating voltage scenarios only. When operating especially at voltages above 24V and current settings beyond 2A, it is recommended to go for the fastest slope setting.

Generally, start with the fastest slope and reduce slope control only if slope-related issues cannot be solved by different means like the optimization of the PCB layout or cabling.

Velocity-Based Mode Control

The TMC3241 allows the configuration of different chopper modes and modes of operation for optimum motor control. Depending on the motor load, the different modes can be optimized for lowest noise and high precision, highest dynamics, or maximum torque at highest velocity. Some of the features like CoolStep or StallGuard2 are useful in a limited velocity range. A number of velocity thresholds allow combining the different modes of operation within an application requiring a wide velocity range.

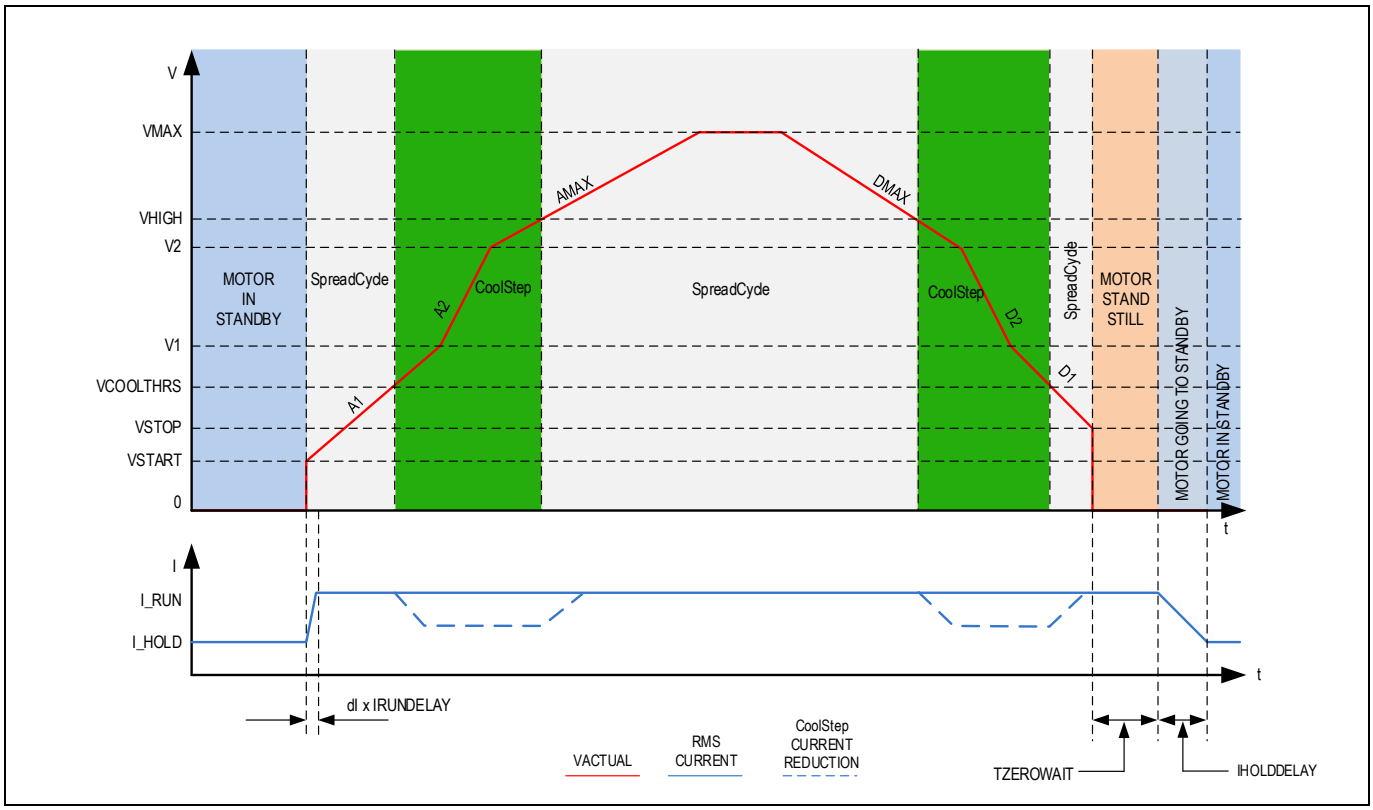


Figure 15. Choice of Velocity-Dependent Modes

Figure 15 shows the most important thresholds and required ordering. VHIGH and VCOOLTHRS are determined by the settings THIGH and TCOOLTHRS. The velocity is described by the time interval TSTEP between each of the two step pulses. This determines the velocity when an external step source is used. TSTEP always is normalized to 256 microstepping. This way, the thresholds do not have to be adapted when the microstep resolution is changed. The thresholds represent the same motor velocity, independent of the microstep settings. TSTEP is compared to these threshold values. A hysteresis of 1/32 TSTEP, respectively, 1/16 TSTEP automatically is applied to avoid continuous toggling of the comparison results when a jitter in the measured TSTEP occurs. The upper switching velocity is higher by 1/32, respectively, 1/16 of the value set as threshold (select with the configuration bit *small_hysteresis* in the *GCONF* register). The motor current scales to its run or hold level, dependent on the standstill flag *stst*.

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

Table 13. Velocity-Based Mode Control Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
stst	Indicates motor standstill in each operation mode. Time is 2^{20}, respectively, 2^{18} clocks after the last step pulse (as selected by *fast_standstill*). Default/reset: 0	0/1	Status bit, read-only.
TPOWERDOWN	This is the delay time after standstill detection (stst) of the motor--to-motor current power down. The time range is about 0 seconds to 4 seconds (with f_{CLK} = 16MHz). Setting 0 is no delay, 1 is one clock cycle delay. Further increment is in discrete steps of 2^{18} clock cycles. Default: 10	0…255	0: No delay 1: t_{CLK} 2 … 255 : Time in multiples of $2^{18} \times t_{CLK}$.

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. Default/reset: 0	0… 1048575	Status register, read-only. Actual measured step time in multiples of t_{CLK}.
TCOOLTHRS	*TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: • StallGuard2 and CoolStep are enabled, if configured, *TCOOLTHRS* ≥ *TSTEP* • Stop-on-stall is enabled and StallGuard stall output signal is enabled (if configured). Default: 0	0	CoolStep off
		1… 1048575	Setting to control the lower velocity threshold for operation with CoolStep and StallGuard.
THIGH	*TSTEP* ≤ *THIGH*: • CoolStep is disabled above the referenced velocity (motor runs with normal current scale). Default: 0	0	No upper velocity threshold.
		1… 1048575	Setting to control the upper threshold for operation with CoolStep.
small_hysteresis	Hysteresis for step frequency comparison based on *TSTEP* (lower velocity threshold) and (*TSTEP* × 15/16) – 1, respectively, (*TSTEP* × 31/32) - 1 (upper velocity threshold). Default: 1	0	Hysteresis is 1/16 (recommended with STEP/DIR source with increased jitter only).
		1	Hysteresis is 1/32.

Ramp Generator

The ramp generator allows motion based on the target position or target velocity. It automatically calculates the motion profile, taking into account the acceleration and velocity settings. The TMC3241 integrates a new type of ramp generator, which offers faster machine operation compared to the classical linear acceleration ramps. The EightPoint ramp generator allows adapting the acceleration ramps to the torque curves of a stepper motor. It uses three different acceleration settings, each for the acceleration and deceleration phases to allow for jerk-minimized ramps.

Enabling the internal Motion Controller

The TMC3241 per standard uses the step and direction Interface as the default input to the driver. In some applications, it may be desired to either completely work with the internal motion controller, or to combine the motion controller functionality, and the option to use an external source for step generation. To allow both, the flag *GCONF.threephase_mcc* allows switching the step source for the driver to the internal motion controller and at the same time change the function of the STEP input to REFL function and DIR input to REFR function. However, the internal ramp generator position counter does not react to step pulses commanded in the STEP and DIR interface mode. Therefore, any ramp generator motion should be terminated before switching to the STEP and DIR mode, and vice versa when switching back.

Real-World Unit Conversion

The TMC3241 uses its internal or external clock signal as a time reference for all internal operations. Thus, all time, velocity, and acceleration settings are referenced to f_{CLK}. For applications with high requirements on absolute timing for single motors, or synchronicity of several motors, it is recommended to use an external quartz oscillator as a time base, or to provide a clock signal from a microcontroller.

v[TMC3241] and a[TMC3241] are the internal units of the TMC3241. These are the values to write to the velocity/acceleration registers of the TMC3241.

Calculator and evaluation tools are available from the product website.

Table 14. Ramp Generator Parameters vs. Units

PARAMETER/SYMBOL	UNIT	DESCRIPTION
f_{CLK}	[Hz]	Clock frequency of the TMC3241; typically 16MHz.
s	[s]	Second
US	Microstep	A microstep is one advance in the 1024 entry sine wave table.
FS	Fullstep	A three-phase motor offers six fullsteps per electrical rotation.
USC - microstep count	-	Microstep resolution in number of microsteps (refers to the number of microsteps between two fullsteps – normally 171).
FSC - fullstep count	-	Motor fullsteps per rotation, for example, 300.
µstep velocity v[Hz]	Microsteps/s	$v[Hz] = v[TMC3241] \times (f_{CLK}[Hz]/2^{24})$
µstep acceleration a[Hz/s]	Microsteps/s^2	$a[Hz/s] = a[TMC3241] \times f_{CLK}[Hz]^2/2^{41}$
Rotations per second v[rps]	Rotations/s	v[rps] = v[microsteps/s]/USC/FSC
RPS acceleration a[rps/s^2]	Rotations/s^2	a[rps/s^2] = a[microsteps/s^2]/USC/FSC
Ramp steps[microsteps] = rs	Microsteps	$rs = (v[TMC3241])^2/a[TMC3241]/2^8$ microsteps during linear acceleration ramp (assuming acceleration from 0 to v).
TSTEP, *Txxx_THRS*	-	$TSTEP = f_{CLK}/f_{171STEP} = f_{CLK}/(f_{STEP} \times 171/USC)$ $= 2^{24}/(VACTUAL \times 171/USC)$ The time reference for velocity thresholds is referred to the actual 1/171 microstep frequency ($f_{171STEP}$) of the step input, respectively, velocity v[Hz].
Ramp generator update rate	[Hz]	$f_{UPDATE} = f_{CLK}/512$ *VACTUAL* updates with this frequency.

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

Motion Profiles

Ramp Mode

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

Three different sets of acceleration and deceleration can be combined freely. The transition velocities V1 and V2 allow for velocity-dependent switching between the three acceleration and deceleration settings. A typical high velocity application uses lower acceleration and deceleration values at higher velocities, as the motor's torque declines at higher velocity. When considering friction in the system, it is clear the deceleration capability of the system is quicker than the acceleration capability. Thus, the deceleration values can be set higher in many applications. This way, the operation speed of the motor in time-critical applications is maximized.

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

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

Note: *VSTART* and *VSTOP*

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

Take care to always set *VSTOP* identical to or above *VSTART*. This ensures that even a short motion can be terminated successfully at the target position.

Set *TVMAX* to zero to disable jerk-reduction.

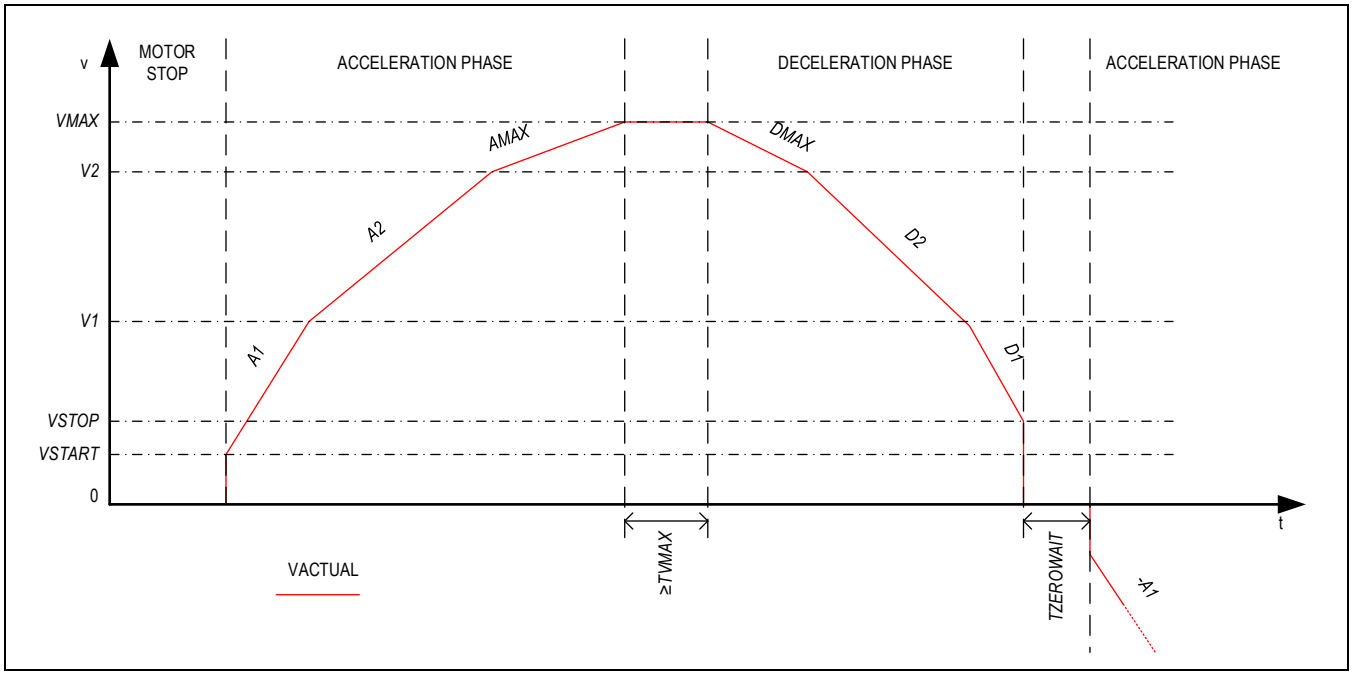


Figure 16. Ramp Generator Velocity Trace Showing Second Move into Negative Direction

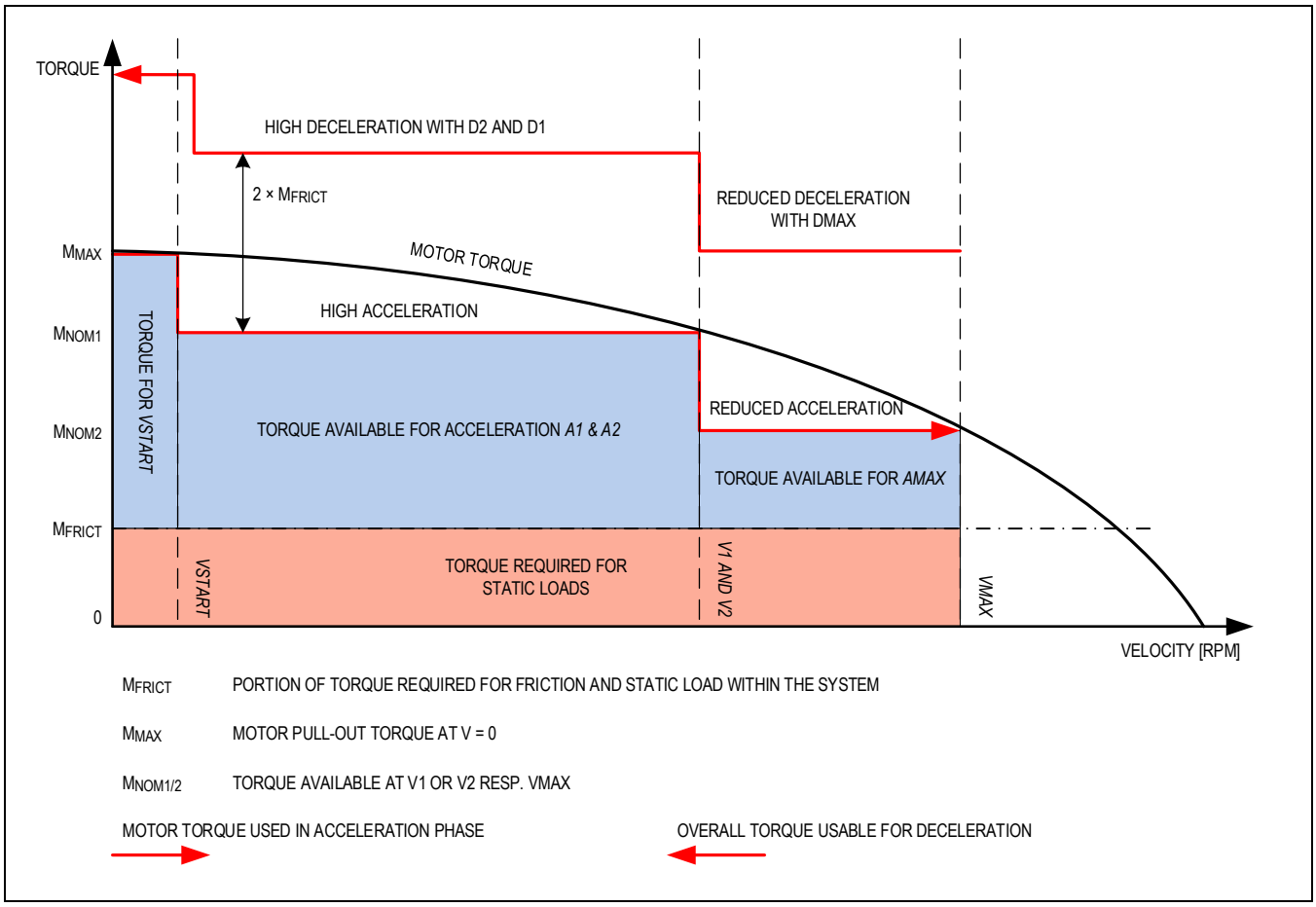


Figure 17. Illustration of Optimized Motor Torque Usage with the Ramp Generator

Eight-Point Ramp

Start and Stop Velocity

When using increased levels of start and stop velocity, it is clear that a subsequent move into the opposite direction provides a jerk identical to *VSTART* + *VSTOP*, rather than only *VSTART*. As the motor probably is not able to follow this, set a time delay for a subsequent move by setting *TZEROWAIT*. An active delay time is flagged by the flag *t_zerowait_active*. Once the target position is reached, the flag position_reached is active.

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

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

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

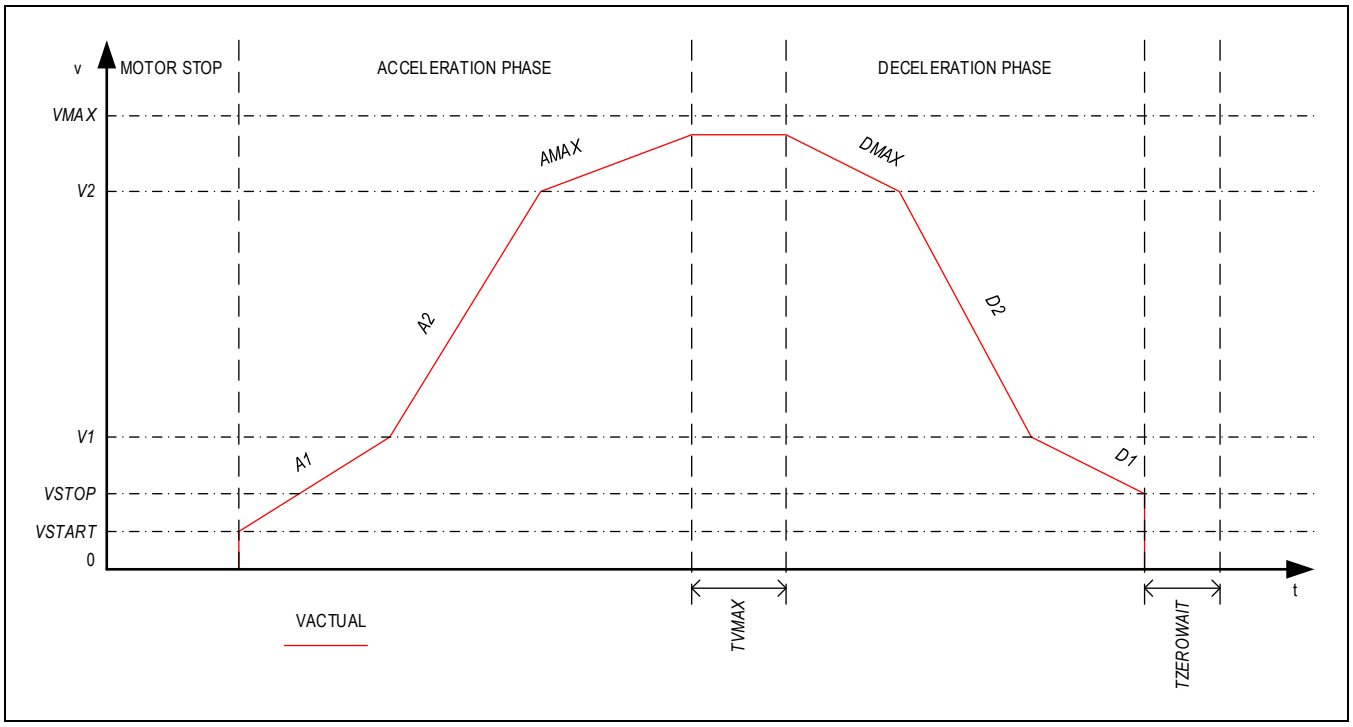


Figure 18. Eight-Point Ramp with VMAX Not Reached Due to Too Low Distance

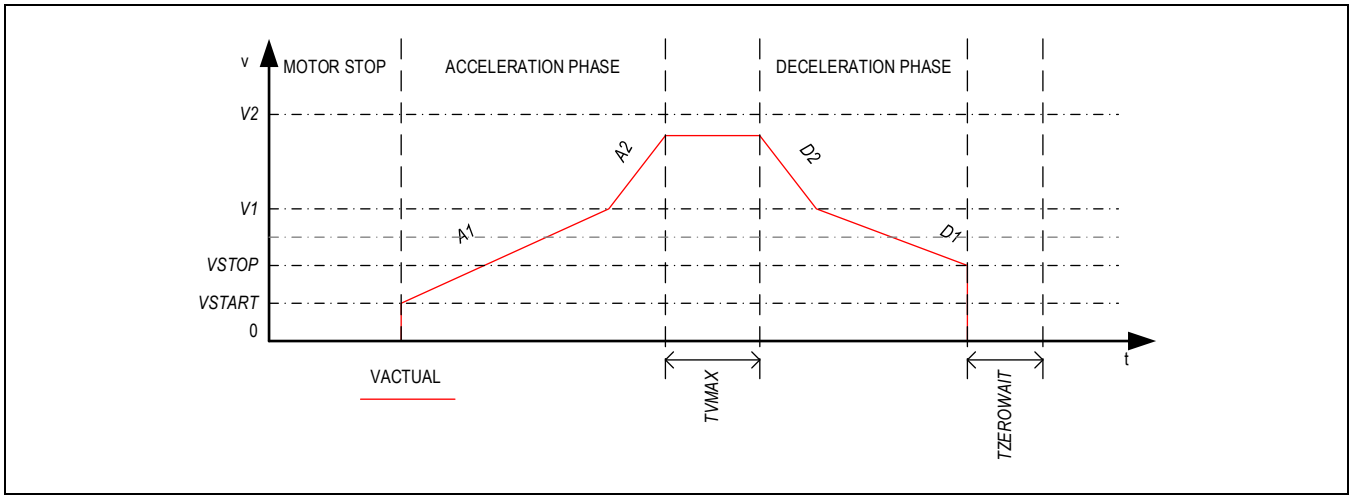


Figure 19. V2 Not Reached and No AMAX and DMAX Phase Due to Low Distance

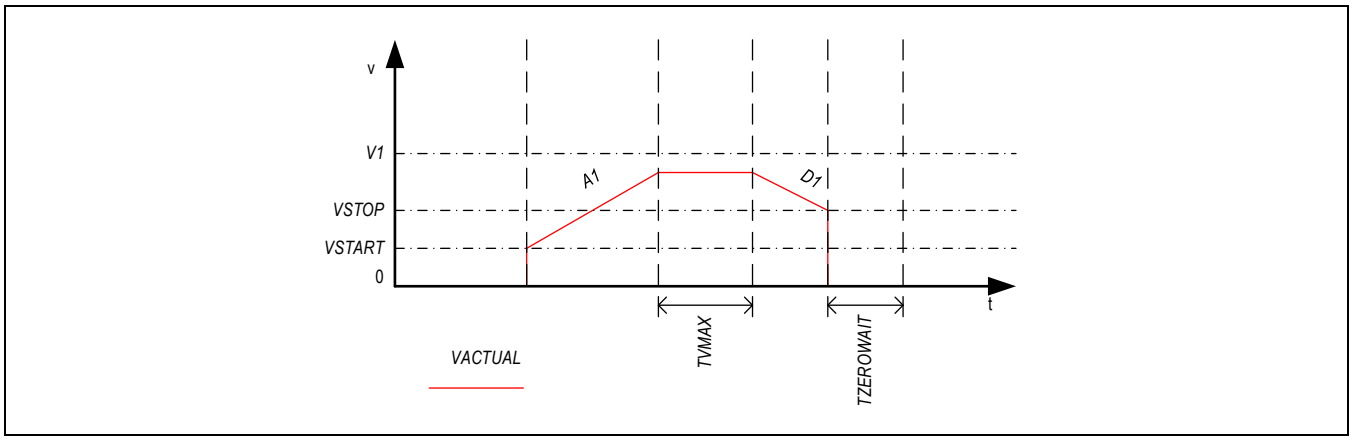


Figure 20. V1 Not Reached Due to Low Distance

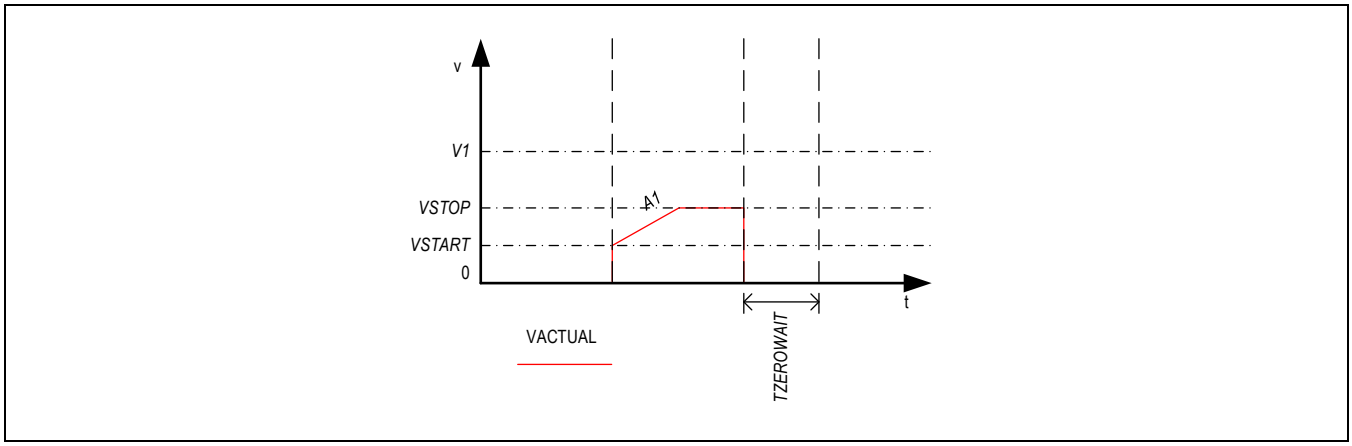


Figure 21. TVMAX Not Kept Due to Low Distance

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

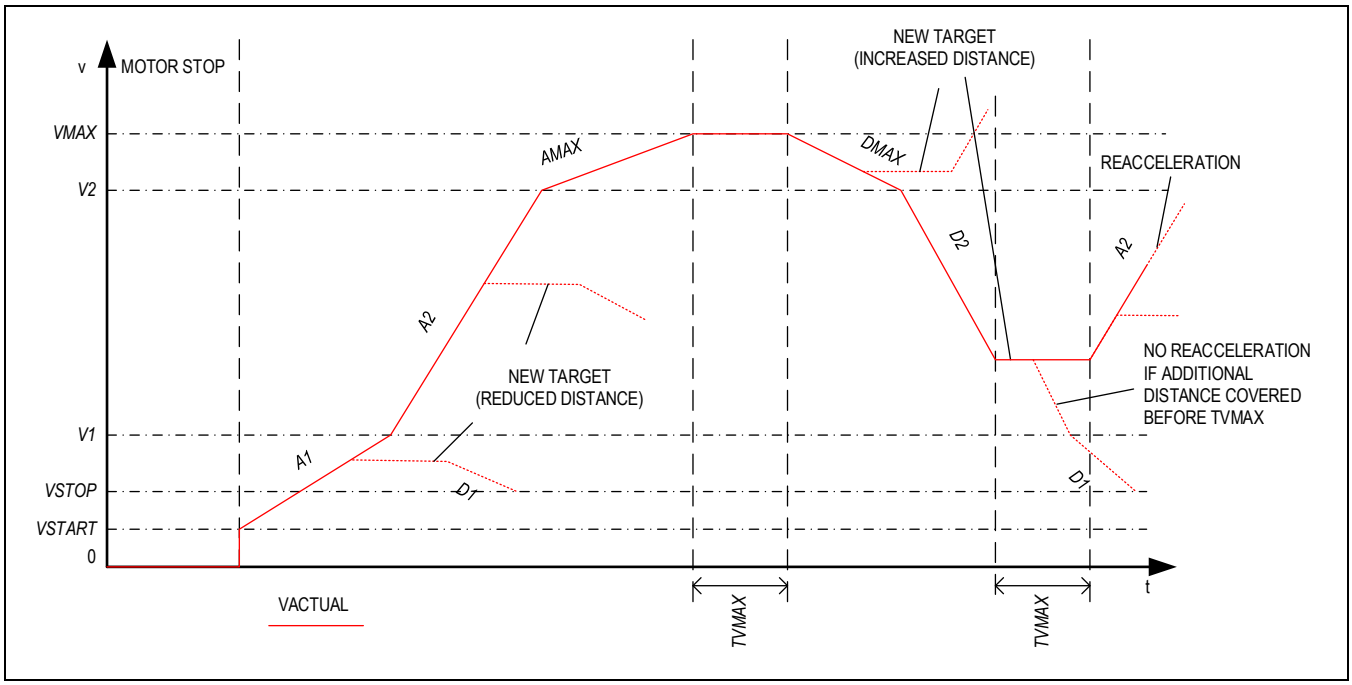


Figure 22. Eight-Point Ramp Examples with On-the-Fly Target Position Change

Velocity Mode

In contrast to the previous product generations, velocity mode movements also use different acceleration and deceleration settings. For velocity mode, only *AMAX*, *A1*, *V1*, *A2, V2,* and *VMAX* are relevant for both acceleration and deceleration. If *V1* and *V2* are zero, the ramp generator always uses *AMAX* to accelerate or decelerate to *VMAX* in this mode.

To decelerate the motor to standstill, it is sufficient to set *VMAX* to zero. The flag vzero signals the standstill of the motor. The flag velocity_reached always signals the target velocity is reached.

Early Ramp Termination

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

Options to terminate the motion using acceleration settings:

a) Switch to the velocity mode, set *VMAX* = 0 and *AMAX* (plus *A1*, *A2*, if used) to the desired deceleration value. This stops the motor using a linear ramp.

b) For a stop in the positioning mode, set *VSTART* = 0 and *VMAX* = 0. *VSTOP* is not used in this case. The driver uses *AMAX*, *A1*, and *A2* (as determined by *V1* and *V2*) for decelerating to zero velocity.

c) For a stop using *DMAX*, *D1*, *D2,* and *VSTOP*, trigger the deceleration phase by copying *XACTUAL* to *XTARGET*. Set *TZEROWAIT* sufficiently to allow the CPU to interact during this time. The driver decelerates and eventually comes to a stop. Poll the actual velocity to terminate the motion during *TZEROWAIT* time using option a) or b).

d) Activate a stop switch. This can be done by the hardware input, example, using a wired 'OR' to the stop switch input. If the hardware input is not used, and the REFL and REFR are tied to a fixed level, enable the stop function (*stop_l_enable*, *stop_r_enable*) and use the inverting function (*pol_stop_l*, *pol_stop_r*) to simulate the switch activation.

e) Utilize the virtual stop switches (*VIRTUAL_STOP_L*, *VIRTUAL_STOP_R*). The position comparison (*XACTUAL* vs. *VIRTUAL_STOP_L/R*) then triggers a stop accordingly.

Application Example: Joystick Control

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

Realize Joystick Control:

- Use the positioning mode to control the motion direction and to set the motion limit(s) as fitting to the target direction. The limits, additionally, can be entered into the virtual stop switches (*VIRTUAL_STOP_L* and *VIRTUAL_STOP_R*) for double safety.
- Modify *VMAX* depending on the joystick input at any time in the range *VSTART* to the maximum value. With *VSTART* = 0, stop the motion by setting *VMAX* = 0. The motion controller uses *A1*, *A2*, and *AMAX* as determined by V1 and V2 to adapt the velocity for ramping up and down. Rewrite *XTARGET* for any change.
- In case the acceleration settings are not modified, do not rewrite *XTARGET*, just modify *VMAX*.
- *DMAX*, *D1*, *D2*, and *VSTOP* are only used when the ramp controller slows down due to reaching the target position, or when the target position is modified to point to the other direction.

Reference Switches

Positioning applications require setting an absolute reference position before the normal operation of the drive.

The reference position can be found using a mechanical stop, which can be detected by StallGuard2 or by a reference switch.

In case of a linear drive, the mechanical motion range must not be exceeded. This can be ensured by enabling the stop switch functions for the left and right reference switches. Therefore, the ramp generator responds to several stop events, as configured in the *SW_MODE* register. There are two ways to stop the motor:

a) Abrupt stop, when the switch is hit. This is useful in an emergency and for StallGuard-based homing.

b) Soft deceleration to zero upon reaching the switch. The motor can be decelerated with ramp deceleration settings (*DMAX*, *V2*, *D2*, *V1*, *D1*) using the soft stop function (bit *en_softstop* = 1). With this method, it slightly moves into the stop switch, as defined by the deceleration settings. Therefore, the stop switches must be positioned with some tolerance to the mechanical end position.

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

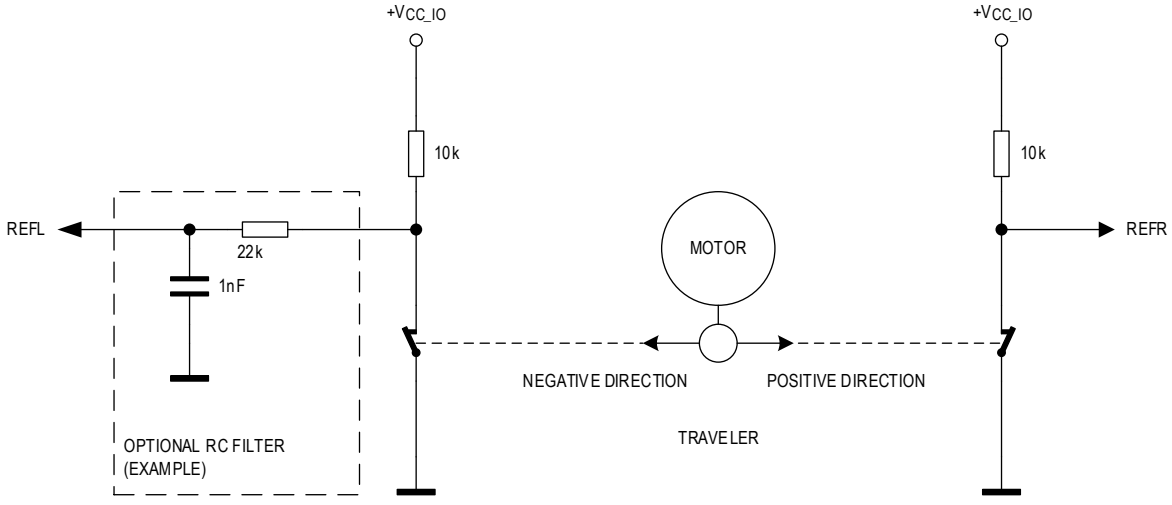


Figure 23. Using Reference Switches (Example)

Normally, open or normally closed switches can be used by programming the switch polarity or selecting the pullup or pulldown resistor configuration. A normally closed switch is failsafe with respect to an interrupt of the switch connection. Fitting switch types are mechanical switches, photo interrupters, or hall sensors.

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

In case of long cables, additional RC filtering might be required near the TMC3241 reference inputs. Adding an RC filter also reduces the danger of destroying the logic level inputs by wiring faults, but it adds a certain delay, which should be considered with respect to the application.

Implementing a Homing Procedure:

A homing procedure is required to allow absolute positioning within a motion range.

Table 15. Implementing a Homing Procedure

NUMBER	TASK	DETAIL
1	Switch Not Pressed	Make sure the home switch is not pressed, example, by moving away from the switch.
2	Activate Position Latching	Activate position latching upon the desired switch event and activate motor (soft) stop upon active switch. StallGuard2-based homing requires a hard stop (*en_softstop* = 0).
3	Start Motion	Start a motion ramp into the direction of the switch. (Move to a more negative position for a left switch, to a more positive position for a right switch). Timeout this motion using a position ramping command.
4	Stop Motion	As soon as the switch is hit, the position is latched and the motor is stopped. Wait until the motor is in standstill again by polling the actual velocity *VACTUAL*, or checking vzero or the standstill flag.
5	Calculate Offset	Switch the ramp generator to hold mode, and calculate the difference between the latched position and actual position. For StallGuard2-based homing or when using hard stop, *XACTUAL* stops exactly at the home position. So, there is no difference (0).
6	Write Absolute Position	Write the calculated difference into the actual position register. Now, homing is finished. A move to position 0 brings back the motor exactly to the switching point. In case StallGuard2 is used for homing, clear the StallGuard2 stop event *event_stop_sg* to release the motor from the stop condition.

Homing with a Third Switch

Some applications use an additional home switch, which operates independent of the mechanical limit switches. The encoder functionality of the TMC3241 provides an additional source for position latching. It allows using the N channel input to snapshot *XACTUAL* with a rising or falling edge event, or both. This function also provides an interrupt output.

Table 16. Homing with a Third Switch Procedure

NUMBER	TASK	DETAIL
1	Activate Latching	Activate the latching function (*ENCMODE*: set *ignoreAB*, *clr_cont*, *neg_edge* or *pos_edge*, and *latch_x_act*). The latching function can then trigger the interrupt output (check by reading *n_event* in *ENC_STATUS* when the interrupt is signaled at DO0).
2	Move to Switch	Move to the direction where the N channel switch should be. In case the motor hits a stop switch (REFL or REFR) before the home switch is detected, reverse the motion direction.
3	Read Position	Read out *XLATCH* once the switch is triggered. It gives the position of the switch event.
4	Write Absolute Position	After the detection of the switch event, stop the motor, and subtract *XLATCH* from the actual position. (A detailed description of the required steps is given in the homing procedure above).

Virtual Reference Switches

The TMC3241 supports virtual reference switches to support applications that only have a single or no reference switch (StallGuard homing) to safely limit the physical motion range. The virtual stop switches become active when the actual motor position (*XACTUAL*) exceeds *VIRTUAL_STOP_R* during a movement into the positive direction or falls below VIRTUAL_STOP_L during a movement in the negative direction. Enable virtual stop switches by setting *en_virtual_stop_l* , respectively, *en_virtual_stop_r*. Each virtual stop switch blocks only the motion into the respective direction.

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

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

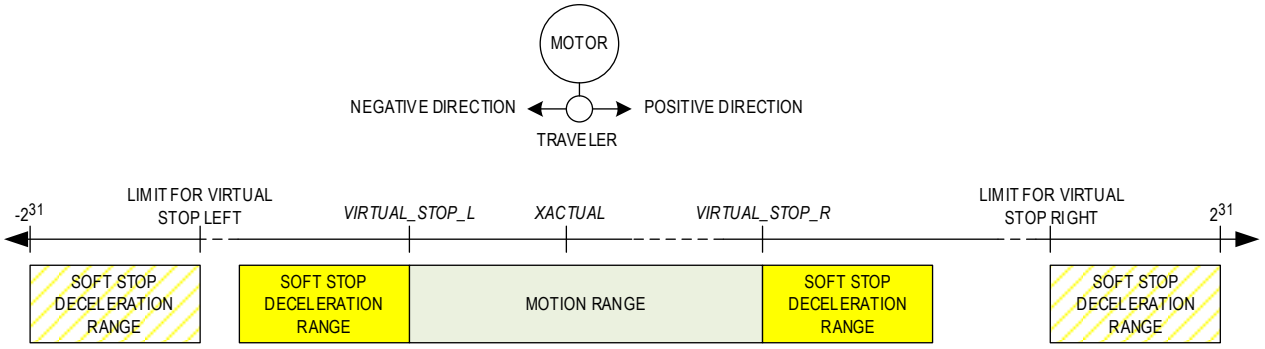


Figure 24. Virtual Stop Switches and Limit Visualization

Ramp Generator Response Time

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

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

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

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

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

External STEP/DIR Driver

The TMC3241 allows using the internal ramp generator to control an external STEP/DIR driver like the ADI-Trinamic TMC2262, TMC2160, or TMC2240 for synchronous dual-axis stepper applications. In this configuration, the internal driver is normally not used, but it may be used in addition to the external driver, example, when two motors move synchronously.

The DO0 and DO1 outputs are enabled for the STEP and DIR output by setting *GCONF* flags *do0_step* and *do1_dir*. Additional internal driver features like µDcStep and automatic motor current control are not available in this mode.

A low level on pin DO1 corresponds to a step in the positive direction (*XACTUAL* increasing), a high level to a step into the negative direction (*XACTUAL* decreasing).

The two options that can be selected for the external STEP signal:

- Double edge (*GCONF.length_step_pulse* = 0). Each toggle of the STEP output corresponds to a step. DIR is valid at least one CLK cycle in advance. Enable the *dedge* function on the external driver to match.

- Single edge (*GCONF.length_step_pulse* > 0). Each positive pulse on the STEP output corresponds to a step. The pulse length is controlled by *length_step_pulse* in CLK cycles. The DIR signal only changes in between step pulses and keeps a minimum setup time equal to the STEP pulse length in advance to the next step pulse.

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

Position Compare Functions

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

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

Table 17. X_COMPARE_REPEAT Options and Periodic Pulse Behavior

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

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

ABN Incremental Encoder Interface

The TMC3241 is equipped with an incremental encoder interface for ABN encoders. The encoder gives positions through digital incremental quadrature signals (usually named A and B) and an index signal (usually named N for null, Z for zero, or I for index).

N Signal

The N signal can be used to clear the encoder position counter *X_ENC* or to take a snapshot to *ENC_LATCH* when passing it. To continuously monitor the N channel and trigger the clearing of the encoder position or latching of the position, where the N channel event is 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.

Check for *n_event* or map the interrupt output to show this event. Clear *n_event* afterwards.

Some encoders require a validation of the N signal by a certain configuration of A and B polarity. This can be controlled by the *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 the A and B channels.

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

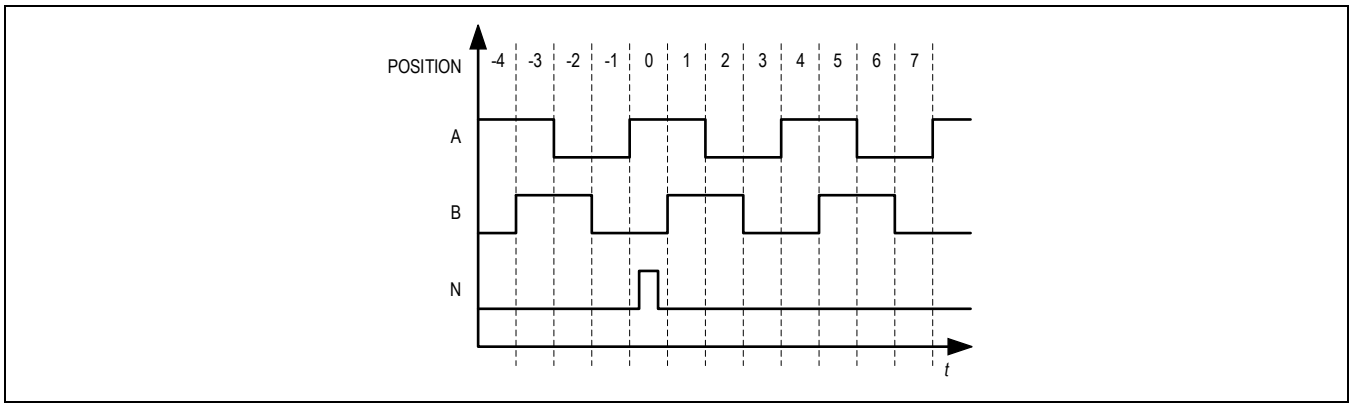


Figure 25. Outline of ABN Signals of an Incremental Encoder

The Encoder Counter X_ENC

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

The Register ENC_STATUS

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

The Encoder Constant ENC_CONST

The encoder constant (or encoder factor) *ENC_CONST* is added to or subtracted from the encoder counter on each polarity change of the quadrature signals AB of the incremental encoder. The encoder constant *ENC_CONST* represents a signed fixed point number (16.16) to facilitate the generic adaption between motors and encoders. In binary mode, the lower 16 bits represent a number from 0 to 65535, describing increments of 1/2^16 = 1/65536 per encoder event. In decimal mode, the lower 16 bits represent a number between 0 and 9999 describing increments of 1/10000 per encoder event. 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 the inversion of the counting direction to match the motor and encoder direction.

A negative encoder factor is used to negate the encoder direction.

- In binary mode, a negative encoder constant is the two's complement of the encoder constant:
  - (FACTOR.FRACTION) = (2^{16} - (FACTOR + 1)) × (2^{16} - FRACTION)
- In decimal mode, a negative encoder constant is calculated likewise using the following equation:
  - (FACTOR.DECIMALS) = (2^{16} - (FACTOR + 1)) × (10000 - DECIMALS)

Examples for setting the encoder constant:

• Encoder factor 1.0	• *ENC_CONST* = 0x0001.0x0000 = FACTOR.FRACTION
• Encoder factor -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.

Setting the Encoder to Match Motor Resolution

Encoder example settings for a 1.2° step angle motor:

- USC = 170.666667 microsteps (1024 wave entries/six fullsteps)
- FSC = 300 fullstep motor

- Factor = FSC × USC/encoder resolution

Table 18. Encoder Example Settings for a 300 Fullstep Motor with 171 Microsteps

ENCODER RESOLUTION	REQUIRED ENCODER FACTOR	COMMENT
200	256	
360	142.2222 = 9320675.5555/2^{16} (binary) = 1422222.2222/10000 (decimal)	No exact match possible!
500	102.4 = 6710886.4/2^{16}(binary) = 1024000/10000 (decimal)	Exact match with decimal setting only.
1000	51.2	Exact match with decimal setting only.
1024	50	
4000	12.8	Exact match with decimal setting only.
4096	12.5	
16384	3.125	

Calculation example:

The encoder constant register is programmed to 51.2 in decimal mode. Therefore, set:

ENC_CONST = 51 × 2 ^{16} + 0.2 × 10000 = 51.2000 = 0x0033.0x07D0

Sine Wave Lookup Table

The TMC3241 provides a programmable lookup table to store the microstep current wave. As a default, the table is preprogrammed with a sine wave, which is a good starting point for most stepper motors. Reprogramming the table to a motor-specific wave allows drastically improved microstepping, especially with low-cost motors. The user benefits are:

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

Microstep Table

The sine wave table uses 1024 entries for an electrical field rotation. Therefore, each six motor fullsteps are mapped to 1024 microsteps. This means, a native motor fullstep is divided into 1024/6 = 170.67 microsteps. Thus, a 300 fullstep three-phase stepper offers 51200 microsteps per rotation, or an eight pole-pair PMSM motor offers 8192 fullsteps per rotation.

To minimize the required memory and amount of data to be programmed, only a quarter of the wave is stored. The internal microstep table maps the microstep wave from 0° to 90°. It is 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*. The 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*, and more

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

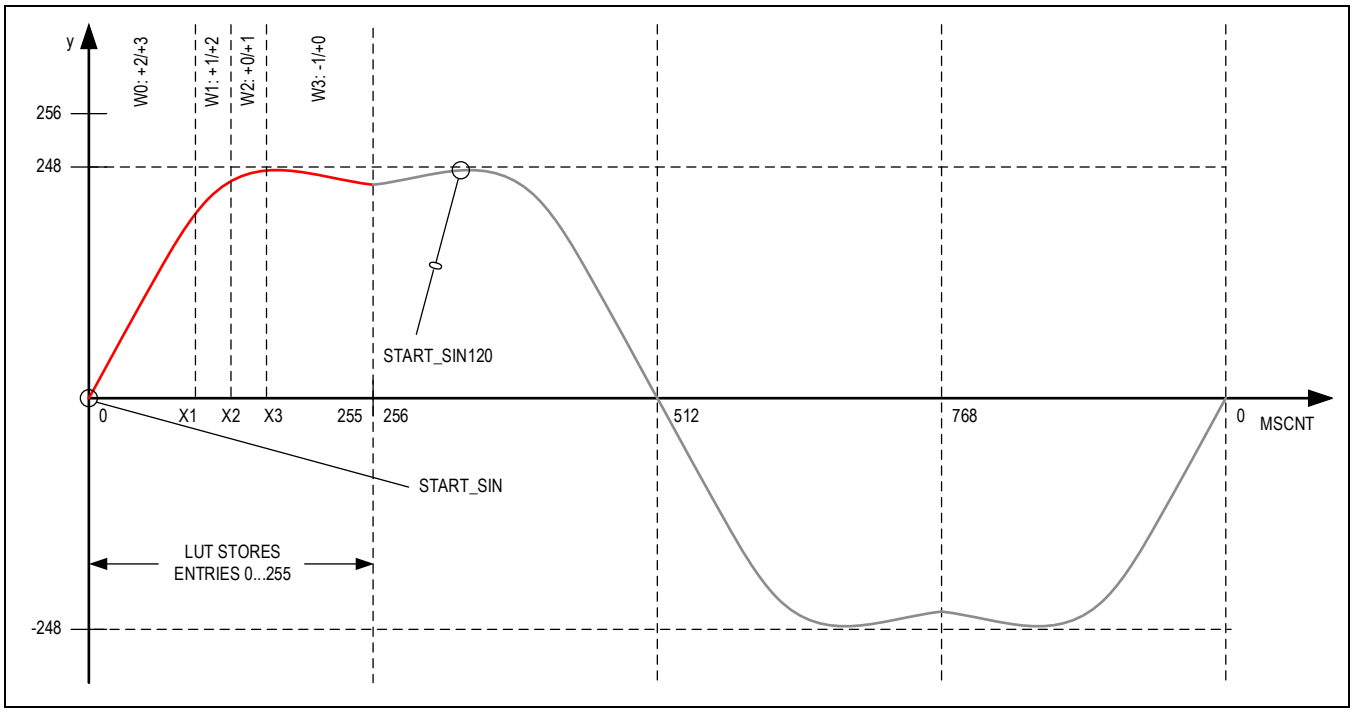


Figure 26. LUT Programming Example

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_U* and *CUR_V*. However, the incremental coding requires an absolute initialization, especially when the microstep table is modified. Therefore, *CUR_V* and *CUR_U* are initialized from *START_SIN120* and *START_SIN* whenever *MSCNT* passes zero.

Matching the Phase Shift to the Motor

Two registers control the starting values of the sine and cosine wave.

- The starting value of the sine wave at microstep 0 should be zero, but it may be offset to 1 or 2 in special cases when modifying waves. It can be defined using the starting point register *START_SIN*.
- The start of the 120° shifted sine wave for the second motor coil must be programmed to *START_SIN120*. For a monotonous wave, this is the amplitude of the wave and corresponds to the sum of increments within a quarter period. With this, the register stores the resulting table entry for a phase shift of 120° for a three-phase motor.
- The third wave CUR_W cannot be read from the device and is calculated from the other two using the following equation: $CUR_W = -(CUR_U + CUR_V)$. When modifying sine wave settings, make sure the resulting difference still is limited to the valid range of ±255 (respectively, ±248 for the default wave).

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 puts the motor into freewheeling. Further, it is a safe hardware function whenever an emergency stop not coupled to the 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.stop_enable* to activate this option. Whenever ENCA is pulled high and as long as it stays high, the motor stops abruptly and goes to the power down state, as configured through *IHOLD* and *IHOLD_DELAY*.

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 *GSTAT.register_reset* flag).

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

If SLEEPN is kept at GND, the IC goes into a 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 the sleep mode and reset condition, the registers must be reconfigured.

The driver should be re-enabled as the last step of the reconfiguration to avoid operating the motor with the wrong settings.

Do not put the drive to sleep mode or reset the driver while a motor is driven with a high current or rotating at high velocity, as energy fed back from the motor might damage the chip!

If not used, connect to V_{CC_IO}.

Diagnostics and Protections

The TMC3241 provides a complete set of diagnostic and protection capabilities, like short-to-GND and short-to-V_S (overcurrent) protection, as well as undervoltage detection. Detecting 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 TMC3241 allows the measurement and read out of the chip temperature. Driver temperature measurement allows preventive action for extreme thermal conditions, for example, by reducing motor current, or by interrupting motion to allow the cool-down of the motor and driver.

For improved system reliability and overall circuit protection, the TMC3241 contains an overvoltage comparator and a trigger output OV (as optional function for mapping at pins DO0 and DO1) to control the external switches in case of excessive supply voltage increase.

In addition, DO0 and DO1 can be configured as outputs of the internal *real-time on-chip scope interface* to visualize the internal parameters as analog signals on a scope.

Diagnostic Outputs

The TMC3241 comes with two configurable diagnostic outputs, DO0 and DO1.

Both pins allow mapping a variety of status information bits usable for CPU interrupts, detection of emergency conditions like an overtemperature or overvoltage, or triggering of external events. The options are configured using the register *DO_CONF*.

Either an open-drain (active low) output signal (default *doosen0x_nOE_PP* = 0) or an active high push-pull output signal (*do0x_nOE_PP* = 1) can be chosen. When using the negative active open-drain output, multiple driver output signals can be logically ORed. An external pullup resistor in the range 4.7kΩ to 100kΩ is required. To safely determine a reset condition, monitor the reset flag by SPI or read out any register to confirm the chip is powered up.

In an optional mode, DO0 and DO1 output analog signals are helpful for debugging and parameter tuning.

- The diagnostic output pin DO0 is a functionally shared pin with RT-OSCI DAC channel 0.
- The diagnostic output pin DO1 is a functionally shared pin with RT-OSCI DAC channel 1.

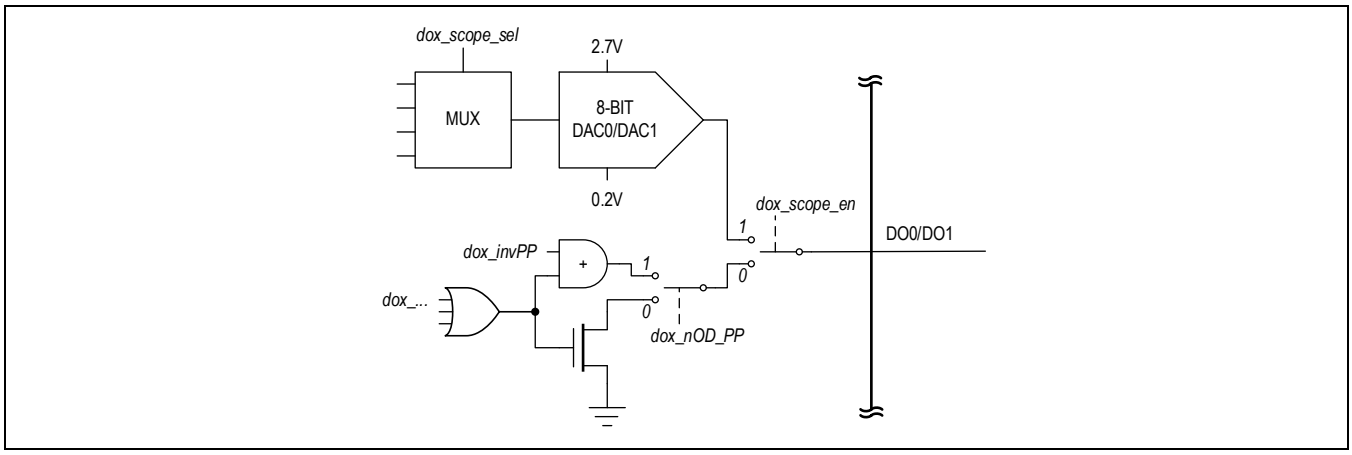


Figure 27. Schematic of DO0 and DO1 Outputs

Overcurrent Protection

Overcurrent protection (OCP) protects the device against short circuits to the rails (supply voltage and ground) and among the outputs (OUTU, OUTV, OUTW).

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

The short detection in the low-side MOSFETs automatically is controlled together with the current range (*CURRENT_RANGE*). This way, it reacts more sensitively as fitting to the selected motor type and peak current. Considering this, selecting the lowest *CURRENT_RANGE* fitting the motor gives the best protection, for example, in case of miswiring or insulation defect. With this, a reduction of *CURRENT_RANGE* while the motor is driven to a high coil current can lead to overcurrent detection. Therefore, *CURRENT_RANGE* should not be modified while the motor is enabled and driven at a current higher than the peak current of the target *CURRENT_RANGE*.

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 affected bridge is disabled immediately. The chopper cycle on the affected coil is terminated and the short counter is increased by each short circuit. It is decreased by one for each phase polarity change. To avoid false triggering upon an ESD event, the short protection retries thrice in consecutive chopper cycles, before a fault flag (*DRV_STATUS.s2g*, *DRV_STATUS.s2vs*) is set and the bridge is continuously disabled.

The device remains active and allows configuration and status read out. A bridge fault does not cause any action in the ramp generator or sequencer, and all consequent actions should be covered by the software.

To re-enable the power bridge, it must be disabled and re-enabled using the DRV_ENN pin, or setting *TOFF* = 0 in *CHOPCONF* and re-enabling the bridges with *TOFF* > 0.

Thermal Protection and Shutdown

The TMC3241 has an internal thermal protection.

To prevent the chip from thermal damage in case of severe overheating, the die temperature is monitored near to each power MOSFET. Whenever the temperature measured by one of the sensors exceeds 165°C (typical value), a fault indication flag (*ot* in *DRV_STATUS*) is raised and the driver is three-stated until the junction temperature drops below approximately 145°C (typical value). After that, the driver is re-enabled.

To allow preventive action before overheating, the TMC3241 supports ADC-based configurable thermal prewarning levels. This can be configured in the register *OTW_OV_VTH* using the parameter *OVERTEMPPREWARNING_VTH*. The ADC senses the chip temperature at the center of the die with some distance to the driver stages and does not see individual heat up of each driver stage. Therefore, the prewarning threshold should be set with an application-situation-specific distance of minimum roughly >20°C to 40°C to the shutdown temperature.

Thus, setting a prewarning limit does not necessarily mean the TMC3241 does not shut down before reaching this limit, as heat may be unevenly distributed on the IC, even if it is set at a lower temperature. Heat mainly is generated by the motor driver stages and is distributed over the six on-chip power MOSFETs, depending on individual coil currents and

polarity. Most critical situations, where the driver MOSFETs can be overheated, are avoided by short-circuit protection. For many applications, the overtemperature prewarning indicates an abnormal operation situation, and can be used to initiate user warning or power reduction measures like motor current reduction. The thermal shutdown is just an emergency measure and temperature rising to the shutdown level should be prevented by design.

Temperature Measurement

The TMC3241 offers functions to measure the internal chip temperature as well as motor temperature.

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

Chip Temperature Measurement

Besides the overtemperature prewarning and 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 for *ADC_TEMP* and *OVERTEMPPREWARNING_VTH* is calculated using the following formula:

$$ADC_TEMP = \frac{(\text{TEMP} + 264.6)}{1.042}$$

$$TEMP[^\circ C] = ADC_TEMP \times 1.042 - 264.6$$

Note: The temperature measurement shows the temperature at the center of the IC. Depending on the motor current and PCB size, the power stage temperature can be significantly higher. Check with a thermal imaging camera to determine the actual temperature difference.

Overvoltage Protection and OV Output

A stepper motor application can generate significant overvoltage, especially when the motor is quickly decelerated from a high velocity, or when the motor stalls, or when it is quickly rotated by an external force. In these situations, energy from the mechanical rotation and energy stored in the motor's magnet field are fed back to the supply rails through the driver output stage.

For typical NEMA17 or larger motors, and also for smaller motors with sufficient flywheel mass, the energy fed back can be substantial, so that the power capacitors and circuit consumption are not sufficient to keep the supply within the limits. A Zener diode can be used for overvoltage protection in systems with low energy feedback, but its inner resistance is comparatively high and power dissipation capability is limited.

To protect the driver as well as the connected circuitry in systems with high energy fed back, the TMC3241 has an overvoltage detection and protection mechanism. It permanently measures the supply voltage using an ADC and compares the result to a programmable threshold. Whenever the ADC result reaches or exceeds the threshold, the internal overvoltage flag DRV_STAT.ov is raised. The overvoltage signal can be mapped to one or both DO outputs, as well as to the encoder N pin (ENCN/OVN) to provide a dedicated output in environments where the encoder N-channel is not used. The ADC updates with 2.45kHz (typical), leading to an update rate of <1ms for OV signal. This way, an external transistor driving an overvoltage dumping resistor chops with approximately 1kHz to keep the supply within the limits.

The upper level for the supply voltage for a given application can be configured in the register OTW_OV_VTH using the parameter *OVERVOLTAGE_VTH*. Keep sufficient headroom to the normal supply voltage, taking into account all tolerances, to ensure the overvoltage detector does not trigger under normal situations, as this might destroy the dumping circuit.

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

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

$$V_S = ADC_VSUPPLY \times 140.9\text{mV}$$

$$OVERVOLTAGE_VTH = V_{S(MAX)}/140.9\text{mV} + 1$$

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

As soon as and as long as *ADC_VSUPPLY* is greater or equal to *OVERVOLTAGE_VTH*, the OV output signal is active (ENCN/OVN is driven low).

An external gate driver should be used to drive the overvoltage protection transistor that dumps energy into a power resistor. As the function first must be mapped to the selected output, the overvoltage protection mechanism should be explicitly enabled by an external signal once the configuration is done. Choose the power-resistor as fitting to the maximum current fed back from the motor at the threshold voltage. The typical motor feedback current is much less than the rated motor coil current. With this, a reference value for the dump resistor can be calculated:

$$R_{DUMP} = V_{S(MAX)}/I_{COILNOM}$$

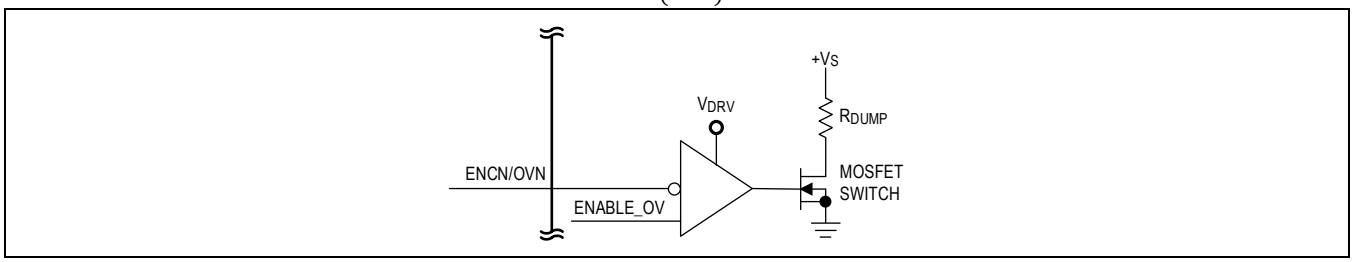


Figure 28. Brake Chopper Circuit Example

Short Protection (Short-to-GND and Short-to-V_S)

The TMC3241 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, for example, when touching the conducting parts connected to the system ground. The short detection is protected against spurious triggering, for example, by electrostatic discharges (ESD), by retrying thrice before switching off the motor.

Once a short condition is safely detected, the corresponding driver bridge is switched off and the *DRV_STATUS.s2g*, respectively, *DRV_STATUS.s2vs* flag is set. To restart the motor, intervene by disabling and re-enabling the driver. Note that the short-to-GND protection cannot protect the system and the power stages for all possible short events as a short event is rather undefined, and a complex network of external components may be involved. Therefore, short circuits should basically be avoided.

Depending on the full-scale current setting, the low-side short protection triggers at different overcurrent protection thresholds. See the **Electrical Characteristics** table for the detailed values.

Open-Load Diagnostics

Interrupted cables are a common cause for systems failing, for example, when connectors are not firmly plugged. The TMC3241 detects open-load conditions by checking if it can reach the desired motor coil current. This way also, undervoltage conditions, high motor velocity settings, or short and overtemperature conditions may trigger the open-load flag. In motor standstill, open load cannot be measured as the coils might eventually have zero current.

To safely detect an interrupted coil connection, operate in SpreadCycle and check the open-load flags following a motion of minimum four times the selected microstep resolution (= four electrical fullsteps) into a single direction using low or nominal motor velocity operation only. However, the *DRV_STATUS.ol* flag has just informative character and does not cause any action of the driver.

Undervoltage Lockout Protection

The TMC3241 features an UVLO protection for V_S, V_{CC_IO}, and the charge pump.

The UVLO condition on V_S is triggered below 4.15V.

The UVLO condition on V_{CC_IO} is triggered below 2.2V.

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

The occurrence of a V_S UVLO condition can be read from the register GSTAT as flag *vm_uvlo*. It is set following a brown out of the supply voltage. This flag is a write-clear flag. It must be actively written with 1 to clear it.

During a V_S or V_{CC_IO} UVLO, no communication with the IC is possible, all registers are reset, and the driver is disabled. The DO0 pin is active low (open-drain) to signal this condition to an external CPU.

ESD Protection

The chip has internal ESD protection on every pin.

The TMC3241 motor phase output pins are protected up to 8kV HBM in the application when using a bypass capacitor of at least 1µF on the positive voltage supply (V_S pins). This is not protection against the hot plugging of a motor.

Clock Oscillator and Clock Input

Internal PLL Block

The TMC3241 has an internal PLL frequency multiplier, allowing operation from either the internal trimmed oscillator or from an external clock source derived from a more precise oscillator. The external clock source can lie anywhere in the range of 1MHz to 32MHz in increments of 1MHz. A frequency divider with a user-defined division factor (*CLOCK_DIVIDER*) derives an internal reference frequency of 1MHz from the selected clock source. Based on this 1MHz clock, the PLL supplies a fixed 16MHz for the internal controller logic and calculations.

The PLL must be correctly configured and run before the operation of the IC. While the PLL is disabled or in a fault mode, the configuration registers can be accessed, but the operation of the driver and the full control logic block are stopped, and the driver is disabled. Any fail of the clock signal or a false PLL configuration puts the IC into this disabled mode and sets error flags *clk_is_stuck* and *clk_loss*.

The configuration of the PLL always requires two register write accesses, first setting and committing parameters, then a second access to clear the error flags. To allow start-up of the PLL, a delay between both the writes is required. The delay is either timed by a wait instruction (at minimum $t_{PLLSTRT}$) or by polling *commit* to be reset to 0. The second access following this delay is to reset the error flags *clk_is_stuck* and *clk_loss*, which in turn enables the control logic and driver blocks.

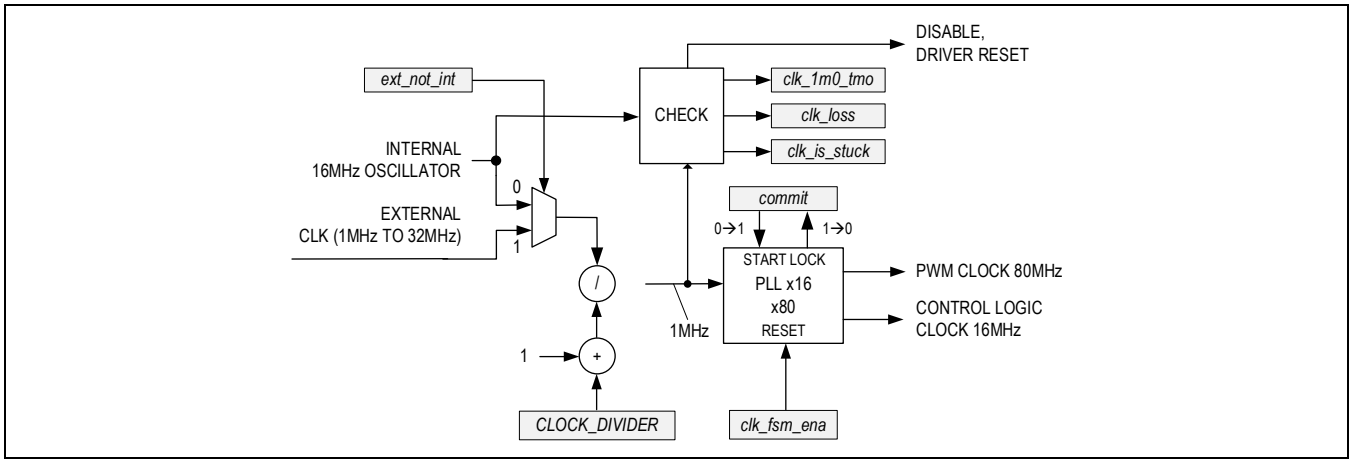


Figure 29. Block Diagram of PLL

Attention: Configure and start the PLL before operating the motor! The PLL error flags must be cleared to operate.

Example

Configure the PLL for operation with the internal 16MHz oscillator.

1. The 16MHz clock requires a divider of 1/16 to yield the 1MHz reference clock.
   Set *CLOCK_DIVIDER* = 15 (= 16 - 1). For internal clock, *ext_not_int* remains 0.
   clk_sys_sel and *clk_fsm_ena* must always be set to 1. Clear *clk_1M0_tmo* by setting this bit to 1.

   Write the PLL register with *commit* flag set:
   PLL = 0x01ED.
2. Poll the PLL register and wait for *commit* to be cleared again.

3. Now, the PLL is ready to operate. Release the digital part reset by clearing *clk_is_stuck* and *clk_loss*. To do so, set both the bits and write the same word into the PLL register, but keep *commit* cleared. PLL = 0x61EC
4. In case any PLL configuration error occurs, first disable the PLL to reset it (set *clk_fsm_ena* = 0) and restart with step 1.

Table 19. PLL Control Register PLL Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
commit	**Write:** This flag determines if the parameters (*ext_not_int, clk_sys_sel CLOCK_DIVIDER*) are transferred to the PLL block. When committing new values, *clk_loss* and *clk_is_stuck* are set to 1. To operate the IC: - Wait until the flag *commit* reads back as 0. - Write *commit* as 0 when clearing *clk_loss* and *clk_is_stuck*. **Read:** After committing changes to the PLL setting, the flag resets to 0 once the PLL is ready.	0	PLL is not touched – just clearing flags.
		1	Transfer setting to PLL and restart PLL. The flag automatically resets to 0 when PLL is ready.
ext_not_int	Selection of source for PLL clock signal. An external clock is recommended for higher precision of velocity and ramp timing, or whenever it is available. Usage of internal clock is beneficial if no precise local oscillator is available. The correct clock source must be selected before starting the PLL.	0	Select internal oscillator (16MHz). Set *CLOCK_DIVIDER* to 15.
		1	Select clock input as clock source.
clk_sys_sel	Set this bit to 1 for normal operation.	0	PLL clock not used, driver off.
		1	Select PLL as clock source.
clk_fsm_ena	Enable PLL. Set this bit to 1 only in combination with the correct clock source selected by *ext_not_int* and *CLOCK_DIVIDER*. Must be cycled to recover from the wrong clock frequency condition (*clk_1M0_tmo set*).	0	PLL block in reset.
		1	Normal operation of PLL.
CLOCK_DIVIDER	PLL clock divider. Set this register to match the frequency of the external or internal clock (frequency in MHz minus 1). Always set to 15 for internal oscillator. Do not operate with a wrong setting or inaccurate clock input, as a marginal setting can lead to the *clk_1M0_tmo* eventually being set!	0…	1MHz external clock…
		14	15MHz external clock
		15	Internal clock or 16MHz external clock
		16…	17MHz…
		31	32 MHz external clock
clk_1M0_tmo	Status bit. Write “1” to clear. Wrong frequency of derived 1MHz clock (this clock is compared to the internal clock) during the start-up phase of PLL. Not checked during operation. During operation, *clk_loss* checks for a drop out of the external clock. This error flag cannot be cleared while wrong values are in the PLL! Before clearing this flag: - Set *clk_fsm_ena* to 0 to disable the PLL. - Make sure the (external) clock and CLOCK_DIVIDER setting are correct, and - Commit values with *commit* and *clk_fsm_ena* set back to 1. Only then all error flags can be cleared.	0	PLL clock is present and roughly correct when compared to internal clock.
		1	External clock frequency is out of range or *CLOCK_DIVIDER* is wrong.
clk_loss	Status bit. Write “1” to clear (*commit* must be 0). Loss of external clock signal or wrong clock frequency. This bit also is set when any change is committed to the PLL configuration.	0	Normal operation
		1	PLL is in error state. Clock signal is lost.

clk_is_stuck	Status bit. Write “1” to clear (*commit* must be 0). External clock signal is stuck. Also is set in combination with *clk_loss*. This bit also is set when any change is committed to the PLL configuration.	0	Normal operation
		1	PLL is in error state. Clock signal has fixed polarity.

Using the Internal Clock

Tie the CLK input pin to GND or to V_{CC_IO} if the internal clock oscillator is to be used. Switching to an external clock requires actively setting the clock source using the PLL register.

The internal clock runs at a typical frequency of 16MHz and is sufficiently stable for PWM generation and chopper timing, as well as for many applications using a single axis, where no synchronized movement among multiple motors is required.

Using an External Clock

For exact timing of the motor motion using the internal motion controller, the use of a crystal stabilized clock is recommended. It either can be directly supplied from a crystal oscillator or distributed by a microcontroller to a number of drivers. Using a lower frequency to distribute a clock signal to multiple ICs is beneficial, as the distribution of a high frequency on a board requires a careful layout in combination with a line driver with impedance matched to the clock tree.

The external clock frequency can lie in the range of 1MHz to 32MHz and must be known for the parameterization of the internal PLL. Using the internal PLL, this frequency is converted to a 16MHz clock for ramp generation.

The required minimum and maximum duty cycle of the clock signal is defined in the **Electrical Characteristics** table. Especially at clock frequencies close to 32MHz, the clock's duty cycle requirements must be satisfied and edges must be clean.

In any case, make sure the clock source supplies full CMOS output logic levels and no reflections occur, leading to instable or multiple transitions of the logic threshold. Steep slopes are required when using a high clock frequency and reduce the risk of disturbance due to the coupling of adjacent traces.

Once the external clock is applied, the external clock input CLK actively must be selected as the clock source by register access to the PLL registers (set *ext_not_int*) and the PLL factor must be configured accordingly. In case the clock fails, the IC goes to an error mode and resets, and the step must be repeated. Check for errors by reading out the *reset* flag and the PLL register flags. Unlike other family members, an automatic fallback to the internal clock is not possible. Therefore, it is not recommended to stop the clock input while running the motor, as a motor going out of control can lead to high and uncontrolled feedback of power into the supply.

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

Quick Configuration Guides

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

PLL Initialization Guide

Before the operation of the control logic and driver blocks, the PLL must be configured and started.

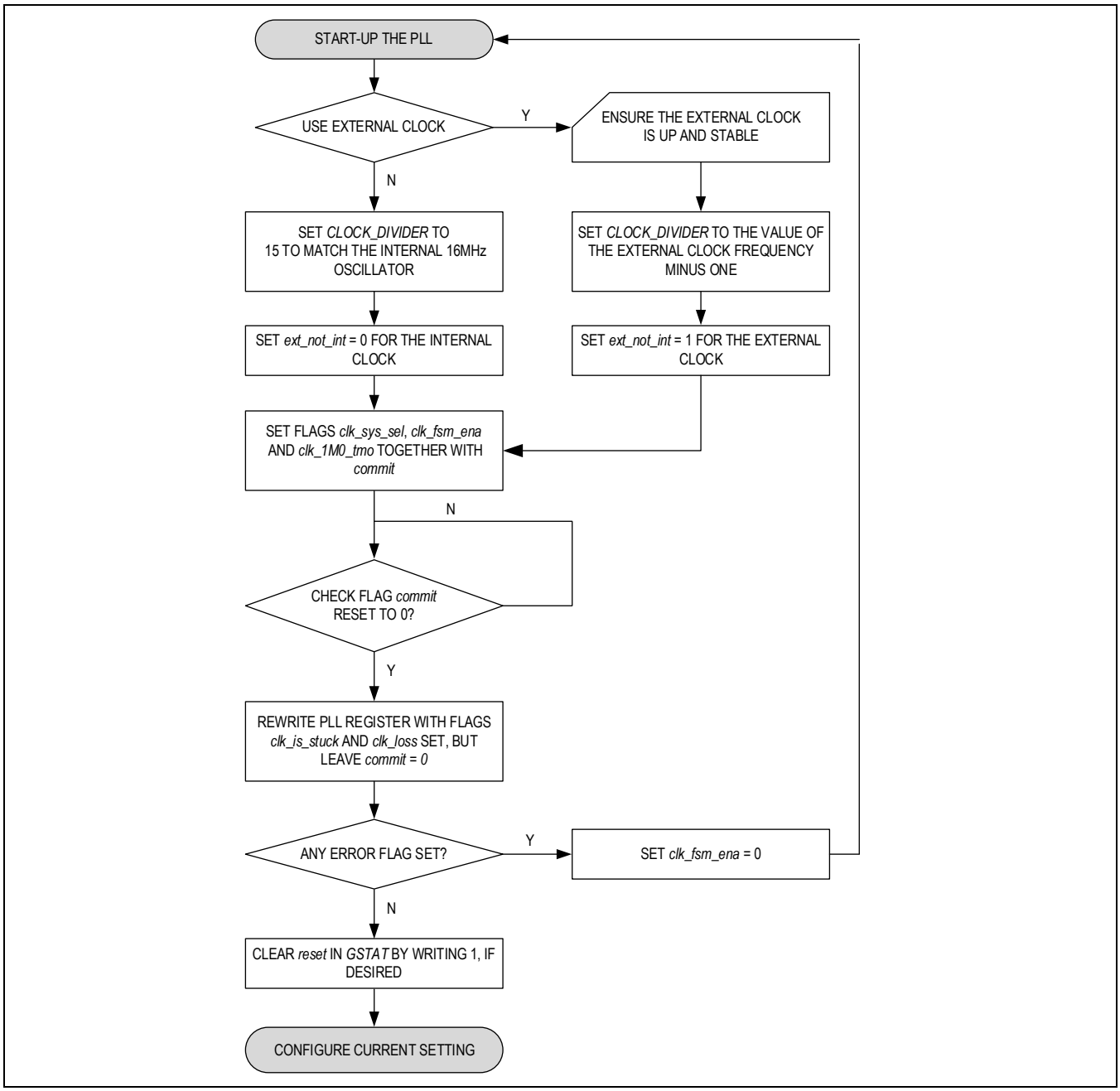


Figure 30. Quick Configuration Guide for Starting the PLL

Current Setting Guide

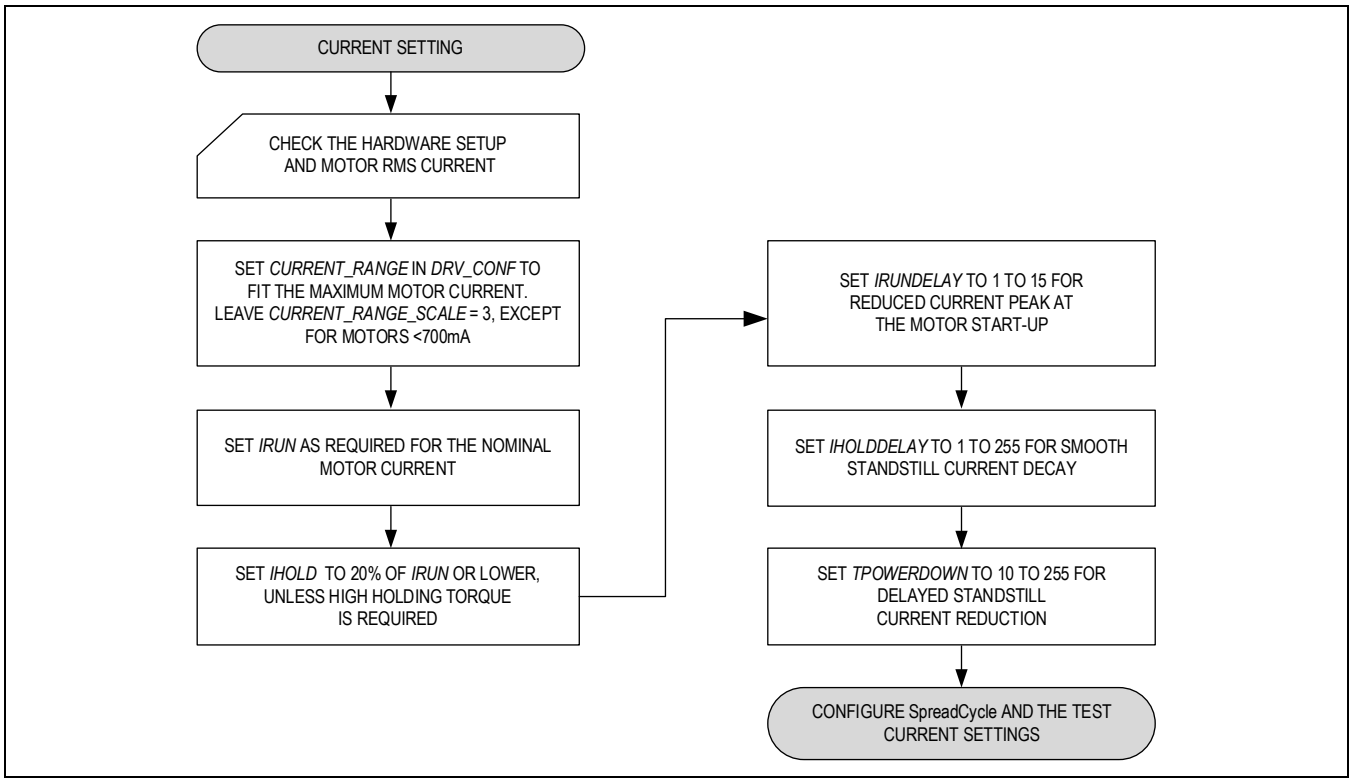


Figure 31. Quick Configuration Guide for Current Setting

SpreadCycle Configuration

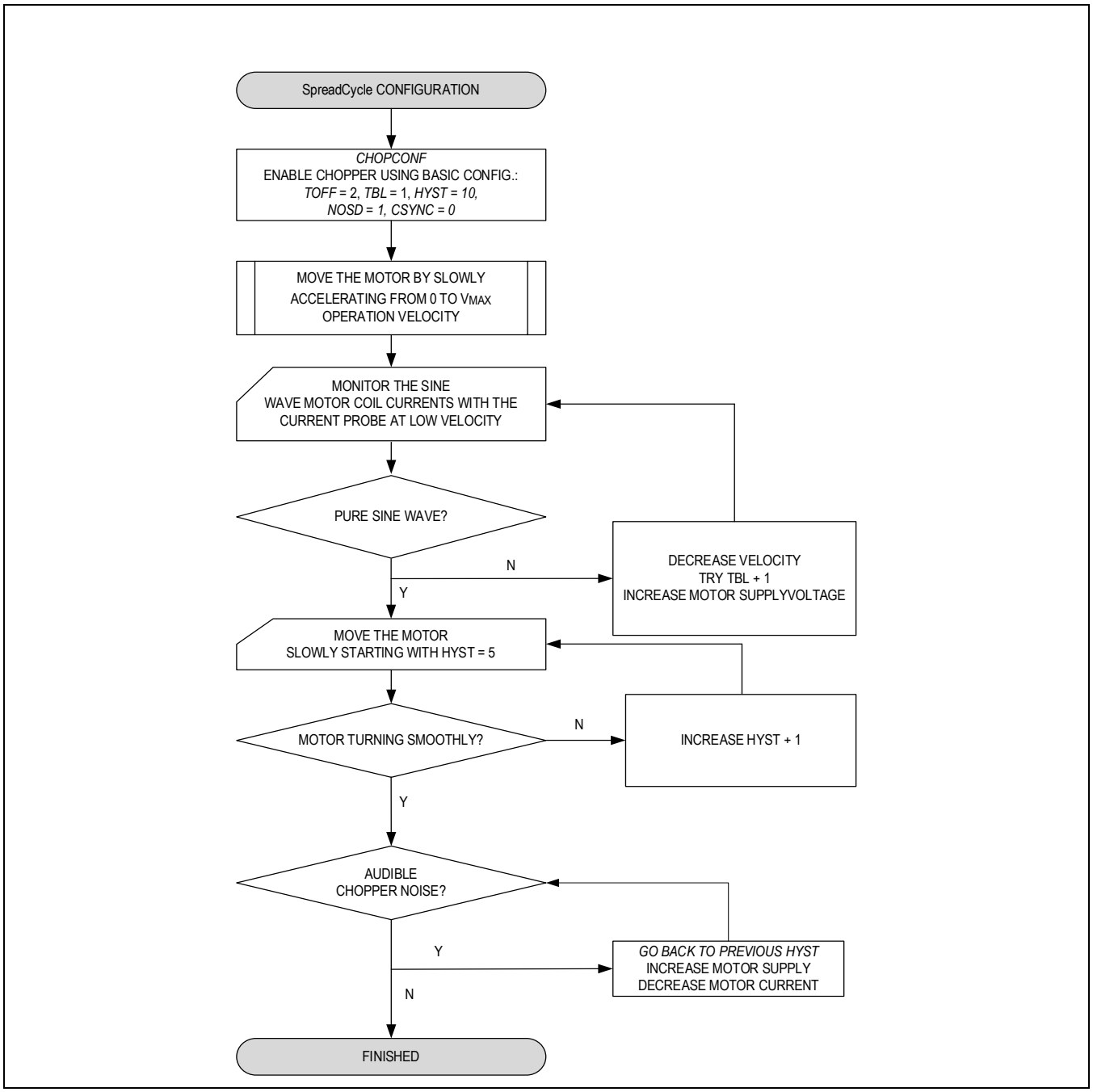


Figure 32. Quick Configuration Guide for SpreadCycle

CoolStep in Combination with SpreadCycle

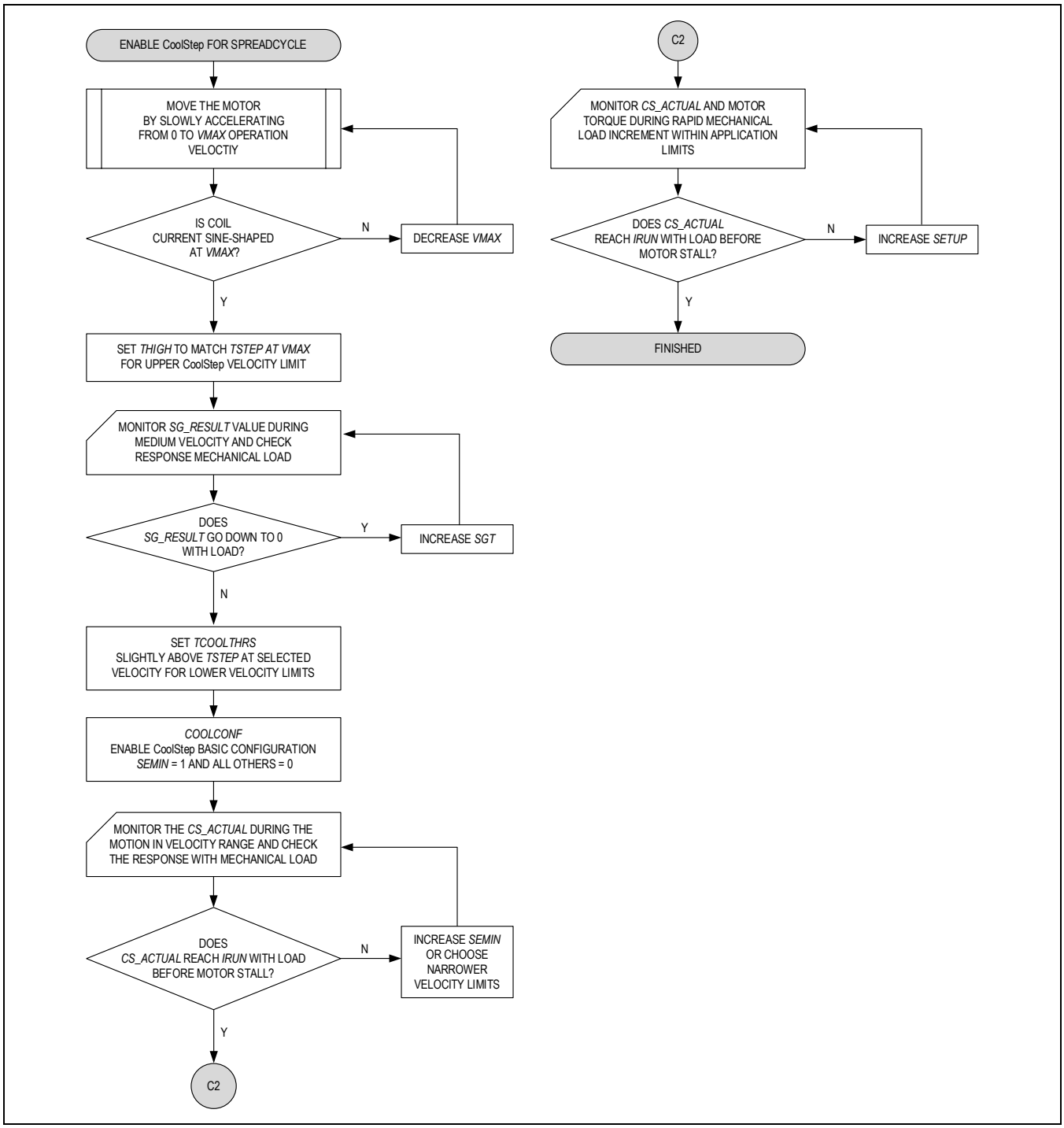


Figure 33. Quick Configuration Guide for CoolStep with SpreadCycle

Moving the Motor Using the Motion Controller

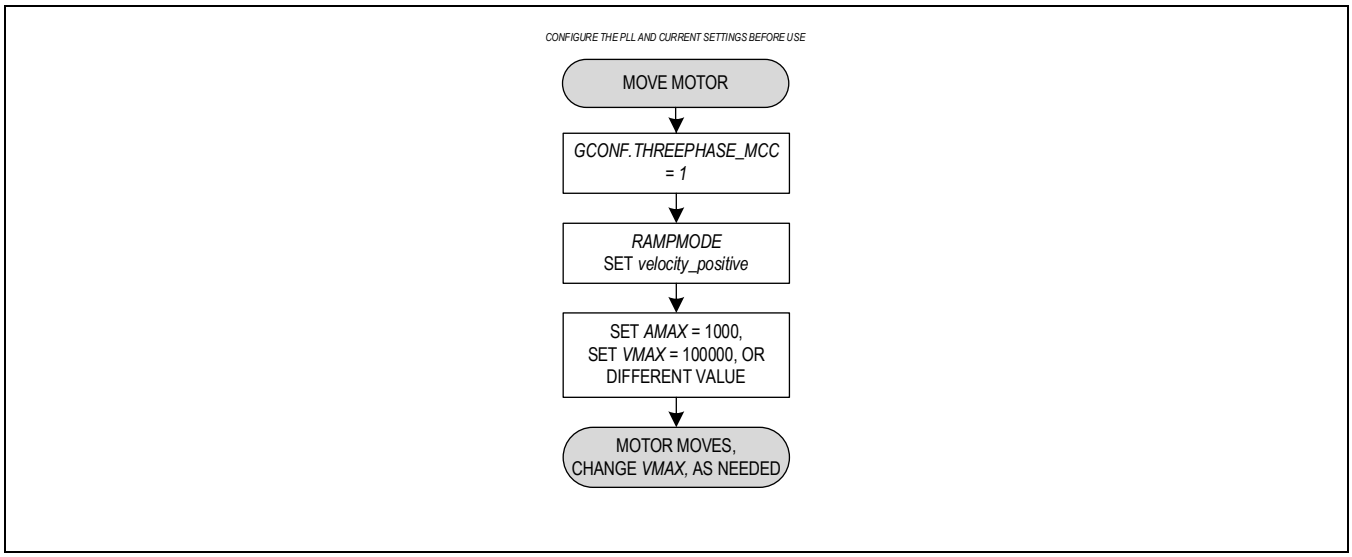


Figure 34. Quick Configuration Guide for Moving a Motor in Velocity Mode

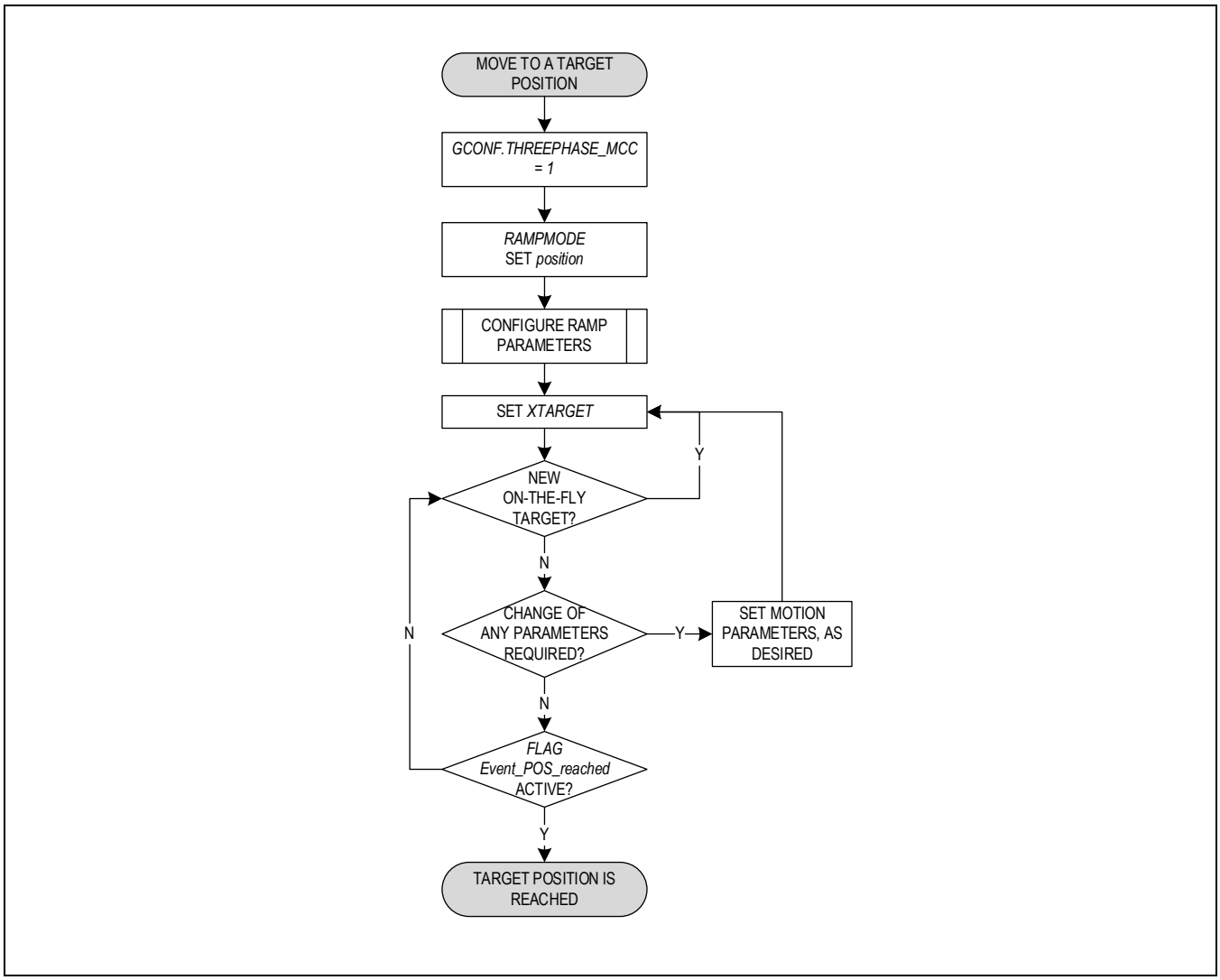


Figure 35. Quick Configuration Guide for Moving a Motor to a Target Position

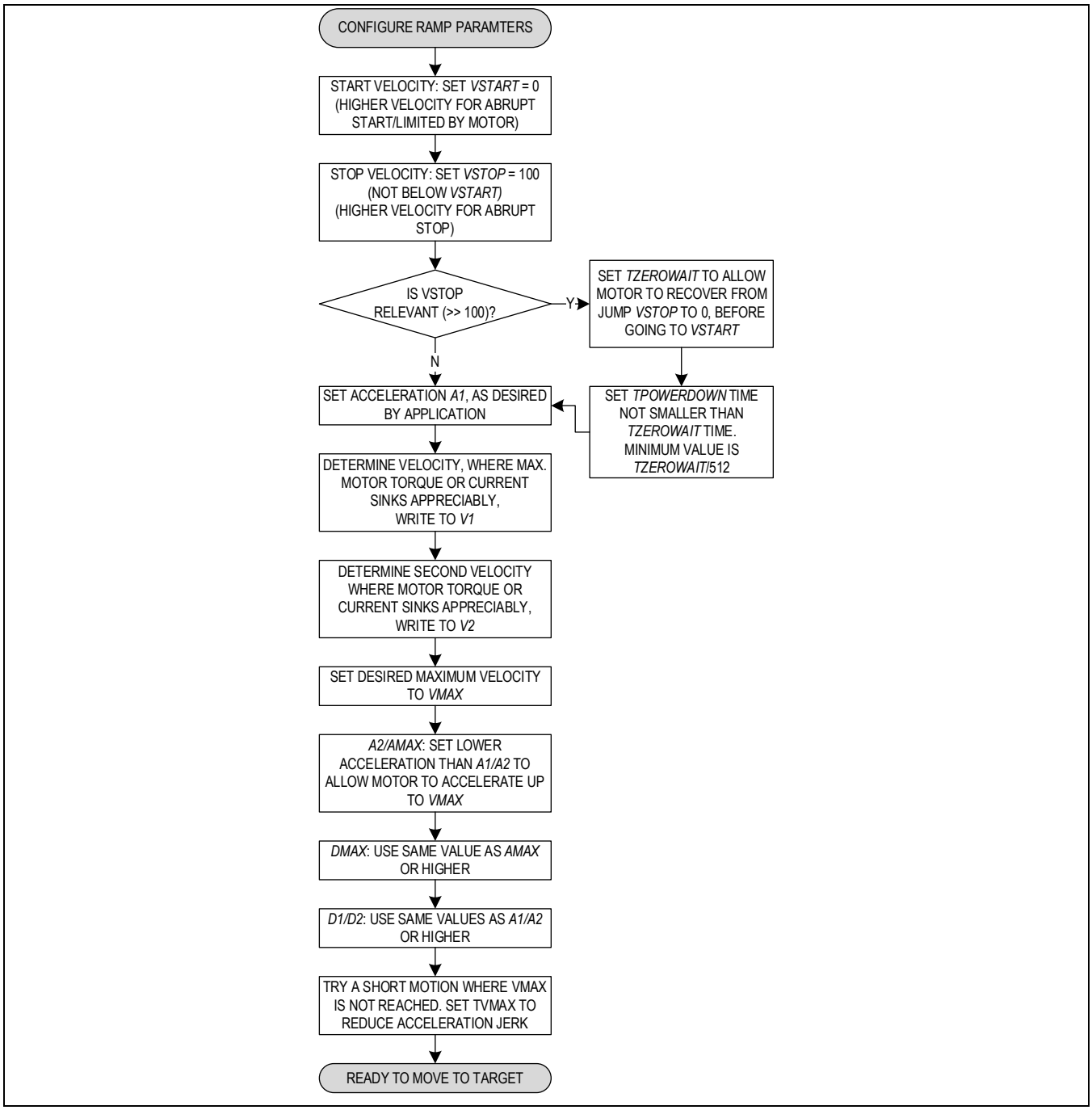


Figure 36. Quick Configuration Guide for Motion Ramp Parameter Settings

Initialization and Configuration Example

The following is a basic initialization example of the TMC3241 registers after power-up or reset. The write() and read() functions are only example function containers in this example. They must be replaced with the actual read/write functions for the specific microcontroller and SPI peripheral implementation used with the TMC3241.

```
//PLL Initialization
Write(PLL, 0x05FF);             //Configure for 16MHz external clock; commit set
Wait(1);                        //Wait 1ms
Write(PLL, 0x65FE);             //Re-Write with commit 0 and reset error flags
Read(PLL);                      //PLL-Register read to poll error flags


//Set conversion factors between pps and internal units
VEL_FACTOR = 0.953674316406     //fCLK/2 / 2^23 (fCLK=16MHz)
ACC_FACTOR = 116.415321827      //fCLK^2 / (512*256) / 2^24 (fCLK=16MHz)


//Most relevant configuration
Write(CHOPCONF, 0x000104A2);    //DEDGE=0, INTPOL=0, MRES=0, TBL=2, CHM=0,
                                // CSYNC=0, SG_SPEED=0, NOSD=1, HYST =10, TOFF=2
Write(DRV_CONF, 0x0000003f);    //full current range(4.24A rms),CURRENT_RANGE=3,
                                //CURRENT_RANGE_SCALE=3,SLOPE_CONTROL=3
Write(IHOLD_IRUN, 0x00077f01); //IRUN=127(50%),IHOLD=1,IHOLDDELAY=7,IRUNDELAY=0


//Ramp generator setup
Write(GCONF,0x40000004);        //Enable Motion Controller
Write(RAMPMODE, 0);             //Positioning Mode
Write(XTARGET, 0);              //Set XTARGET and XACTUAL to 0
Write(XACTUAL, 0);
Write(VSTART, 1/VEL_FACTOR);    //Set start velocity
Write(A1, 25600/ACC_FACTOR);    //Set acceleration between VSTART and V1
Write(V1, 25600/VEL_FACTOR);    //Set first acceleration phase threshold vel
Write(A2, 38400/ACC_FACTOR);    //Set acceleration between V1 and V2
Write(V2, 38400/VEL_FACTOR);    //Set second acceleration phase threshold vel
Write(AMAX, 51200/ACC_FACTOR); //Set acceleration between V2 and VMAX
Write(VMAX, 51200/VEL_FACTOR); //Set final VMAX
Write(DMAX, 51200/ACC_FACTOR); //Set deceleration between VMAX and V2
Write(D2, 38400/ACC_FACTOR);    //Set deceleration between V2 and V1
Write(D1, 25600/ACC_FACTOR);    //Set deceleration between V1 and VSTOP
Write(VSTOP, 10/VEL_FACTOR);    //Set stop velocity
Write(TCOOLTHRS, 1048575);      //Set velocity threshold for enabling coolstep
```

Design for Sustainability

Sustainable growth is one of the most important and urgent challenges today. Analog Devices contributes by designing highly efficient motor driver products to minimize energy consumption, ensuring best customer experience, and long-term satisfaction by smooth and silent run, while minimizing the demand for external resources, example, for power supply and cooling infrastructure, reducing motor size and magnet material use by intelligent control and advanced algorithms.

Register Map

GCR Register Overview

Shows all the related registers of the GCR block.

Legend:
R: Field can be read but not written.
RW: Field can be read and written.
W1C:Field can be read and cleared by writing a 1 to it.

ADDRESS AND NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x00 **GCONF**		threephase_mcc						
				OV_nN	length_steppulse			
		direct_mode	stop_enable	small_hysteresis	shaft	multistep_filt		fast_standstill
0x01 **GSTAT**								
			vccio_uv	vm_uvlo	register_reset	uv_cp	drv_err	reset
0x02 **DO_CONF**	do1_invPP	do1_nOD_PP	do0_invPP	do0_nOD_PP			do1_ev_n_deviation	do1_ev_pos_reached
	do1_ev_stop_sg	do1_ev_stop_ref		do1_ov	do1_xcomp	do1_dir	do1_step	do1_index
	do1_stall	do1_otpw	do1_error	do0_ev_n_deviation	do0_ev_pos_reached	do0_ev_stop_sg	do0_ev_stop_ref	
	do0_ov	do0_xcomp	do0_dir	do0_step	do0_index	do0_stall	do0_otpw	do0_error
0x03 **DO_SCOPE_CONF**								
				do1_scope_sel				
				do1_scope_en				do0_scope_sel[4]
	do0_scope_sel[3:0]							do0_scope_en
0x04 **IOIN**								
						SILICON_RV		
		ext_clk	ext_res_det					
	reserved		ENCN	DRV_ENN	ENCA	ENCB	REFR	REFL

ADDRESS AND NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x05 **X_COMPARE**	X_COMPARE[31:24]							
	X_COMPARE[23:16]							
	X_COMPARE[15:8]							
	X_COMPARE[7:0]							
0x06 **X_COMPARE_REPEAT**								
	X_COMPARE_REPEAT[23:16]							
	X_COMPARE_REPEAT[15:8]							
	X_COMPARE_REPEAT[7:0]							
0x0A **DRV_CONF**								
			SLOPE_CONTROL		CURRENT_RANGE_SCALE		CURRENT_RANGE	
0x0B **PLL**								
		clk_is_stuck	clk_loss	clk_1m0_tmo			CLOCK_DIVIDER[4:3]	
	CLOCK_DIVIDER[2:0]				clk_fsm_ena	clk_sys_sel	ext_not_int	commit
0x10 **IHOLD_IRUN**					IRUNDELAY			
	IHOLDDELAY							
	IRUN							
	IHOLD							
0x11 **TPOWERDOWN**								
	TPOWERDOWN							
0x12 **TSTEP**								
					TSTEP[19:16]			
	TSTEP[15:8]							
	TSTEP[7:0]							
0x14 **TCOOLTHRS**								
					TCOOLTHRS[19:16]			
	TCOOLTHRS[15:8]							
	TCOOLTHRS[7:0]							

ADDRESS AND NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x15 **THIGH**		
		THIGH[19:16]
	THIGH[15:8]	
	THIGH[7:0]	
0x1A **STEPS_LOST**		
		STEPS_LOST[19:16]
	STEPS_LOST[15:8]	
	STEPS_LOST[7:0]	
0x20 **RAMPMODE**		
		RAMPMODE
0x21 **XACTUAL**	XACTUAL[31:24]	
	XACTUAL[23:16]	
	XACTUAL[15:8]	
	XACTUAL[7:0]	
0x22 **VACTUAL**		
	VACTUAL[23:16]	
	VACTUAL[15:8]	
	VACTUAL[7:0]	
0x23 **VSTART**		
		VSTART[17:16]
	VSTART[15:8]	
	VSTART[7:0]	
0x24 **A1**		
		A1[17:16]
	A1[15:8]	
	A1[7:0]	
0x25 **V1**		
		V1[19:16]
	V1[15:8]	
	V1[7:0]	
0x26 **AMAX**		
		AMAX[17:16]
	AMAX[15:8]	
	AMAX[7:0]	

ADDRESS AND NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x27 **VMAX**		
	VMAX[22:16]	
	VMAX[15:8]	
	VMAX[7:0]	
0x28 **DMAX**		
	DMAX[17:16]	
	DMAX[15:8]	
	DMAX[7:0]	
0x29 **TVMAX**		
	TVMAX[15:8]	
	TVMAX[7:0]	
0x2A **D1**		
	D1[17:16]	
	D1[15:8]	
	D1[7:0]	
0x2B **VSTOP**		
	VSTOP[17:16]	
	VSTOP[15:8]	
	VSTOP[7:0]	
0x2C **TZEROWAIT**		
	TZEROWAIT[15:8]	
	TZEROWAIT[7:0]	
0x2D **XTARGET**	XTARGET[31:24]	
	XTARGET[23:16]	
	XTARGET[15:8]	
	XTARGET[7:0]	
0x2E **V2**		
	V2[19:16]	
	V2[15:8]	
	V2[7:0]	
0x2F **A2**		
	A2[17:16]	
	A2[15:8]	
	A2[7:0]	

ADDRESS AND NAME	FIELDS				MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)			
0x30 **D2**								
							D2[17:16]	
	D2[15:8]							
	D2[7:0]							
0x31 **AACTUAL**								
	AACTUAL[23:16]							
	AACTUAL[15:8]							
	AACTUAL[7:0]							
0x34 **SW_MODE**								
		virtual_ stop_enc	en_virtual_ stop_r	en_virtual_ stop_l	en_ softstop	sg_stop	en_latch_ encoder	latch_r_ inactive
	latch_r_ active	latch_l_ inactive	latch_l_ active	swap_lr	pol_stop_r	pol_stop_l	stop_r_ enable	stop_l_ enable
0x35 **RAMP_STAT**								
	status_ virtual_ stop_r	status_ virtual_ stop_l	status_sg	second_ move	t_ zerowait_ active	vzero	position_ reached	velocity_ reached
	event_ pos_ reached	event_ stop_sg	event_ stop_r	event_ stop_l	status_ latch_r	status_ latch_l	status_ stop_r	status_ stop_l
0x36 **XLATCH**	XLATCH[31:24]							
	XLATCH[23:16]							
	XLATCH[15:8]							
	XLATCH[7:0]							
0x38 **ENCMODE**								
						enc_sel_ decimal	latch_x_act	clr_enc_x
	pos_neg_edge		clr_once	clr_cont	ignore_AB	pol_N	pol_B	pol_A
0x39 **X_ENC**	X_ENC[31:24]							
	X_ENC[23:16]							
	X_ENC[15:8]							
	X_ENC[7:0]							

ADDRESS AND NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)							
0x3A **ENC_CONST**	ENC_CONST[31:24]								
	ENC_CONST[23:16]								
	ENC_CONST[15:8]								
	ENC_CONST[7:0]								
0x3B **ENC_STATUS**									
									n_event
0x3C **ENC_LATCH**	ENC_LATCH[31:24]								
	ENC_LATCH[23:16]								
	ENC_LATCH[15:8]								
	ENC_LATCH[7:0]								
0x3E **VIRTUAL_STOP_L**	VIRTUAL_STOP_L[31:24]								
	VIRTUAL_STOP_L[23:16]								
	VIRTUAL_STOP_L[15:8]								
	VIRTUAL_STOP_L[7:0]								
0x3F **VIRTUAL_STOP_R**	VIRTUAL_STOP_R[31:24]								
	VIRTUAL_STOP_R[23:16]								
	VIRTUAL_STOP_R[15:8]								
	VIRTUAL_STOP_R[7:0]								
0x58 **ADC_VSUPPLY_TEMP**									ADC_ TEMP[8]
	ADC_TEMP[7:0]								
									ADC_ VSUPPLY[8]
	ADC_VSUPPLY[7:0]								
0x5A **OTW_OV_VTH**									OVERTEM PPREWAR NING_ VTH[8]
	OVERTEMPPREWARNING_VTH[7:0]								
									OVERVOL TAGE_ VTH[8]
	OVERVOLTAGE_VTH[7:0]								

ADDRESS AND NAME	FIELDS	MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)		
0x60 **MSLUT_0**	MSLUT_0[31:24]			
	MSLUT_0[23:16]			
	MSLUT_0[15:8]			
	MSLUT_0[7:0]			
0x61 **MSLUT_1**	MSLUT_1[31:24]			
	MSLUT_1[23:16]			
	MSLUT_1[15:8]			
	MSLUT_1[7:0]			
0x62 **MSLUT_2**	MSLUT_2[31:24]			
	MSLUT_2[23:16]			
	MSLUT_2[15:8]			
	MSLUT_2[7:0]			
0x63 **MSLUT_3**	MSLUT_3[31:24]			
	MSLUT_3[23:16]			
	MSLUT_3[15:8]			
	MSLUT_3[7:0]			
0x64 **MSLUT_4**	MSLUT_4[31:24]			
	MSLUT_4[23:16]			
	MSLUT_4[15:8]			
	MSLUT_4[7:0]			
0x65 **MSLUT_5**	MSLUT_5[31:24]			
	MSLUT_5[23:16]			
	MSLUT_5[15:8]			
	MSLUT_5[7:0]			
0x66 **MSLUT_6**	MSLUT_6[31:24]			
	MSLUT_6[23:16]			
	MSLUT_6[15:8]			
	MSLUT_6[7:0]			
0x67 **MSLUT_7**	MSLUT_7[31:24]			
	MSLUT_7[23:16]			
	MSLUT_7[15:8]			
	MSLUT_7[7:0]			
0x68 **MSLUTSEL**	X3			
	X2			
	X1			
	W3	W2	W1	W0

ADDRESS AND NAME	FIELDS							MSB (TOP LEFT) TO LSB (BOTTOM RIGHT)
0x69 **MSLUTSTART**								
	START_SIN120							
	START_SIN							
0x6A **MSCNT**								
							MSCNT[9:8]	
	MSCNT[7:0]							
0x6B **MSCURACT**								CUR_V[8]
	CUR_V[7:0]							
								CUR_U[8]
	CUR_U[7:0]							
0x6C **CHOPCONF**			dedge	intpol	MRES			
								TBL[1]
	TBL[0]	chm		CSYNC	sg_speed	nosd	HYST[5:4]	
	HYST[3:0]				TOFF			
0x6D **COOLCONF**								sfilt
	thigh_sg_off	sgt						
	seimin	sedn			semax			
		seup			semin			
0x6F **DRV_STATUS**	stst		ol		s2g	otpw	ot	stallguard
	CS_ACTUAL							
				s2vs	ov	seq_stopped	SG_RESULT[9:8]	
	SG_RESULT[7:0]							

0x00: GCONF

Default: 0x00000014

Global Configuration Flags

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[30] **threephase_mcc**	RW 0x0	Enable use of internal ramp generator for three-phase driver.
	0: STEP_DIR 1: RAMP	Step/Dir driver Use internal ramp generator.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[12] **OV_nN**	RW 0x0	Enable use of ENCN as overvoltage output (high active open-drain).
	0: N_CHANNEL 1: OV_PIN	Use ENCN as N-channel of encoder input (default). Use ENCN as overvoltage output (high active open-drain). Hint: In this case, the N signal seen by the IC is always 0.
[11:8] **length_steppulse**	RW 0x0	Length of step pulse and dir to step setup time when using step pulse output signal.
	0: HALF 1: CLK_1 2: CLK_2 3: CLK_3 4: CLK_4 5: CLK_5 6: CLK_6 7: CLK_7 8: CLK_8 9: CLK_9 10: CLK_10 11: CLK_11 12: CLK_12 13: CLK_13 14: CLK_14 15: CLK_15	Double edge output signal: each step toggles the STEP output. Each steppulse has length of one clock cycle. Each steppulse has length of two clock cycles. Each steppulse has length of three clock cycles. Each steppulse has length of four clock cycles. Each steppulse has length of five clock cycles. Each steppulse has length of six clock cycles. Each steppulse has length of seven clock cycles. Each steppulse has length of eight clock cycles. Each steppulse has length of nine clock cycles. Each steppulse has length of ten clock cycles. Each steppulse has length of eleven clock cycles. Each steppulse has length of twelve clock cycles. Each steppulse has length of thirteen clock cycles. Each steppulse has length of fourteen clock cycles. Each steppulse has length of fifteen clock cycles
[6] **direct_mode**	RW 0x0	Enable direct motor phase current control using serial interface.
	0: NORMAL_MODE 1: DIRECT_MODE	Normal operation Motor coil currents and polarity directly programmed using serial interface. Register XTARGET (0x2D) specifies signed coil V current (bits 8...0) and coil U current (bits 24...16). In this mode, the current is scaled by the IHOLD setting.
[5] **stop_enable**	RW 0x0	Motor hard stop function enable.
	0: NONE 1: EM_STOP	Normal operation Emergency stop: ENCA stops the sequencer when tied high (no steps are executed by the sequencer, motor goes to standstill state).
[4] **small_hysteresis**	RW 0x1	Hysteresis control for TSTEP comparison of velocity thresholds.
	0: HYST_1_16 1: HYST_1_32	Hysteresis for step frequency comparison is 1/16 (higher tolerance to frequency jitter with external STEP and DIR source). Hysteresis for step frequency comparison is 1/32 (recommended with hardware-based ramp generator).
[3] **shaft**	RW 0x0	Change motor direction invert DIR signal (STEP and DIR mode only).
	0: DIR 1: DIR_INV	Default motor direction. Inverse motor direction (allowed for STEP and DIR mode only).
[2] **multistep_filt**	RW 0x1	Enable step input filtering.
	0: FILT_DIS 1: FILT_EN	Step input filtering disabled. Enable step input filtering optimization with external step source (default = 1).

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[0] **fast_standstill**	RW 0x0	Step pulse timeout for standstill detection.
	0: STST_2_20	Normal time: 2^20clocks (required if very low step frequencies are used, for example, with low microstep resolution).
	1: STST_2_18	Short time: 2^18 clocks (recommended for fast current reduction).

0x01: GSTAT

Default: 0x00000009
Global Status Flags
(Rewrite with '1' bit to clear respective flags.)

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[5] **vccio_uv**	RW, W1C 0x0	1: VCC_IO undervoltage occurred since last reset. Clear once after initial power-up.
	0: NORMAL_OPERATION	Normal operation
	1: VCCIO_UV_DET	V_{CC_IO} dropped below the lower threshold after last access to this register. Interface commands may have been lost.
[4] **vm_uvlo**	RW, W1C 0x0	1: V_M undervoltage occurred since last reset. Clear once after initial power-up.
	0: NORMAL_OPERATION	Normal operation
	1: VM_UVLO_DET	V_M had undervoltage, potential loss of motor position.
[3] **register_reset**	RW, W1C 0x1	Indicates a reset of the configuration registers.
	0: OPERATIONAL	Normal operation
	1: REG_RES_DET	Indicates the register map is reset. All registers are cleared to reset values.
[2] **uv_cp**	RW, W1C 0x0	Charge pump undervoltage condition flag. Clear once after power-up.
	0: OPERATIONAL	Normal operation
	1: CP_UV_DET	Indicates an undervoltage on the charge pump. The driver is disabled during undervoltage. This flag is latched for information.
[1] **drv_err**	RW, W1C 0x0	Driver error flag
	0: OPERATIONAL	Normal operation
	1: DRV_ERR_DET	Indicates the driver is shut down due to overtemperature or short-circuit detection. Read DRV_STATUS for details. The flag can only be cleared when the temperature is below the limit again.
[0] **reset**	RW, W1C 0x1	Reset flag
	0: NO_RESET	Normal operation
	1: RESET_DET	Indicates the IC is reset since the last clear of this bit.

0x02: DO_CONF

Default: 0xA0000000

DO0 and DO1 output pin configuration.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31] **do1_invPP**	RW 0x1	Select polarity for DO1 in push-pull mode.
	0: HIGH_ACTIVE 1: LOW_ACTIVE	High active output Low active output
[30] **do1_nOD_PP**	RW 0x0	DO1 output type configuration.
	0: OPEN_DRAIN 1: PUSH_PULL	DO1 is open-drain output (active low). Enable DO1 push-pull output.
[29] **do0_invPP**	RW 0x1	Select polarity for DO0 in push-pull mode.
	0: HIGH_ACTIVE 1: LOW_ACTIVE	High active output Low active output
[28] **do0_nOD_PP**	RW 0x0	DO0 output type configuration.
	0: OPEN_DRAIN 1: PUSH_PULL	DO0 is open-drain output (active low). Enable DO0 push-pull output.
[25] **do1_ev_n_deviation**	RW 0x0	Map N event and deviation warning to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[24] **do1_ev_pos_reached**	RW 0x0	Map RAMPSTAT_EVENT_POS_REACHED to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[23] **do1_ev_stop_sg**	RW 0x0	Map RAMPSTAT_EVENT_SG_STOP to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[22] **do1_ev_stop_ref**	RW 0x0	Map RAMPSTAT_EVENT_STOPL || RAMPSTAT_EVENT_STOPR to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[20] **do1_ov**	RW 0x0	Map detection of overvoltage to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[19] **do1_xcomp**	RW 0x0	Map position comparator to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[18] **do1_dir**	RW 0x0	Map DIR output signal to DO1.
	0: DISABLED 1: ENABLED	Disable Enable

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17] **do1_step**	RW 0x0	Map STEP output signal to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[16] **do1_index**	RW 0x0	Map index signal to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[15] **do1_stall**	RW 0x0	Map StallGuard signal SG to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[14] **do1_otpw**	RW 0x0	Map OTPW (overtemperature prewarning) to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[13] **do1_error**	RW 0x0	Map driver error condition to DO1.
	0: DISABLED 1: ENABLED	Disable Enable
[12] **do0_ev_n_deviation**	RW 0x0	Map N event and deviation warning to DO0.
	0: DISABLED 1: ENABLED	Disable Enable
[11] **do0_ev_pos_reached**	RW 0x0	Map RAMPSTAT_EVENT_POS_REACHED to DO0.
	0: DISABLED 1: ENABLED	Disable Enable
[10] **do0_ev_stop_sg**	RW 0x0	Map RAMPSTAT_EVENT_SG_STOP to DO0.
	0: DISABLED 1: ENABLED	Disable Enable
[9] **do0_ev_stop_ref**	RW 0x0	Map RAMPSTAT_EVENT_STOPL || RAMPSTAT_EVENT_STOPR to DO0.
	0: DISABLED 1: ENABLED	Disable Enable
[7] **do0_ov**	RW 0x0	Map detection of overvoltage to DO0.
	0: DISABLED 1: ENABLED	Disable Enable
[6] **do0_xcomp**	RW 0x0	Map position comparator to DO0.
	0: DISABLED 1: ENABLED	Disable Enable
[5] **do0_dir**	RW 0x0	Map DIR output signal to DO0.
	0: DISABLED 1: ENABLED	Disable Enable

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[4] **do0_step**	RW 0x0	Map STEP output signal to DO0.
	0: DISABLED Disable 1: ENABLED Enable	
[3] **do0_index**	RW 0x0	Map index signal to DO0.
	0: DISABLED Disable 1: ENABLED Enable	
[2] **do0_stall**	RW 0x0	Map StallGuard signal SG to DO0.
	0: DISABLED Disable 1: ENABLED Enable	
[1] **do0_otpw**	RW 0x0	Map OTPW (overtemperature prewarning) to DO0.
	0: DISABLED Disable 1: ENABLED Enable	
[0] **do0_error**	RW 0x0	Map driver error condition to DO0. DO0 always shows the reset-status. It is active low during reset condition.
	0: DISABLED Disable 1: ENABLED Enable	

0x03: DO_SCOPE_CONF

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[20:16] **do1_scope_sel**	RW 0x00	Selection of DO1 RT-OSCI source.
	0: RESERVED_0 — Reserved 1: RESERVED_1 — Reserved 2: RESERVED_2 — Reserved 3: RESERVED_3 — Reserved 4: RESERVED_4 — Reserved 5: RESERVED_5 — Reserved 6: RESERVED_6 — Reserved 7: ADC_TEMP — ADC_TEMPERATURE 8: RESERVED_8 — Reserved 9: RESERVED_9 — Reserved 10: RESERVED_10 — Reserved 11: RESERVED_11 — Reserved 12: RESERVED_12 — Reserved 13: CS_ACTUAL — CS_ACTUAL 14: RESERVED_14 — Reserved 15: SG_RESULT — SG_RESULT 16: RESERVED_16 — Reserved 17: CUR_U — CUR_U 18: CUR_V — CUR_V 19: DAC_X — Reserved 20: RESERVED_20 — Reserved 21: RESERVED_21 — Reserved 22: MSCNT — MSCNT 23: RESERVED_23 — Reserved 24: RESERVED_24 — Reserved 25: TSTEP_V_20_13 — TSTEP_VELOCITY[20:13] 26: TSTEP_V_18_11 — TSTEP_VELOCITY[18:11] 27: TSTEP_V_16_9 — TSTEP_VELOCITY[16:9] 28: USER — USER_VALUE (=DO_CONF[20:13])	
[12] **do1_scope_en**	RW 0x0	Enable RT-OSCI on DO1 pin.
	0: DISABLED — RT-OSCI disabled. 1: ENABLED — RT-OI enabled.	

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[8:4] **do0_scope_sel**	RW 0x00	Selection of D0 RT-OSCI source
	0: RESERVED_0 Reserved 1: RESERVED_1 Reserved 2: RESERVED_2 Reserved 3: RESERVED_3 Reserved 4: RESERVED_4 Reserved 5: RESERVED_5 Reserved 6: RESERVED_6 Reserved 7: ADC_TEMP ADC_TEMPERATURE 8: RESERVED_8 Reserved 9: RESERVED_9 Reserved 10: RESERVED_10 Reserved 11: RESERVED_11 Reserved 12: RESERVED_12 Reserved 13: CS_ACTUAL CS_ACTUAL 14: RESERVED_14 Reserved 15: SG_RESULT SG_RESULT 16: RESERVED_16 Reserved 17: CUR_U CUR_U 18: CUR_V CUR_V 19: DAC_X Reserved 20: RESERVED_20 Reserved 21: RESERVED_21 Reserved 22: MSCNT MSCNT 23: RESERVED_23 Reserved 24: RESERVED_24 Reserved 25: TSTEP_V_20_13 TSTEP_VELOCITY[20:13] 26: TSTEP_V_18_11 TSTEP_VELOCITY[18:11] 27: TSTEP_V_16_9 TSTEP_VELOCITY[16:9] 28: USER USER_VALUE (=DO_CONF[7:0])	
[0] **do0_scope_en**	RW 0x0	Enable RT-OSCI on DO0 pin.
	0: DISABLED RT-OSCI disabled. 1: ENABLED RT-OSCI enabled.	

Note: TSTEP_VELOCITY: absolute velocity (23-bit) derived from STEP input.

0x04: IOIN

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[18:16] **SILICON_RV**	R, unsigned 0x0	Silicon revision number
[14] **ext_clk**	R 0x0	Detection of valid external clock CLK.
	0: INT_OSC The internal oscillator is used for generating the system clock. 1: EXT_OSC An external clock signal is used for generating the system clock.	
[13] **ext_res_det**	R 0x0	Check of external reference resistor.
	0: REF_RES_FAULT Resistor is missing, driver cannot operate. 1: REF_RES_DET Normal operation: External resistor between REF and GND.	

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[7] **reserved**	R 0x0	
[5] **ENCN**	R 0x0	State of pin ENCN.
	0: LOW 1: HIGH	Low logic level High logic level
[4] **DRV_ENN**	R 0x0	Driver disabled/enabled state.
	0: LOW 1: HIGH	Enable Disable
[3] **ENCA**	R 0x0	State of pin ENCA.
	0: LOW 1: HIGH	Low logic level High logic level
[2] **ENCB**	R 0x0	State of pin ENCB.
	0: LOW 1: HIGH	Low logic level High logic level
[1] **REFR**	R 0x0	REFR pin state (DIR pin in STEP&DIR).
[0] **REFL**	R 0x0	REFL pin state (STEP pin in STEP&DIR).

0x05: X_COMPARE

Default: 0xFFFFFFFF

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **X_COMPARE**	RW, signed 0xFFFFFFFF	Position comparison register for motion controller position strobe. X_COMPARE is an absolute position. The position pulse is available on output DO0 or DO1. When XACTUAL = X_COMPARE , the output signal PP (position pulse) is high. It returns to a low state if the positions do not match. If X_COMPARE_REPEAT is > 1, X_COMPARE is the position reference for the periodic position strobe trigger output.

0x06: X_COMPARE_REPEAT

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[23:0] **X_COMPARE_REPEAT**	RW 0x000000	This register defines a relative distance in microsteps (based on MRES configuration). If set to >1, the position compare pulse is raised every time a multiple of X_COMPARE_REPEAT µsteps are made. X_COMPARE defines the position of the first pulse.
	0: 0	Repetition mechanism off.

0x0A: DRV_CONF

Default: 0x0000003C

Driver current range and slope settings.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[5:4] **SLOPE_CONTROL**	RW 0x3	Slope control setting.
	0: SC_100V_US 100V/µs 1: SC_200V_US 200V/µs 2: SC_400V_US 400V/µs 3: SC_800V_US 800V/µs	
[3:2] **CURRENT_RANGE_SCALE**	RW 0x3	Allows scaling down the current range to support low motors with a full scale rating below 750mA RMS . Use only with CURRENT_RANGE = 0 setting.
	0: SCALE_025 25% 1: SCALE_050 50% 2: SCALE_075 75% 3: SCALE_100 100%/normal setting	
[1:0] **CURRENT_RANGE**	RW 0x0	This setting allows a basic adaptation of the drivers RDS(ON) current sensing to the motor current range. Select the lowest fitting range for best current precision. The value is the peak current setting.
	0: RMS_1A 1ARMS 1: RMS_2A 2ARMS 2: RMS_3A 3ARMS 3: RMS_4A 4ARMS	

0x0B: PLL

Default: 0x00000000

PLL configuration and operation.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[14] **clk_is_stuck**	RW, W1C 0x0	Missing external clock signal. Also is set together with CLK loss and whenever new values are committed to the PLL.
	0: CLK_OK Clock OK 1: CLK_NONE Clock missing	

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[13] **clk_loss**	RW, W1C 0x0	Clock fault detection for external clock. Is set when the clock frequency or divider is out of range. Also is set whenever new values are committed to the PLL.
	0: CLK_OK 1: CLK_LOSS	Clock OK Clock is faulty or lost.
[12] **clk_1m0_tmo**	RW, W1C 0x0	During start-up of the PLL, it checks that the divided clock frequency is in range by comparing it to the internal oscillator. When out of range, this flag is set. Before reset, disable clk_fsm_ena and re-enable while setting the correct parameters.
	0: CLK_OK 1: TIMEOUT	Clock OK Clock timeout
[9:5] **CLOCK_DIVIDER**	RW 0x00	Clock divider value (set to have Input_clock/(CLOCK_DIVIDER + 1) = 1MHz).
	0: CLK_1M 1: CLK_2M 2: CLK_3M 3: CLK_4M 4: CLK_5M 5: CLK_6M 6: CLK_7M 7: CLK_8M 8: CLK_9M 9: CLK_10M 10: CLK_11M 11: CLK_12M 12: CLK_13M 13: CLK_14M 14: CLK_15M 15: CLK_16M_INT 16: CLK_17M 17: CLK_18M 18: CLK_19M 19: CLK_20M 20: CLK_21M 21: CLK_22M 22: CLK_23M 23: CLK_24M 24: CLK_25M 25: CLK_26M 26: CLK_27M 27: CLK_28M 28: CLK_29M 29: CLK_30M 30: CLK_31M 31: CLK_32M	1MHz (pin CLK) 2MHz (pin CLK) 3MHz (pin CLK) 4MHz (pin CLK) 5MHz (pin CLK) 6MHz (pin CLK) 7MHz (pin CLK) 8MHz (pin CLK) 9MHz (pin CLK) 10MHz (pin CLK) 11MHz (pin CLK) 12MHz (pin CLK) 13MHz (pin CLK) 14MHz (pin CLK) 15MHz (pin CLK) 16MHz (pin CLK) or internal oscillator 17MHz (pin CLK) 18MHz (pin CLK) 19MHz (pin CLK) 20MHz (pin CLK) 21MHz (pin CLK) 22MHz (pin CLK) 23MHz (pin CLK) 24MHz (pin CLK) 25MHz (pin CLK) 26MHz (pin CLK) 27MHz (pin CLK) 28MHz (pin CLK) 29MHz (pin CLK) 30MHz (pin CLK) 31MHz (pin CLK) 32MHz (pin CLK)
[3] **clk_fsm_ena**	RW 0x0	Enable PLL FSM.
	0: DISABLE 1: ENABLE	PLL disabled, chip goes to reset but registers bank is kept. Chip in operation.
[2] **clk_sys_sel**	RW 0x0	Primary clock selector
	0: INTERNAL 1: PLL	Internal 16MHz clock with PLL disabled. The chip remains in reset during this time. PLL clock

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[1] **ext_not_int**	RW 0x0	
	0: INT_OSC 1: EXT_OSC	Select internal clock oscillator. Select external clock signal applied to pin CLK.
[0] **commit**	RW 0x0	Commit changes to PLL logic (self-clear bit). This bit clears when the PLL is stabilized. Wait until the bit is cleared before resetting error flags clk_loss and clk_is_stuck.
	0: NO_OP 1: UPDATE_SETTINGS	Write 0 to allow resetting clk_loss and clk_is_stuck, and release the driver from the reset state. Set to 1 after new valid values are programmed to ext_not_int, clk_sys_sel, and CLOCK_DIVIDER. The chip goes to a reset state (except for register space) during this time.

0x10: IHOLD_IRUN

Default: 0x0407F040

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[27:24] **IRUNDELAY**	RW, unsigned 0x4	Controls the number of clock cycles for motor power-up after start is detected. Set a higher value to reduce inrush current at motor start. Delay per current increment step in multiples of IRUNDELAY × 64 clocks.
[23:16] **IHOLDDELAY**	RW 0x07	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: 0	Instant power down.
[15:8] **IRUN**	RW, unsigned 0xF0	Motor run current for SpreadCycle (0 = 1/256… 255 = 256/256). Info: Choose CURRENT_RANGE in a way that normal IRUN is 128 to 255 for best microstep performance.
[7:0] **IHOLD**	RW, unsigned 0x40	Standstill current. (0 = 1/256… 255 = 256/256)

0x11: TPOWERDOWN

Default: 0x0000000A

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[7:0] **TPOWERDOWN**	RW, unsigned 0x0A	TPOWERDOWN sets the delay time after standstill (stst) of the motor to motor current power down. Time range is about 0 seconds to 4 seconds. A certain minimum value allows the remeasurement of motor coil resistance in applications where major heat-up is expected. Reset default = 10 0…((2^(8)) - 1) × 2^(18) tCLK

0x12: TSTEP

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[19:0] **TSTEP**	R, unsigned 0x00000	Actual measured time between two 1/256 microsteps derived from the step input frequency in units of 1/fCLK . Measured value is (2^(20)) - 1 in case of overflow or standstill. All TSTEP related thresholds use a hysteresis of 1/16 of the compare value to compensate for jitter in the clock or step frequency. The flag small_hysteresis modifies the hysteresis to a smaller value of 1/32. (Txxx × 15/16) – 1, or (Txxx × 31/32) - 1 is used as a second compare value for each comparison value. This means the lower switching velocity equals the calculated setting but the upper switching velocity is higher, as defined by the hysteresis setting. When working with the motion controller, the measured TSTEP for a given velocity V is in the range (2^(24)/V) ≤ TSTEP ≤ 2^(24)/(V - 1).

0x14: TCOOLTHRS

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[19:0] **TCOOLTHRS**	RW, unsigned 0x00000	Lower threshold velocity for switching on CoolStep and StallGuard (unsigned). Set this parameter to disable CoolStep at low speeds, where it cannot work reliably. The stop-on-stall function (enable with sg_stop when using internal motion controller) and the stall output signal are enabled upon exceeding this velocity. It is disabled again once the velocity falls below this threshold. TCOOLTHRS ≥ TSTEP ≥ THIGH: CoolStep is enabled, if configured. TCOOLTHRS ≥ TSTEP: Stop-on-stall is enabled, if configured. Stall output signal (D0/1) is enabled, if configured.

0x15: THIGH

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[19:0] **THIGH**	RW, unsigned 0x00000	Upper velocity limit for CoolStep. TSTEP < THIGH disables CoolStep.

0x1A: STEPS_LOST

Default: 0x00000000

Lost steps counter for Step&Dir operation.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[19:0] **STEPS_LOST**	R, signed 0x00000	Only in STEP and DIR mode: with stop-on-stall enabled, counts the number of step pulses missed since stall or after raising ENC_A stop.

0x20: RAMPMODE

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[1:0] **RAMPMODE**	RW 0x0	Motion controller ramping mode.
	0: POS_MODE	Positioning mode (using all A, D, and V parameters).
	1: VEL_POS	Velocity mode to positive VMAX (using A1, A2, AMAX, V1, V2, VMAX).
	2: VEL_NEG	Velocity mode to negative VMAX (using A1,A2, AMAX, V1, V2, VMAX).
	3: HOLD	Hold mode (velocity remains unchanged, unless stop event occurs).

0x21: XACTUAL

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **XACTUAL**	RW, signed 0x00000000	Actual motor position (signed). Note: This value normally should only be modified when homing the drive. In positioning mode, modifying the register content starts a motion.

0x22: VACTUAL

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[23:0] **VACTUAL**	R, signed 0x000000	Actual motor velocity from ramp generator (signed). The sign matches the motion direction. A negative sign means motion to lower XACTUAL. ±(2^(23)) - 1 [µsteps/t]

0x23: VSTART

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **VSTART**	RW, unsigned 0x00000	Motor start velocity (unsigned). For universal use, set VSTOP ≥ VSTART. This is not required if the motion distance is sufficient to ensure deceleration from VSTART to VSTOP. 0…(2^(18)) - 1 [µsteps/t]

0x24: A1

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **A1**	RW, unsigned 0x00000	First acceleration between VSTART and V1 (unsigned). 0…(2^(18)) -1 [µsteps/ta^(2)]

0x25: V1

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[19:0] **V1**	RW, unsigned 0x00000	First acceleration/deceleration phase threshold velocity (unsigned). 0: Disables A1 and D1 phase, use AMAX, DMAX only. 0…(2^(20)) - 1 [µsteps/t]

0x26: AMAX

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **AMAX**	RW, unsigned 0x00000	Second acceleration between V1 and VMAX (unsigned) This is the acceleration and deceleration value for velocity mode. 0…(2^(18)) - 1 [µsteps/ta^(2)]

0x27: VMAX

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[22:0] **VMAX**	RW, unsigned 0x000000	Motion ramp target velocity (for positioning ensure VMAX ≥ VSTART) (unsigned). This is the target velocity in velocity mode. It can be changed any time during a motion. 0…(2^(23)) - 512 [µsteps/t]

0x28: DMAX

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **DMAX**	RW, unsigned 0x00000	Deceleration between VMAX and V1 (unsigned). 0…(2^(18)) - 1 [µsteps/ta^(2)]

0x29: TVMAX

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[15:0] **TVMAX**	RW, unsigned 0x0000	Minimum time for constant velocity segments in multiple of 512 clocks. 0: Disables minimum duration setting for constant velocity phase. >0: A minimum duration of constant velocity is inserted in between of any change from acceleration to deceleration or vice versa to reduce jerk. 0…(2^(16)) - 1 × 512 tCLK Note: Configure this register after setting VMAX when in position mode and standstill. Set TVMAX = 0 during velocity mode to avoid triggering the TVMAX delay when switching back to ramp mode.

0x2A: D1

Default: 0x0000000A

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **D1**	RW, unsigned 0x0000A	Deceleration between V1 and VSTOP (unsigned). Attention: Do not set 0 in positioning mode, even if V1 = 0! 1…(2^(16)) - 1 [µsteps/ta^(2)] Reset default = 10

0x2B: VSTOP

Default: 0x0000000A

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **VSTOP**	RW, unsigned 0x0000A	Motor stop velocity (unsigned). Hint: Set VSTOP ≥ VSTART to allow positioning for short distances. Attention: Do not set 0 in positioning mode, minimum 10 recommend! 1…(2^(18)) - 1 [µsteps/t] Reset default = 10

0x2C: TZEROWAIT

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[15:0] **TZEROWAIT**	RW, unsigned 0x0000	Defines the waiting time after ramping down to zero velocity before next movement or direction inversion can start. Time range is about 0 seconds to 2 seconds. This setting avoids excess acceleration, for example, from VSTOP to -VSTART. 0…(2^(16)) - 1 × 512 tCLK

0x2D: XTARGET

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **XTARGET**	RW, signed 0x00000000	Target position for ramp mode (signed). Write a new target position to this register to activate the ramp generator positioning in RAMPMODE = 0. Initialize all velocity, acceleration, and deceleration parameters before. Hint: The position is allowed to wrap around, thus, XTARGET value optionally can be treated as an unsigned number. Note: The maximum possible displacement is ±((2^(31)) - 1). Hint: When increasing V1, D1, or DMAX during a motion, rewrite XTARGET afterwards to trigger a second acceleration phase, if desired. -2^(31)…+(2^(31)) – 1 When GCONF.DIRECT_MODE is set, specifies signed coil V current (bits 8...0) and coil U current (bits 24...16).

0x2E: V2

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[19:0] **V2**	RW, unsigned 0x00000	Velocity difference from VMAX for activation of acceleration segments with AMAX/2 and DMAX/2. 0: Disables AMAX/2 and DMAX/2 phase, use AMAX, DMAX only. 0…(2^(20)) - 1 [µsteps/t]

0x2F: A2

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **A2**	RW, unsigned 0x00000	Acceleration between V1 and V2 (unsigned). 0…(2^(18)) - 1 [µsteps/ta^(2)]

0x30: D2

Default: 0x0000000A

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[17:0] **D2**	RW, unsigned 0x0000A	Deceleration between V2 and V1 (unsigned). Attention: Do not set 0 in positioning mode, even if V2 = 0! 1…(2^(18)) - 1 [µsteps/ta^(2)] Reset default = 10

0x31: AACTUAL

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[23:0] **AACTUAL**	R, signed 0x000000	Current acceleration used by the ramp generator.

0x34: SW_MODE

Default: 0x00000000
Switch mode configuration.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[14] **virtual_stop_enc**	RW 0x0	Source for virtual stop (VIRTUAL_STOP_L and VIRTUAL_STOP_R).
	0: SRC_XACTUAL 1: SRC_X_ENC	Virtual stop relates to ramp generator position XACTUAL. Virtual stop relates to encoder position X_ENC.
[13] **en_virtual_stop_r**	RW 0x0	Enables automatic motor stop during active right virtual stop condition.
	0: DISABLED 1: ENABLED	Disable Enable
[12] **en_virtual_stop_l**	RW 0x0	Enables automatic motor stop during active left virtual stop condition.
	0: DISABLED 1: ENABLED	Disable Enable
[11] **en_softstop**	RW 0x0	The soft-stop mode always uses the deceleration ramp settings DMAX, V1, D1, V2, D2, VSTOP, and TZEROWAIT for stopping the motor. A stop occurs when the velocity sign matches the reference switch position (REFL, VIRTUAL_STOP_L for negative velocities, REFR, VIRTUAL_STOP_R for positive velocities) and the respective switch stop function is enabled. A hard stop also uses TZEROWAIT before the motor is released. Note: Do not use soft stop in combination with StallGuard2.
	0: HARD_STOP 1: SOFT_STOP	Hard stop Soft stop

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[10] **sg_stop**	RW 0x0	Enable stop by StallGuard. Disable to release the motor after the stop event. Program TCOOLTHRS for velocity threshold. This stop always results in a hard stop, ignoring the setting of en_softstop. Note: Do not enable during motor spin-up, wait until the motor velocity exceeds a certain value, where StallGuard delivers a stable result. This velocity threshold should be programmed using TCOOLTHRS.
	0: DISABLED 1: ENABLED	Disabled Enabled
[9] **en_latch_encoder**	RW 0x0	Latch encoder position to ENC_LATCH upon reference switch event.
	0: DISABLED 1: ENABLED	Disable Enable
[8] **latch_r_inactive**	RW 0x0	Activates latching of the position to XLATCH upon an inactive going edge on the right reference switch input REFR. The active level is defined by pol_stop_r.
	0: DISABLED 1: ENABLED	Disable Enable
[7] **latch_r_active**	RW 0x0	Activates latching of the position to XLATCH upon an active going edge on the right reference switch input REFR. Note: Activate latch_r_active to detect any spurious stop event by reading status_latch_r.
	0: DISABLED 1: ENABLED	Disable Enable
[6] **latch_l_inactive**	RW 0x0	Activates latching of the position to XLATCH upon an inactive going edge on the left reference switch input REFL. The active level is defined by pol_stop_l.
	0: DISABLED 1: ENABLED	Disable Enable
[5] **latch_l_active**	RW 0x0	Activates latching of the position to XLATCH upon an active going edge on the left reference switch input REFL. Note: Activate latch_l_active to detect any spurious stop event by reading status_latch_l.
	0: DISABLED 1: ENABLED	Disable Enable
[4] **swap_lr**	RW 0x0	Swap the left and right reference switch inputs REFL and REFR.
	0: DEFAULT 1: SWAPPED	Normal order Left and right switch swapped.
[3] **pol_stop_r**	RW 0x0	Inverts the active polarity of the right reference switch input.
	0: NON_INV 1: INV	Noninverted, high active: a high level on REFR stops the motor. Inverted, low active: a low level on REFR stops the motor.
[2] **pol_stop_l**	RW 0x0	Inverts the active polarity of the left reference switch input.
	0: NON_INV 1: INV	Noninverted, high active: a high level on REFL stops the motor. Inverted, low active: a low level on REFL stops the motor.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[1] **stop_r_enable**	RW 0x0	Enables automatic motor stop during active right reference switch input. Hint: The motor restarts in case the stop switch is released.
	0: DISABLED 1: ENABLED	Disable Enable
[0] **stop_l_enable**	RW 0x0	Enables automatic motor stop during active left reference switch input. Hint: The motor restarts in case the stop switch is released.
	0: DISABLED 1: ENABLED	Disable Enable

0x35: RAMP_STAT

Default: 0x0000C000

Ramp status and switch event status

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[15] **status_virtual_stop_r**	R 0x1	Virtual reference switch right status.
	0: INACTIVE 1: ACTIVE	Inactive Active
[14] **status_virtual_stop_l**	R 0x1	Virtual reference switch left status
	0: INACTIVE 1: ACTIVE	Inactive Active
[13] **status_sg**	R 0x0	Signals an active StallGuard input from the driver. Hint: When polling this flag, stall events may be missed – activate sg_stop to ensure not to miss the stall event.
	0: NOSTALL 1: STALL	Momentary inactive Momentary active
[12] **second_move**	RW, W1C 0x0	Signals that the automatic ramp required moving back in the opposite direction, example, due to on-the-fly parameter change. (write '1' to clear). #type = COW
	0: INACTIVE 1: ACTIVE	Normal motion Direction is changed to reach target.
[11] **t_zerowait_active**	R 0x0	Signals that TZEROWAIT is active after a motor stop. During this time, the motor is in standstill.
	0: INACTIVE 1: ACTIVE	Not active Motion controller in TZEROWAIT time.
[10] **vzero**	R 0x0	Signals the actual velocity is 0.
	0: V_NOT_0 1: V_0	Actual velocity is not 0. Actual velocity is 0.
[9] **position_reached**	R 0x0	Signals the target position is reached. This flag is set while XACTUAL and XTARGET match.
	0: POS_NOT_REACHED 1: POS_REACHED	XTARGET not reached. XTARGET reached.
[8] **velocity_reached**	R 0x0	Signals the target velocity is reached. This flag is set while VACTUAL and VMAX match.
	0: VMAX_NOT_REACHED 1: VMAX_REACHED	VMAX not reached. VMAX reached.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[7] **event_pos_reached**	RW, W1C 0x0	Signals the target position is reached (position_reached is active). (Write '1' to clear flag and interrupt condition). #type = COW
	0: INACTIVE 1: ACTIVE	Inactive Active
[6] **event_stop_sg**	RW, W1C 0x0	Signals an active StallGuard2 stop event. Resetting the register clears the stall condition and the motor may restart motion, unless the motion controller is stopped. (Write '1' to clear flag and interrupt condition). #type = COW
	0: INACTIVE 1: ACTIVE	Inactive Active
[5] **event_stop_r**	R 0x0	Active stop right condition due to stop switch or virtual stop. The stop condition and interrupt condition can be removed by setting RAMP_MODE to hold mode or by commanding a move to the opposite direction. In soft_stop mode, the condition remains active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or stop function also clears the flag, but the motor continues motion. This bit is available for the interrupt output signal.
	0: INACTIVE 1: ACTIVE	Inactive Active
[4] **event_stop_l**	R 0x0	Active stop left condition due to stop switch or virtual stop. The stop condition and interrupt condition can be removed by setting RAMP_MODE to hold mode or by commanding a move to the opposite direction. In soft_stop mode, the condition remains active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor continues motion. This bit is available for the interrupt output.
	0: INACTIVE 1: ACTIVE	Inactive Active
[3] **status_latch_r**	RW, W1C 0x0	1: Latch right ready (enable position latching using SW_MODE settings latch_r_active or latch_r_inactive) (write '1' to clear).
	0: INACTIVE 1: ACTIVE	Not latched Latched
[2] **status_latch_l**	RW, W1C 0x0	1: Latch left ready (enable position latching using SW_MODE settings latch_l_active or latch_l_inactive) (write '1' to clear).
	0: INACTIVE 1: ACTIVE	Not latched Latched
[1] **status_stop_r**	R 0x0	Reference switch right status.
	0: INACTIVE 1: ACTIVE	Inactive Active
[0] **status_stop_l**	R 0x0	Reference switch left status.
	0: INACTIVE 1: ACTIVE	Inactive Active

0x36: XLATCH

Default: 0x00000000
Ramp generator latch position

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **XLATCH**	R, signed 0x00000000	Ramp generator latch position, latches XACTUAL upon a programmable switch event (see SW_MODE). Hint: The encoder position can be latched to ENC_LATCH together with XLATCH to allow consistency checks.

0x38: ENCMODE

Default: 0x00100000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[10] **enc_sel_decimal**	RW 0x0	Encoder prescaler mode selection.
	0: BINARY 1: DECIMAL	Encoder prescaler divisor binary mode: counts ENC_CONST(fractional part)/65536. Encoder prescaler divisor decimal mode: counts in ENC_CONST(fractional part)/10000.
[9] **latch_x_act**	RW 0x0	Position latch configuration.
	0: DISABLED 1: LATCH_XACTUAL	Disabled Also latch XACTUAL position together with X_ENC. Allows latching the ramp generator position upon an N channel event as selected by pos_edge and neg_edge.
[8] **clr_enc_x**	RW 0x0	Encoder latch configuration.
	0: KEEP 1: CLEAR	Upon N event, X_ENC is latched to ENC_LATCH only. Latch and additionally clear encoder counter X_ENC at N-event.
[7:6] **pos_neg_edge**	RW 0x0	N channel event sensitivity.
	0: HIGH_ACTIVE 1: RISING_EDGE 2: FALLING_EDGE 3: BOTH_EDGES	N channel event is active during an active N event level. N channel is valid upon active going N event. N channel is valid upon inactive going N event. N channel is valid upon active going and inactive going N event.
[5] **clr_once**	RW 0x0	Position latch configuration.
	0: OFF 1: LATCH_ONCE	Disabled Latch or latch and clear X_ENC on the next N event following the write access.
[4] **clr_cont**	RW 0x0	Position latch configuration.
	0: OFF 1: LATCH_CONT	Disabled 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).

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[3] **ignore_AB**	RW 0x0	N event configuration
	0: N_AB_MATCH 1: N_AB_IGNORED	An N event occurs only when polarities given by pol_N, pol_A and pol_B match. Ignore A and B polarity for N channel event.
[2] **pol_N**	RW 0x0	Defines active polarity of N.
	0: LOW_ACTIVE 1: HIGH_ACTIVE	Low active High active
[1] **pol_B**	RW 0x0	Required B polarity for an N channel event.
	0: NEG 1: POS	Negative polarity Positive polarity
[0] **pol_A**	RW 0x0	Required A polarity for an N channel event.
	0: NEG 1: POS	Negative polarity Positive polarity

0x39: X_ENC

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **X_ENC**	RW, signed 0x00000000	Actual encoder position (signed).

0x3A: ENC_CONST

Default: 0x00010000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **ENC_CONST**	RW, signed 0x00010000	Accumulation constant (signed). 16-bit integer part, 16-bit fractional part. X_ENC accumulates ± ENC_CONST/(2^(16) × X_ENC) (binary) or ± ENC_CONST/(10^(4) × X_ENC) (decimal) ENCMODE bit enc_sel_decimal switches between decimal and binary setting. Use the sign to match rotation direction! Binary: ± [µsteps/2^(16)] ±(0…32767.9999847) Decimal: ±(0.0…32767.9999) Reset default = 1.0 (= 65536)

0x3B: ENC_STATUS

Default: 0x00000000
Encoder status information.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[0] **n_event**	RW, W1C 0x0	Signals detection of a previous or active N event. Write one to clear. #type = COW
	0: NO_EVENT 1: EVENT_DET	No event Event detected.

0x3C: ENC_LATCH

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **ENC_LATCH**	R, signed 0x00000000	Encoder position X_ENC latched on N event.

0x3E: VIRTUAL_STOP_L

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **VIRTUAL_STOP_L**	RW, signed 0x00000000	Virtual stop switch based on encoder or ramp position. A stop is raised, based on the signed comparison. virtual_stop_enc = 1: X_ENC <= VIRTUAL_STOP_L virtual_stop_enc = 0: XACTUAL <= VIRTUAL_STOP_L -2^(31)…+(2^(31)) - 1

0x3F: VIRTUAL_STOP_R

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **VIRTUAL_STOP_R**	RW, signed 0x00000000	Virtual stop switch based on encoder. A stop is raised, based on the signed comparison. virtual_stop_enc = 1: X_ENC >= VIRTUAL_STOP_R virtual_stop_enc = 0: XACTUAL >= VIRTUAL_STOP_R -2^(31)…+(2^(31)) - 1

0x58: ADC_VSUPPLY_TEMP

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[24:16] **ADC_TEMP**	R, unsigned 0x000	Temperature [°C] = (1.042 × ADC_TEMP - 264.6)°C
[8:0] **ADC_VSUPPLY**	R, unsigned 0x000	Vs [V] = 0,1409V × ADC_VSUPPLY

0x5A: OTW_OV_VTH

Default: 0x01FF01FF

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[24:16] **OVERTEMPPREWARNING_VTH**	RW, unsigned 0x1FF	Temperature prewarning limit. Configure, as desired, to warn in case of excess heat up. When ADC_TEMP >= OVERTEMPPREWARNING_VTH, otpw in DRV_STATUS is set to 1.
[8:0] **OVERVOLTAGE_VTH**	RW, unsigned 0x1FF	Selection of overvoltage threshold to match system environment. When ADC_VSUPPLY >= OVERVOLTAGE_VTH, ov in DRV_STATUS is set to 1. Output this flag to a diagnostic output pin (DO0 or DO1), if desired.

0x60: MSLUT_0

Default: 0xAAAAB554
Microstep table entries 0…31.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_0**	RW, unsigned 0xAAAAB554	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x61: MSLUT_1

Default: 0x4A9554AA
Microstep table entries 32…63.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_1**	RW, unsigned 0x4A9554AA	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x62: MSLUT_2

Default: 0x24492929
Microstep table entries 64…95.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_2**	RW, unsigned 0x24492929	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x63: MSLUT_3

Default: 0x10104222
Microstep table entries 96…127.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_3**	RW, unsigned 0x10104222	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x64: MSLUT_4

Default: 0xFBFFFFFF
Microstep table entries 128…159.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_4**	RW, unsigned 0xFBFFFFFF	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x65: MSLUT_5

Default: 0xB5BB777D
Microstep table entries 160…191.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_5**	RW, unsigned 0xB5BB777D	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x66: MSLUT_6

Default: 0x49295556
Microstep table entries 192…223.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_6**	RW, unsigned 0x49295556	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x67: MSLUT_7

Default: 0x00404222
Microstep table entries 224…255.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:0] **MSLUT_7**	RW, unsigned 0x00404222	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_U and CUR_V are stored for MSCNT position 0 in START_SIN and START_SIN120. ofs31, ofs30, …, ofs01, ofs00 … ofs255, ofs254, …, ofs225, ofs224 Reset default = sine wave table

0x68: MSLUTSEL

Default: 0xFFFF8056

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31:24] **X3**	RW, unsigned 0xFF	LUT segment 3 start. The sine wave lookup 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

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[23:16] **X2**	RW, unsigned 0xFF	LUT segment 2 start. The sine wave lookup 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
[15:8] **X1**	RW, unsigned 0x80	LUT segment 1 start. The sine wave lookup 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
[7:6] **W3**	RW 0x1	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
	0: W3_SUB1_ADD0 Current MSLUT entry 0 (1): -1 (+0) to sinewave. 1: W3_ADD0_ADD1 Current MSLUT entry 0 (1): +0 (+1) to sinewave. 2: W3_ADD1_ADD2 Current MSLUT entry 0 (1): +1 (+2) to sinewave. 3: W3_ADD2_ADD3 Current MSLUT entry 0 (1): +2 (+3) to sinewave.	
[5:4] **W2**	RW 0x1	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
	0: W2_SUB1_ADD0 Current MSLUT entry 0 (1): -1 (+0) to sinewave. 1: W2_ADD0_ADD1 Current MSLUT entry 0 (1): +0 (+1) to sinewave. 2: W2_ADD1_ADD2 Current MSLUT entry 0 (1): +1 (+2) to sinewave. 3: W2_ADD2_ADD3 Current MSLUT entry 0 (1): +2 (+3) to sinewave.	

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[3:2] **W1**	RW 0x1	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
	0: W1_SUB1_ADD0 1: W1_ADD0_ADD1 2: W1_ADD1_ADD2 3: W1_ADD2_ADD3	Current MSLUT entry 0 (1): -1 (+0) to sinewave. Current MSLUT entry 0 (1): +0 (+1) to sinewave. Current MSLUT entry 0 (1): +1 (+2) to sinewave. Current MSLUT entry 0 (1): +2 (+3) to sinewave.
[1:0] **W0**	RW 0x2	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
	0: W0_SUB1_ADD0 1: W0_ADD0_ADD1 2: W0_ADD1_ADD2 3: W0_ADD2_ADD3	Current MSLUT entry 0 (1): -1 (+0) to sinewave. Current MSLUT entry 0 (1): +0 (+1) to sinewave. Current MSLUT entry 0 (1): +1 (+2) to sinewave. Current MSLUT entry 0 (1): +2 (+3) to sinewave.

0x69: MSLUTSTART

Default: 0x00D60000

Start values are transferred to the microstep registers CUR_U and CUR_V whenever the reference position MSCNT = 0 is passed.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[23:16] **START_SIN120**	RW, unsigned 0xD6	START_SIN120 gives the absolute value for cosine wave microstep table entry at MSCNT = 0 (table position 214).
[7:0] **START_SIN**	RW, unsigned 0x00	START_SIN gives the absolute value at microstep table entry 0.

0x6A: MSCNT

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[9:0] **MSCNT**	R, unsigned 0x000	Microstep counter. Indicates actual position in the microstep table for CUR_U. Note: Move to a position where MSCNT is zero before reinitializing MSLUTSTART or MSLUT and MSLUTSEL.

0x6B: MSCURACT

Default: 0x00F70000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[24:16] **CUR_V**	R, signed 0x0F7	Actual microstep current for motor phase V (sine wave 120°) as read from MSLUT (not scaled by current).
[8:0] **CUR_U**	R, signed 0x000	Actual microstep current for motor phase U (sine wave) as read from MSLUT (not scaled by current).

0x6C: CHOPCONF

Default: 0x10010000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[29] **dedge**	RW 0x0	Enable double edge step pulses.
	0: RISING_EDGE 1: BOTH_EDGES	Disabled Enable step impulse at each step edge to reduce step frequency requirement.
[28] **intpol**	RW 0x1	Interpolation to 256 microsteps.
	0: DISABLED 1: ENABLED	No interpolation The actual microstep resolution (MRES) is extrapolated to 256 microsteps for smoothest motor operation (useful for STEP and DIR operation, only).
[27:24] **MRES**	RW 0x0	Microstep resolution selection. The electrical values given describe the number of steps taken per electrical quarter sine wave (E). The resulting mechanical microstep resolution (M) that describes the number of microsteps between two motor fullsteps is 2/3 of the corresponding value. Take into account that the microstep position, when switching to a lower resolution, determines the sequence of patterns. step width = 2^MRES [electrical microsteps] step width = 2/3 × 2^MRES [motor microsteps].
	0: RES_256 1: RES_128 2: RES_64 3: RES_32 4: RES_16 5: RES_8 6: RES_4 7: RES_2 8: RES_1	E: 256, M: 170.66 E: 128, M: 85.33 E: 64, M: 42.66 E: 32, M: 21.33 E: 16, M: 10.66 E: 8, M: 5.33 E: 4, M: 2.66 E: 2, M: 1.33 E: 1, M: 0.66
[16:15] **TBL**	RW 0x2	Blank time selection for current measurement. Note: 1 or 2 is recommended for most applications.
	0: TBL_1250 1: TBL_1750 2: TBL_2500 3: TBL_3625	~1.25µs ~1.75µs ~2.5µs ~3.625µs

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[14] **chm**	RW 0x0	Chopper mode.
	0: SPREADCYCLE 1: RESERVED	Standard mode (SpreadCycle) Reserved
[12] **CSYNC**	RW 0x0	Chopper synchronization.
	0: FREE 1: STEP_SYNC	Chopper runs freely. Chopper is synchronized to step frequency.
[11] **sg_speed**	RW 0x0	StallGuard speed.
	0: DYNAMIC 1: FILTERED	Standard mode, high time resolution for stallGuard. StallGuard uses more filtering, use for low motor velocity only.
[10] **nosd**	RW 0x0	Skip slow decay phase.
	0: SLOW_DECAY_EN 1: SLOW_DECAY_SKIP	Each chopper on cycle is followed by a slow decay phase, as set by TOFF. Slow decay phases are skipped between the chopper phases, except directly following a short-to-GND or chopper synchronization. Minimum blank time then is 36 clocks.
[9:4] **HYST**	RW, unsigned 0x00	DAC hysteresis setting: %000000 … %111111: 0 … 63 (1/512 of this setting adds to coil current setting). Attention: Effective HYST/2 must be ≤ 255-sinewave peak (248 at maximum current setting). Reduce current setting to 230 for maximum hysteresis. Do not work with too small setting (poor performance).
[3:0] **TOFF**	RW 0x0	TOFF off-time and driver enable. Off-time setting controls duration of slow decay phase. N CLK = 24 + 32 × TOFF %0000: Driver disable, all bridges off. %0001: 1 – use only with TBL ≥ 2. %0010 … %1111: 2 … 15
	0: DRIVER_OFF 1: TOFF_3_5US 2: TOFF_5_5US 3: TOFF_7_5US 4: TOFF_9_5US 5: TOFF_11_5US 6: TOFF_13_5US 7: TOFF_15_5US 8: TOFF_17_5US 9: TOFF_19_5US 10: TOFF_21_5US 11: TOFF_23_5US 12: TOFF_25_5US 13: TOFF_27_5US 14: TOFF_29_5US 15: TOFF_31_5US	Driver disabled and all bridges off. Slow decay phase duration ~3.5µs Slow decay phase duration ~5.5µs Slow decay phase duration ~7.5µs Slow decay phase duration ~9.5µs Slow decay phase duration ~11.5µs Slow decay phase duration ~13.5µs Slow decay phase duration ~15.5µs Slow decay phase duration ~17.5µs Slow decay phase duration ~19.5µs Slow decay phase duration ~21.5µs Slow decay phase duration ~23.5µs Slow decay phase duration ~25.5µs Slow decay phase duration ~27.5µs Slow decay phase duration ~29.5µs Slow decay phase duration ~31.5µs

0x6D: COOLCONF

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[24] **sfilt**	RW 0x0	StallGuard2 filter enable.
	0: FILT_DIS 1: FILT_EN	Standard mode, high time resolution for StallGuard. Filtered mode, StallGuard signal updated for each three measurements only to compensate for motor pole tolerances.
[23] **thigh_sg_off**	RW 0x0	Stop-on-stall handling when exceeding VHIGH.
	0: DIS_CS 1: DIS_CS_STOP_ON_STALL	Exceeding VHIGH only disables CoolStep. Exceeding VHIGH disables stop-on-stall function as well as CoolStep.
[22:16] **sgt**	RW, signed 0x00	StallGuard2 threshold value. This signed value controls the StallGuard2 level for the stall signal 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.
[15] **seimin**	RW 0x0	Minimum current for smart current control.
	0: IRUN_DIV2 1: IRUN_DIV4	1/2 of current setting (IRUN). 1/4 of current setting (IRUN).
[14:12] **sedn**	RW 0x0	Current down step speed.
	0: STEP_DOWN_EACH_8 1: STEP_DOWN_EACH_4 2: STEP_DOWN_EACH_2 3: STEP_DOWN_EACH_1 4: STEP_DOWN2_EACH_1 5: STEP_DOWN4_EACH_1 6: STEP_DOWN8_EACH_1 7: STEP_DOWN16_EACH_1	For each 8 StallGuard2 values, decrease by one. For each 4 StallGuard2 values, decrease by one. For each 2 StallGuard2 values, decrease by one. For each StallGuard2 value, decrease by one. For each StallGuard2 value, decrease by two. For each StallGuard2 value, decrease by four. For each StallGuard2 value, decrease by eight. For each StallGuard2 value, decrease by sixteen.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[11:8] **semax**	RW 0x0	StallGuard2 hysteresis value for smart current control. If the StallGuard2 result is equal to or above (SEMIN + SEMAX + 1) × 32, the motor current is decreased to save energy.
	0: SEMAX_32	Reduce current when (32 × SEMIN + 32) smaller or equal to SG_RESULT.
	1: SEMAX_64	Reduce current when (32 × SEMIN + 64) smaller or equal to SG_RESULT.
	2: SEMAX_96	Reduce current when (32 × SEMIN + 96) smaller or equal to SG_RESULT.
	3: SEMAX_128	Reduce current when (32 × SEMIN + 128) smaller or equal to SG_RESULT.
	4: SEMAX_160	Reduce current when (32 × SEMIN + 160) smaller or equal to SG_RESULT.
	5: SEMAX_192	Reduce current when (32 × SEMIN + 192) smaller or equal to SG_RESULT.
	6: SEMAX_224	Reduce current when (32 × SEMIN + 224) smaller or equal to SG_RESULT.
	7: SEMAX_256	Reduce current when (32 × SEMIN + 256) smaller or equal to SG_RESULT.
	8: SEMAX_288	Reduce current when (32 × SEMIN + 288) smaller or equal to SG_RESULT.
	9: SEMAX_320	Reduce current when (32 × SEMIN + 320) smaller or equal to SG_RESULT.
	10: SEMAX_352	Reduce current when (32 × SEMIN + 352) smaller or equal to SG_RESULT.
	11: SEMAX_384	Reduce current when (32 × SEMIN + 384) smaller or equal to SG_RESULT.
	12: SEMAX_416	Reduce current when (32 × SEMIN + 416) smaller or equal to SG_RESULT.
	13: SEMAX_448	Reduce current when (32 × SEMIN + 448) smaller or equal to SG_RESULT.
	14: SEMAX_480	Reduce current when (32 × SEMIN + 480) smaller or equal to SG_RESULT.
	15: SEMAX_512	Reduce current when (32 ×SEMIN + 512) smaller or equal to SG_RESULT.
[6:5] **seup**	RW 0x0	Current up step width. Current increment steps per measured StallGuard2 value.
	0: STEP_UP_8	8
	1: STEP_UP_16	16
	2: STEP_UP_32	32
	3: STEP_UP_64	64

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[3:0] **semin**	RW 0x0	Minimum StallGuard2 value for smart current control and smart current enable. If the StallGuard2 result falls below SEMIN × 32, the motor current is increased to reduce motor load angle.
	0: OFF 1: SEMIN_32 2: SEMIN_64 3: SEMIN_96 4: SEMIN_128 5: SEMIN_160 6: SEMIN_192 7: SEMIN_224 8: SEMIN_256 9: SEMIN_288 10: SEMIN_320 11: SEMIN_352 12: SEMIN_384 13: SEMIN_416 14: SEMIN_448 15: SEMIN_480	CoolStep smart current control off. Increase current on SG_RESULT smaller 32. Increase current on SG_RESULT smaller 64. Increase current on SG_RESULT smaller 96. Increase current on SG_RESULT smaller 128. Increase current on SG_RESULT smaller 160. Increase current on SG_RESULT smaller 192. Increase current on SG_RESULT smaller 224. Increase current on SG_RESULT smaller 256. Increase current on SG_RESULT smaller 288. Increase current on SG_RESULT smaller 320. Increase current on SG_RESULT smaller 352. Increase current on SG_RESULT smaller 384. Increase current on SG_RESULT smaller 416. Increase current on SG_RESULT smaller 448. Increase current on SG_RESULT smaller 480.

0x6F: DRV_STATUS

Default: 0x00000000

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[31] **stst**	R 0x0	Standstill indicator. This flag indicates motor standstill in each operation mode. Depending on fast_standstill setting, this occurs 2^(20)^()clocks or 2^(18) clocks after the last step pulse.
	0: INACTIVE 1: ACTIVE	Motor moving. Motor in standstill.
[29] **ol**	R 0x0	Open load indicator.
	0: OPERATIONAL 1: OPEN_LOAD	Normal operation Open load detected. 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.
[27] **s2g**	R 0x0	Short-to-ground indicator.
	0: 0 1: 1	Normal operation Short-to-GND detected. The driver is disabled. The flags stay active until the driver is disabled by software (TOFF = 0) or by the ENN input.
[26] **otpw**	R 0x0	Overtemperature prewarning flag.
	0: OPERATIONAL 1: OTPW_REACHED	Normal operation Overtemperature prewarning threshold is exceeded.

BITS AND NAME	TYPE AND RESET	DESCRIPTION
[25] **ot**	R 0x0	Overtemperature flag.
	0: OPERATIONAL 1: ERROR	Normal operation Overtemperature limit is reached. Drivers are disabled until the IC has cooled down.
[24] **stallguard**	R 0x0	StallGuard status.
	0: NORMAL 1: STALL	Normal operation Motor stall detected by StallGuard.
[23:16] **CS_ACTUAL**	R, unsigned 0x00	Actual motor target current (actual CoolStep current if CoolStep function is enabled). Use for monitoring CoolStep, and IRUN and IHOLD current scaling.
[12] **s2vs**	R 0x0	Short-to-supply indicator.
	0: OPERATIONAL 1: ERROR	No error Short-to-supply detected. The driver is disabled. The flags stay active until the driver is disabled by software (TOFF = 0) or by the ENN input.
[11] **ov**	R 0x0	Overvoltage flag indicator. Flag is set if ADC_VSUPPLY >= OVERVOLTAGE_VTH.
	0: NO_OV 1: OV	Threshold not yet reached. Threshold reached.
[10] **seq_stopped**	R 0x0	1: Sequencer is stopped by ENC_A.
	0: OPERATIONAL 1: STOPPED	Sequencer running. Sequencer stopped.
[9:0] **SG_RESULT**	R, unsigned 0x000	StallGuard2 result Mechanical load measurement: The StallGuard2 result gives a means to measure mechanical motor load. A higher value means lower mechanical load. A value of 0 signals highest load. With optimum SGT setting, this is an indicator for a motor stall. The stall detection compares SG_RESULT to 0 to detect a stall. SG_RESULT is used as a base for CoolStep operation by comparing it to a programmable upper and lower limit. StallGuard2 works best with microstep operation.

Ordering Information

PART NUMBER	TEMPERATURE RANGE	PIN-PACKAGE
TMC3241AFV+	-40°C to +125°C	30L FC2QFN, 6mm × 6mm
TMC3241AFV+T	-40°C to +125°C	30L FC2QFN, 6mm × 6mm

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

T Denotes tape-and-reel.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	10/26	Initial release	—

NOTES

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices. Trademarks and registered trademarks are the property of their respective owners. All Analog Devices products contained herein are subject to release and availability.