65V/4ARMS+ Fully Integrated Smart Stepper Motor Controller and Driver

TMC5262

General Description

The TMC5262 is a miniaturized, smart, high-power, single-axis 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.

StealthChop+ ensures noiseless operation and maximum efficiency, optimal motor torque control, and dampening of mid-range motor resonances for a smoothly running motor with low mechanical vibrations. Combined with StallGuard+, CoolStep+, and µDcStep, this results in a highly efficient and versatile stepper motor driver. An unloaded Nema 17 motor automatically runs with as low as a few 10mA of supply current.

The TMC5262 combines a flexible eight-point ramp generator for automatic positioning with a smart 256 microstep motor driver and two integrated 65V, 4.25A 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 an on-chip scope interface for tuning and checking (RT-OCSI), 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 x 6mm FCQFN package.

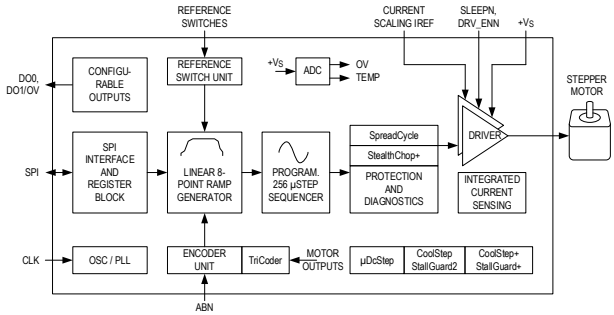
Key Applications

- Textile, Sewing Machines, Knitting Machines
- Lab Automation, IVD
- Robotics and Factory Automation
- 3D Printers, Label Printers
- POS, Massage Chairs
- ATM, Cash Recycler, Bill Validators, Cash Machines
- 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 H-Bridge (Typical at 25°C):
  - I_{RMS} = 4.25A_{RMS} (6A Full-Scale Sine Peak)
  - I_{MAX} = 9.0A (Bridge Peak Current)
- SPI for Configuration, Control, and Diagnostics
- Fully Integrated Lossless Current Sensing
- Eight-Point Motion Controller for Low Jerk
- Step and Direction Interface Option
- (Periodic) Position Compare Function
- Incremental Encoder and Reference Switch Interface
- TriCoder Sensorless Standstill Stepless Detection and Full-Step Encoder
- Highest Resolution of 256 Microsteps
- Programmable Sine Wave Table to Match Motor
- StealthChop+ Silent and Smooth Motor Operation with Resonance Dampening
- SpreadCycle Highly Dynamic Motor Chopper Option
- StallGuard2, StallGuard+ Sensorless Load Detection
- CoolStep, CoolStep+ Current Control for Application Energy Savings up to 90%
- µDcStep Load Dependent Velocity Adaptation
- Motor Coil Resistance Measurement Integrated
- Passive Braking and Freewheeling Mode
- Chip Temperature and Supply Voltage Measurement
- Full Protection and Diagnostics
- Real-Time On-Chip Scope Interface (RT-OCSI)
- Overvoltage Protection Output
- Compact 30-Pin 6mm x 6mm FCQFN Package

Simplified Block Diagram



19-102041; Rev.0, 09/25

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

Downloaded from Arrow.com.

TABLE OF CONTENTS

General Description 1
Key Applications 1
Benefits and Features 1
Simplified Block Diagram 1
Absolute Maximum Ratings 10
Electrical Characteristics 10
Package Information 15
FC2QFN30 6mm x 6mm 15
Package Marking Specification 15
Pin Descriptions 15
Pin Configurations 17
Functional Diagrams 18
Detailed Description 19
Principles of Operation 19
Full Featured Motion Controller and Driver 19
Key Concepts 20
Control Interface 20
Integrated Eight-Point Motion Controller 20
Step and Direction Interface 20
Automatic Standstill Power Down 21
StealthChop+ and SpreadCycle Driver 21
Mechanical Load Sensing with StallGuard2 and StallGuard+ 22
Load Adaptive Current Control with CoolStep and CoolStep+ 22
uDcStep 22
Real Time On-Chip Scope Interface 22
TriCoder 22
Encoder Interface 22
Serial Peripheral Interface (SPI) 24
SPI Datagram Structure 24
Selection of Write/Read (WRITE_notREAD) 24
SPI Status Bits Transferred with Each Datagram Read Back 25
Data Alignment 25
SPI Signals 25
SPI Timing 25
Step and Direction Interface 27

Downloaded from Arrow.com.

Enabling the STEP and DIR Interface 27
Timing 27
Changing Resolution 27
MicroPlyer Step Interpolator and Standstill Detection 28
StealthChop+ 30
StealthChop+ Principle of Operation 31
Enabling StealthChop+ Procedure 32
Why Do Detailed Tuning of StealthChop+? 32
Configuration of the StealthChop+ Current Regulator 33
Current Regulator P Tuning 34
Current Regulator I Tuning 34
Rule of Thumb for Current Regulator PI Tuning 35
Example for Current Regulator PI Tuning 35
CUR_P and CUR_I Tuning Using a Scope 36
Configuring StealthChop+ Angle Regulator 37
Application Range of the Angle Regulator 38
Angle P Tuning 38
Angle I Tuning 39
Rule of Thumb for Angle PI 41
Options for the StealthChop+ Chopper 41
Chopper Frequency 41
Freewheeling and Passive Braking Modes 42
StallGuard+ 43
Understanding *SGP_RESULT* 43
Stall Detection 44
Low Velocity Stall Detection 45
COIL_INDUCT Setting 45
COIL_INDUCT Experimental Optimization 45
R_COIL Measurement 46
Example R_COIL_AUTO Interpretation 47
R_COIL Measurement Flow Chart 47
CoolStep+ 50
Procedure to Configure CoolStep+ 51
Tuning CoolStep+ 51
Target Load Reserve Generator 51
Rule of Thumb for Load Reserve 52
CoolStep+ Regulator 53
CoolStep+ P Tuning 54

Downloaded from Arrow.com.

CoolStep+ I Tuning 54
COOL_PI_DOWN_LIMIT Tuning 54
Rule of Thumb for CoolStep+ PI and Slope Limits 55
Velocity Thresholds Relevant for StealthChop+ 56
Understanding the Back-EMF Constant of a Motor 57
Overview of Velocity Dependent Features 58
Open Load Detection 60
µDcStep 61
Enabling µDcStep 62
SpreadCycle and Classic Chopper 63
SpreadCycle Chopper 64
Direct Mode 66
Classic Constant Off-Time Chopper 66
StallGuard2 Load Measurement 67
Tuning StallGuard2 Threshold SGT 68
Variable Velocity Limits *TCOOLTHRS* and *THIGH* 70
Small Motors with High Torque Ripple and Resonance 70
Temperature Dependence of Motor Coil Resistance 70
Accuracy and Reproducibility of StallGuard2 Measurement 70
StallGuard2 Update Rate and Filter 70
Detecting a Motor Stall 71
Homing with StallGuard2 71
Limits of the StallGuard2 Operation 71
CoolStep Load Adaptive Current Scaling 71
Setting Up for CoolStep 71
Tuning CoolStep 73
Response Time 73
Low Velocity and Standby Operation 73
Integrated Current Sense 74
Setting the Full-Scale Current Range 74
Interpreting ADC Values 76
Slope Control 76
Velocity-Based Mode Control 77
Ramp Generator 79
Real-World Unit Conversion 79
Motion Profiles 80
Ramp Mode 80
Eight-Point Ramp 81

Downloaded from Arrow.com.

Start and Stop Velocity 81
Velocity Mode 84
Early Ramp Termination 84
Application Example: Joystick Control 85
Reference Switches 85
Implementing a Homing Procedure: 86
Homing with a Third Switch 86
Virtual Reference Switches 87
Ramp Generator Response Time 87
External STEP/DIR Driver 87
Position Compare Functions 88
TriCoder – Back-EMF Sensorless Standstill Steploss Detection 89
TriCoder BEMF Decoder Principle of Operation 89
Motor and Velocity Requirements for TriCoder Operation 90
Time to TriCoder Enable 90
Configuring the TriCoder 90
Enable TriCoder 90
Operational Settings 91
ABN Incremental Encoder Interface 92
N Signal 92
The Encoder Counter X_ENC 92
The Register ENC_STATUS 92
The Encoder Constant ENC_CONST 92
Setting the Encoder to Match Motor Resolution 93
Sine Wave Lookup Table 94
Microstep Table 94
Matching the Phase Shift to the Motor 95
Reset, Disable/Stop, and Power Down 97
Emergency Stop 97
External Reset and Sleep Mode 97
Diagnostics and Protections 98
Diagnostic Outputs 98
Real-Time On-Chip Scope Interface (RT-OCSI) 98
RT-OCSI Examples of Use 101
Overcurrent Protection 102
Thermal Protection and Shutdown 103
Temperature Measurement 103
Chip Temperature Measurement 103

Downloaded from Arrow.com.

Motor Temperature Measurement 103
Overvoltage Protection and OV Output 104
Short Protection (Short-to-GND and Short-to-V_S) 104
Open-Load Diagnostics 105
Undervoltage Lockout Protection 105
ESD Protection 105
Clock Oscillator and Clock Input 106
Internal PLL Block 106
Using the Internal Clock 108
Using an External Clock 108
Quick Configuration Guides 109
PLL Initialization Guide 109
Current Setting Guide 111
StealthChop+ Configuration 112
StallGuard+ in Combination with StealthChop+ 113
StallGuard+ in Combination with StealthChop+ 114
CoolStep+ in Combination with StealthChop+ 115
SpreadCycle Configuration 116
CoolStep in Combination with SpreadCycle 117
Moving the Motor Using the Motion Controller 118
Initialization and Configuration Example 120
Design for Sustainability 120
Register Map 121
Ordering Information 240

Downloaded from Arrow.com.

LIST OF FIGURES

Figure 1. Package Marking (Top View) 15
Figure 2. TMC5262 Pinout with Bump Side Down 17
Figure 3. Block Diagram 18
Figure 4. Block Diagram with Typical External Components 19
Figure 5. Automatic Motor Current Control at Standstill and Ramp-Up 21
Figure 6. Power Saving with StealthChop+ and CoolStep+ vs. Pure SpreadCycle (No Load and High Load, Motor QSH4218-47-28-040 at 24V Supply) 22
Figure 7. SPI Timing Diagram 26
Figure 8. STEP and DIR Signal Timing 27
Figure 9. STEP and DIR Signal Input Filter Structure 27
Figure 10. MicroPlyer Microstep Interpolation with Rising STEP Frequency (Example: 16 to 256) 29
Figure 11. StealthChop+ Regulator Scheme 31
Figure 12. StealthChop+ Current Regulator 33
Figure 13. Tuning the Current Regulator by Monitoring the Current Step Response When Switching from 0 to a Higher IHOLD Current. 34
Figure 14. Too Low CUR_P (50% CUR_P_{50}) and CUR_I (25% CUR_P_{50}): Slow Current Ramp Up, Overshoot Upon Deceleration 36
Figure 15. High CUR_P (10 × CUR_P_{50}) Gives Fast Response at Start and Stop. Remaining Difference Only Slowly Compensated Due to Low CUR_I (10% of CUR_P) 36
Figure 16. High CUR_P (10 × CUR_P_{50}) with Increased CUR_I (25% of CUR_P) Yields Quick Current Regulation to the Set Point. 37
Figure 17. CUR_P = CUR_P_{50} with Increased CUR_I (2 × CUR_P_{50}) also Yields Quick Current Regulation to the Set Point with Minor Over- and Undershoot. 37
Figure 18. StealthChop+ Angle Regulator 38
Figure 19. Motor in Mid-Range Resonance (Close to Point of Stalling) Resolved by Increasing ANGLE_P to 250 39
Figure 20. Switching on ANGLE_I from 0 to Any Value Starts Compensating the ANGLE_ERROR to +-0 39
Figure 21. Motor Direction of Rotation Reversed Twice. Reversal with Too Low ANGLE_I Leads to Phase Jerk due to Slow Adaptation. The Dip Shows ANGLE_ERROR Cleared to 0 at a Velocity Below the Limit Given by T_RCOIL_MEAS. 40
Figure 22. Increasing ANGLE_I from a Low Value up to 50 to Dampen Mid-Range Resonance 40
Figure 23. Trace of Motor Current Using ADC Results at Low Velocity in StealthChop+ to Check for Undisturbed Current Measurement 42
Figure 24. Structure and Basic Parameters of StallGuard+ Load Reserve Calculation 43
Figure 25. StallGuard+ Value in Different Quadrants of Motor Operation 44
Figure 26. Using SGP_THRS for Stopping the Motor Upon High Load (Plotting Velocity VACTUAL and SGP_RESULT) 44
Figure 27. Good Setting – Motor Starts Stalling when SGP_RESULT Comes Close to 0 Versus Too Low Setting (Stall Starts Before Reaching 0) 46
Figure 28. Change in StallGuard+ Result SGP_RESULT due to Coil Heatup when Working at Very Low Velocity 46
Figure 29. Flowchart for R_{COIL} Resistance Measurement (Following Motor Current Setting and Enabling of StealthChop+) 48
Figure 30. Motor Torque vs. Load Angle 50
Figure 31. Relative Efficiency vs. Load Angle 50
Figure 32. Function of Target Load Reserve Generator Showing Asymmetric Setting with Reduced Efficiency in Generatoric Operation 52
Figure 33. Structure and Basic Parameters of CoolStep+ Regulator 53
Figure 34. PI Regulator Setting too Small – SGP_RESULT (Green) Undershoots Upon Load Increment vs. Sufficiently High Setting 54
Figure 35. Limitation of Falling Slope Controlled by COOL_PI_DOWN_LIMIT 55
Figure 36. SGP_RESULT Decreasing with Rising Mechanical Load and Tracking COOLSTEP_LOAD_RESERVE Until Maximum Current (CS_ACTUAL = 255) is Reached. Limited Slope for Current Reduction (COOL_PI_DOWN_LIMIT) Upon Load Decreasing 55
Figure 37. Example for Velocity Profile with µDcStep 61

Downloaded from Arrow.com.

Figure 38. Typical Chopper Decay Phases (Sense Resistor is Only Symbolic for Current Measurement in Low-Side Path) 63
Figure 39. SpreadCycle Chopper Scheme Showing the Coil Current During a Chopper Cycle 65
Figure 40. Classic Constant Off-Time Chopper with Offset Showing Coil Current 66
Figure 41. Zero Crossing with Classic Chopper and Correction Using Sine Wave Offset 67
Figure 42. Function Principle of StallGuard2 68
Figure 43. Example: Optimum SGT Setting and StallGuard2 Reading with an Example Motor 70
Figure 44. CoolStep Adapts Motor Current to the Load 72
Figure 45. Choice of Velocity-Dependent Modes 77
Figure 46. Ramp Generator Velocity Trace Showing Second Move into Negative Direction 80
Figure 47. Illustration of Optimized Motor Torque Usage with the Ramp Generator 81
Figure 48. Eight-Point Ramp with VMAX Not Reached Due to Too Low Distance 82
Figure 49. V2 Not Reached and No AMAX and DMAX Phase Due to Low Distance 82
Figure 50. V1 Not Reached Due to Low Distance 83
Figure 51. TVMAX Not Kept Due to Low Distance 83
Figure 52. Eight-Point Ramp Examples with On-the-Fly Target Position Change 84
Figure 53. Using Reference Switches (Example) 85
Figure 54. Virtual Stop Switches and Limit Visualization 87
Figure 55. Detection of Motor Movement 89
Figure 56. .Outline of ABN Signals of an Incremental Encoder 92
Figure 57. LUT Programming Example 94
Figure 58. Shifting the Cosine Wave through OFFSET_SIN90 95
Figure 59. Schematic of DO0 and DO1 Outputs 98
Figure 60. Monitoring CoolStep+ Using the RT-OCSI 101
Figure 61. Monitoring the Current PI Regulator Using the RT-OCSI 101
Figure 62. Monitoring the CoolStep+ PI Regulator Using the RT-OCSI 102
Figure 63. Brake Chopper Circuit Example 104
Figure 64. Block Diagram of PLL 106
Figure 65. Quick Configuration Guide for Starting the PLL 109
Figure 66. Quick Configuration Guide for Current Setting 111
Figure 67. Quick Configuration Guide for StealthChop+ Operation 112
Figure 68. Quick Configuration Guide for Basic StallGuard+ Configuration (Following Configuration of StealthChop+)113
Figure 69. Quick Configuration Guide for Basic StallGuard+ Configuration (Following Configuration of StealthChop+)114
Figure 70. Quick Configuration Guide for CoolStep+ (Following Configuration of StealthChop+) 115
Figure 71. Quick Configuration Guide for SpreadCycle 116
Figure 72. Quick Configuration Guide for CoolStep with SpreadCycle 117
Figure 73. Quick Configuration Guide for Moving a Motor in Velocity Mode 118
Figure 74. Quick Configuration Guide for Moving a Motor to a Target Position 118
Figure 75. Quick Configuration Guide for Motion Ramp Parameter Settings 119

Downloaded from Arrow.com.

LIST OF TABLES

Table 1. Packaging Marking Description 15
Table 2. TMC5262 Feature Overview 20
Table 3. SPI Datagram Structure 24
Table 4. SPI Read/Write Example Flow 24
Table 5. SPI_STATUS – Status Flags Transmitted with SPI Access in Bits 39 to 32 25
Table 6. Fullstep/Half Step Lookup Table Values for Phase A/B Coil Currents 28
Table 7. Enabling StealthChop Procedure 32
Table 8. Selection Options for PWM Frequency 41
Table 9. Procedure to Configure CoolStep+ 51
Table 10. Velocity Thresholds Relevant for StealthChop+ 56
Table 11. Overview of Velocity Dependent StealthChop+ Features 58
Table 12. µDcStep Enable Procedure 62
Table 13. Parameters Controlling SpreadCycle and Classic Constant Off-Time Chopper 64
Table 14. SpreadCycle Mode Parameters 65
Table 15. Parameters Controlling Constant Off-Time Chopper Mode 67
Table 16. StallGuard2-Related Parameters 68
Table 17. Procedure to Tune StallGuard 68
Table 18. Procedure to Tune StallGuard Low Velocity 69
Table 19. CoolStep Critical Parameters 71
Table 20. CoolStep Additional Parameters and Status Information 72
Table 21. Full-Scale Current Register Configuration for R_{REF} = 12kΩ 74
Table 22. Parameters Controlling the Motor Current 75
Table 23. Velocity-Based Mode Control Parameters 77
Table 24. Ramp Generator Parameters vs. Units 79
Table 25. Implementing a Homing Procedure 86
Table 26. Homing with a Third Switch Procedure 86
Table 27. X_COMPARE_REPEAT Options and Periodic Pulse Behavior 88
Table 28. TriCoder Control Parameters 91
Table 29. Encoder Example Settings for a 200 Fullstep Motor with 256 Microsteps 93
Table 30. Real-Time On-Chip Scope Interface (RT-OCSI) DAC Configuration Options 99
Table 31. PLL Control Register PLL Parameters 107

Downloaded from Arrow.com.

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
OA1, OA2, OB1, OB2 .....-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
VDD 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							

Downloaded from Arrow.com.

(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}		mV
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			

Downloaded from Arrow.com.

(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						
IREF Pin Resistor	R_{REF}			12		kΩ
IREF Output Voltage	V_{REF}		0.882	0.9	0.918	V
Full Scale Current Constant	KIFS	IFS = 1.5A		18		A x kΩ
Full Scale Current Constant	KIFS	IFS = 3A		36		A x kΩ
Full Scale Current Constant	KIFS	IFS = 4.5A		54		A x 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 rising edge to driver on			1	µs
Disable Time	t_{DIS}	Time from DRV_ENN pin falling 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

Downloaded from Arrow.com.

(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 x 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							

Downloaded from Arrow.com.

(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

Downloaded from Arrow.com.

Package Information

FC2QFN30 6mm x 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	G – 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

Downloaded from Arrow.com.

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
10	REFL	Left reference input for internal ramp generator; Optional use as STEP input.	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)
9	REFR	Right reference input for internal ramp generator; optional use as DIR input.	DI
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)
19	OB2	Motor coil B output 2	A
17	OB1	Motor coil B output 1	A
2	OA2	Motor coil A output 2	A
4	OA1	Motor coil A output 1	A
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	AI
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 (OCSI) 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.	AI
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 (OCSI) analog output for application tuning.	DO / AO

Downloaded from Arrow.com.

Pin Configurations

TMC5262

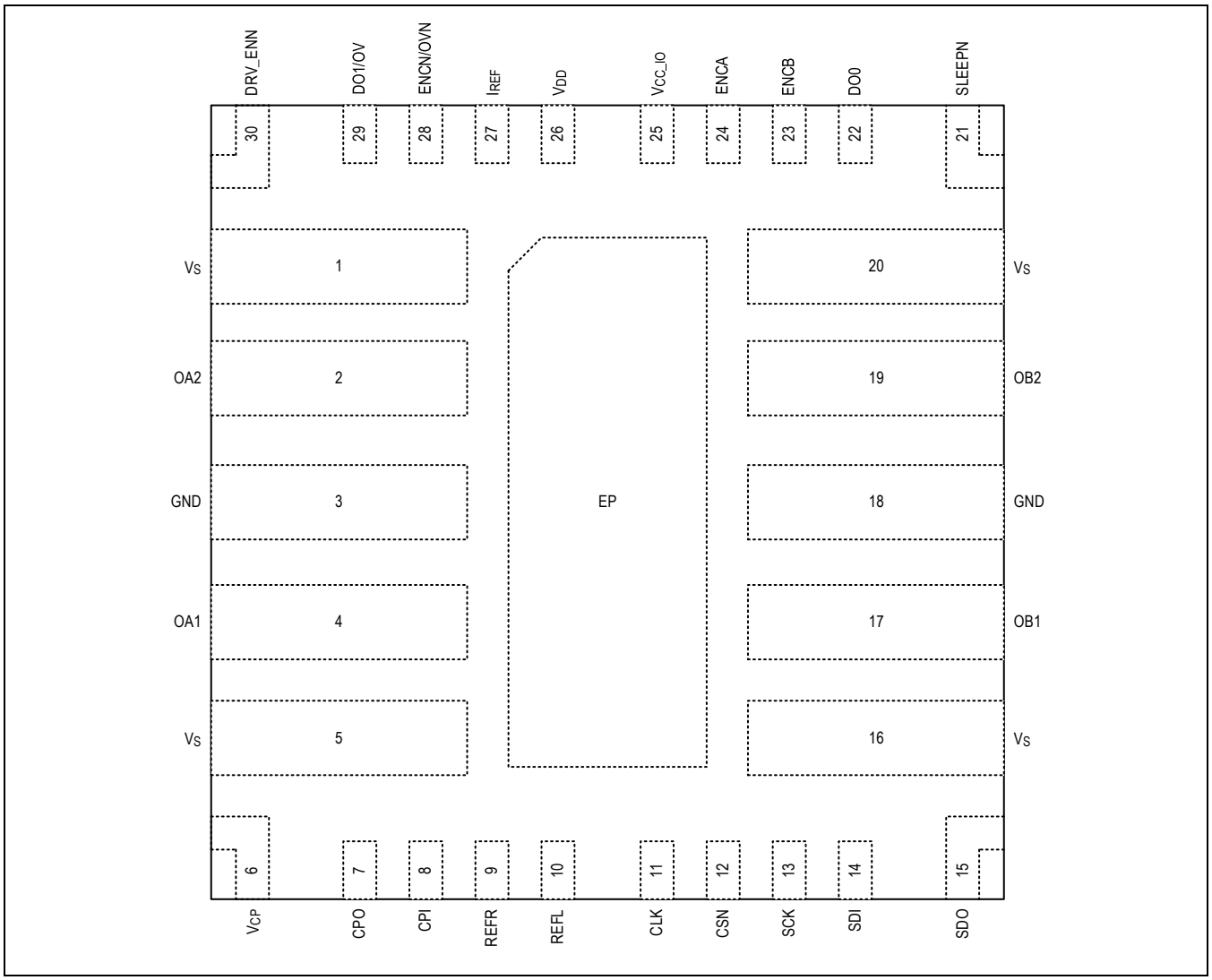


Figure 2. TMC5262 Pinout with Bump Side Down

Downloaded from Arrow.com.

Functional Diagrams

TMC5262

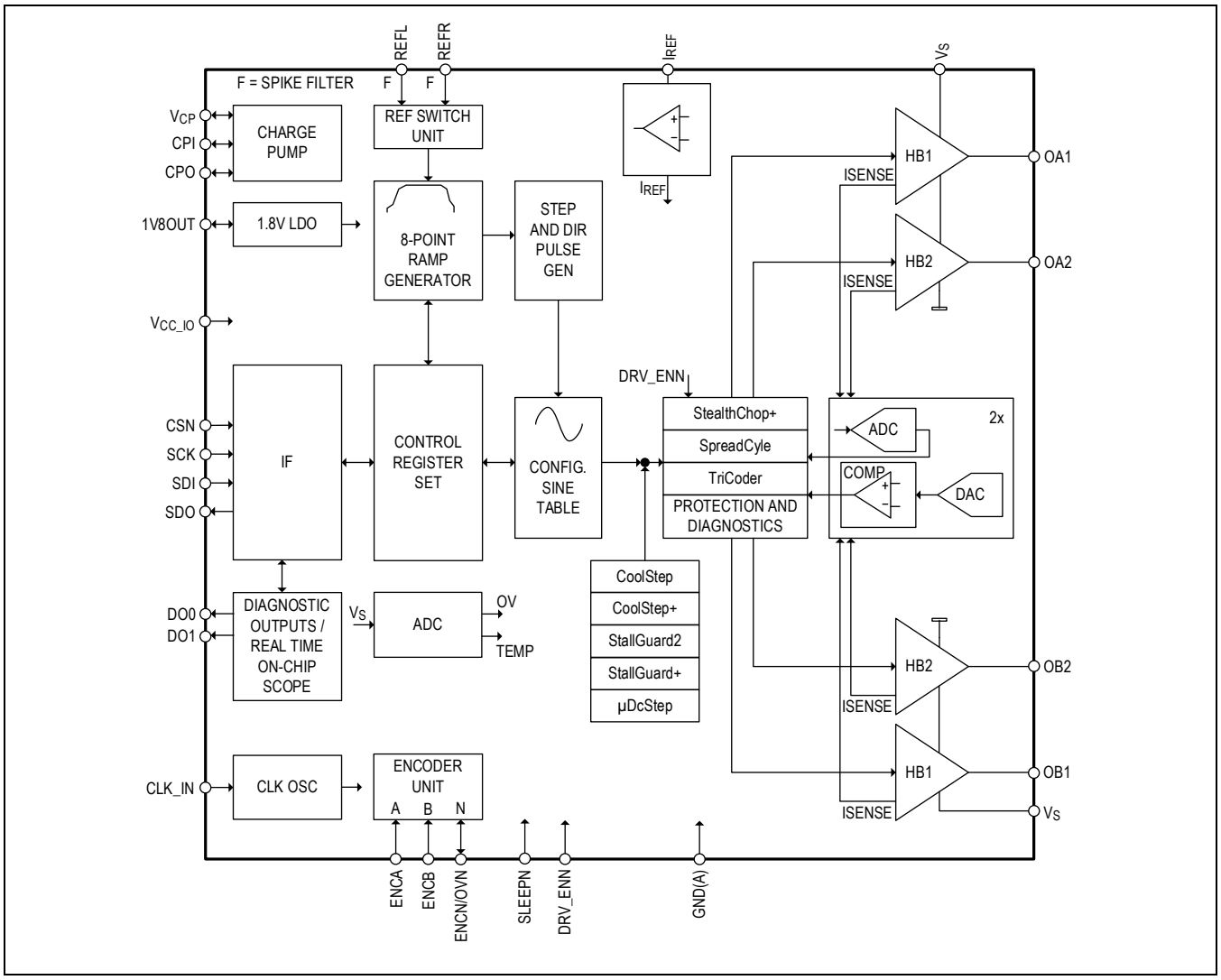


Figure 3. Block Diagram

Downloaded from Arrow.com.

Detailed Description

Principles of Operation

Full Featured Motion Controller and Driver

The TMC5262 is an intelligent stepper motor motion controller and driver chip – a cDriver. This smart power conversion component directly drives a two-phase stepper motor and interfaces to a CPU for configuration, control, and diagnostic feedback. All 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 TMC5262 offers numerous unique functions, which are enabled by the system-on-chip integration of the driver and controller.

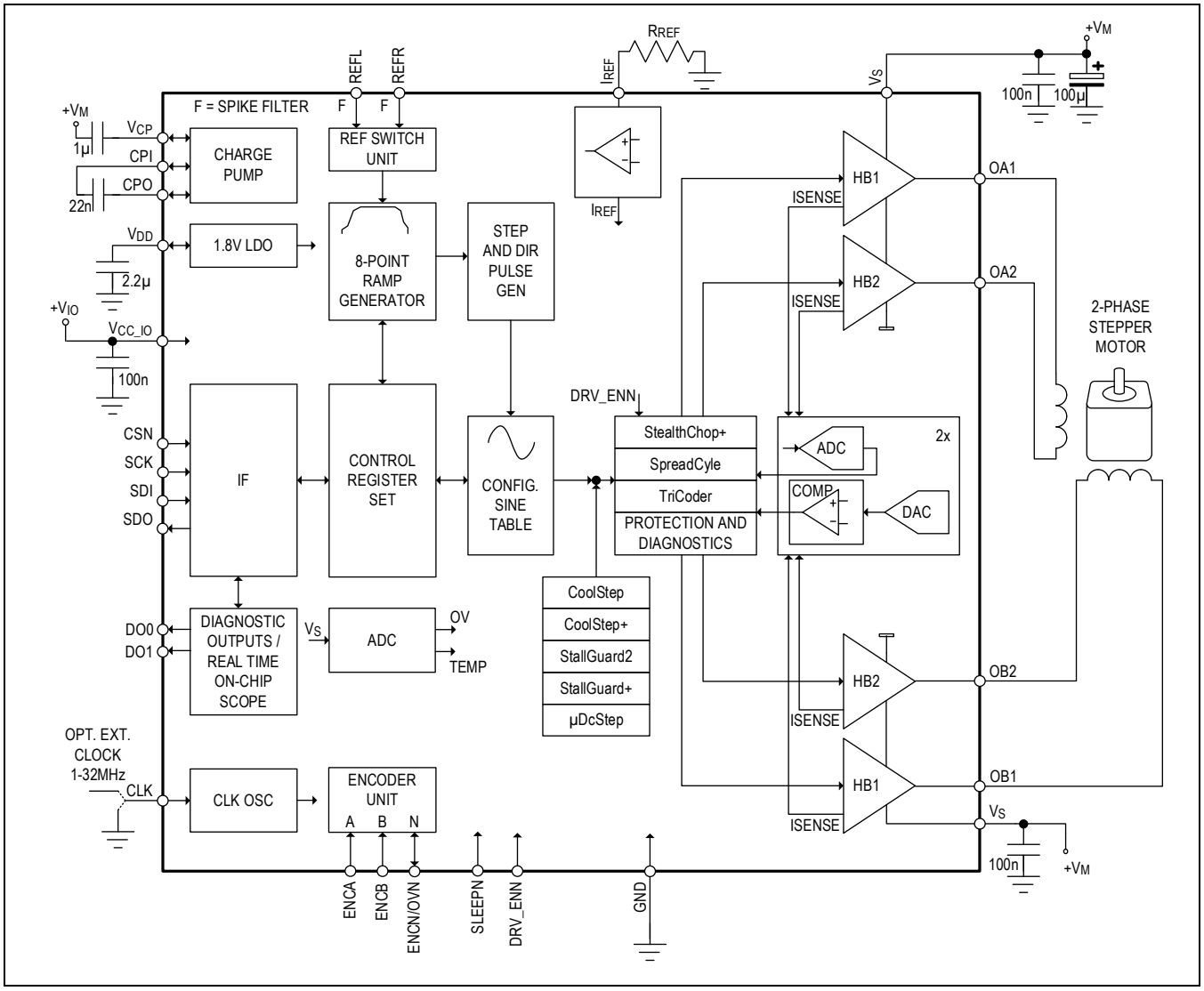


Figure 4. Block Diagram with Typical External Components

Downloaded from Arrow.com.

Key Concepts

The TMC5262 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. TMC5262 Feature Overview

FEATURE	DESCRIPTION
StealthChop+	Silent voltage-controlled motor chopper for smooth motor operation and resonance dampening.
StallGuard+	Superior sensorless stall detection and mechanical load measurement for StealthChop+.
CoolStep+	Uses StallGuard+ to adapt the motor current for best energy efficiency and lowest heating of motor and driver.
µDcStep	CoolStep+-based mode for stepper motor operation with automatic load-dependent velocity adaptation to prevent overload while moving as fast as possible.
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.
TriCoder	Sensorless standstill steploss detection feature making use of the motor back-EMF while the motor is not powered.
Real-Time On-Chip Scope Interface	Live analog two-channel debug option using integrated DACs at the diagnostic outputs.

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 TMC5262 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.

Integrated Eight-Point Motion Controller

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

Benefits

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

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

Downloaded from Arrow.com.

for control over slow interfaces such as optically isolated couplers. On each active edge, the state sampled from the DIR input determines whether to step forward or back. Each step can be a fullstep or a microstep 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.

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.

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

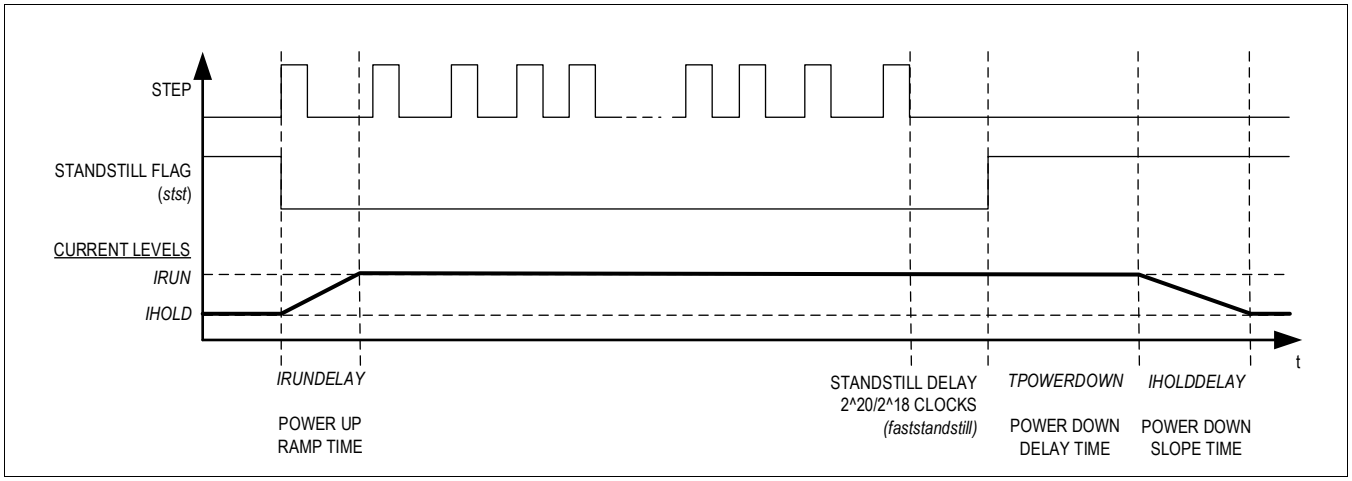


Figure 5. Automatic Motor Current Control at Standstill and Ramp-Up

StealthChop+ and SpreadCycle Driver

StealthChop+ is a voltage-controlled motor chopper principle integrating highly precise sensorless load measurement, resonance dampening, and premium energy efficiency into its concept while providing best position precision and ultra-silent motor operation.

For highest velocity applications going far into the motor's field-weakening range, SpreadCycle is an alternative to StealthChop+. It is an advanced cycle-by-cycle chopper mode. It offers smooth operation and good resonance dampening over a wide range of speed and load. The SpreadCycle chopper scheme automatically integrates and tunes fast decay cycles to guarantee smooth zero-crossing performance.

StealthChop+ and SpreadCycle may even be used in a combined configuration for the best of both worlds. StealthChop+ for no-noise standstill and slow movement, silent, and smooth performance covering the motor's normal velocity range, where its full torque is available. SpreadCycle at high velocity for high dynamics and highest peak velocity at low vibration.

Benefits

- Significantly improved microstepping even with low-cost motors.
- Motor runs smooth and quiet.
- Resonance cancelling, especially for mid-range resonances.
- Absolutely no standby noise.
- Reduced mechanical resonance improves torque output.

Downloaded from Arrow.com.

Mechanical Load Sensing with StallGuard2 and StallGuard+

StallGuard2 and StallGuard+ provide an accurate measurement of the load on the motor. It can be used for stall detection as well as other uses at loads below those that stall the motor, such as CoolStep and CoolStep+ load-adaptive current reduction.

This gives more information on the drive, allowing functions like sensorless homing and diagnostics of the drive mechanics. While StallGuard2 combines with SpreadCycle chopper, StallGuard+ uses a different principle to combine with StealthChop+.

Load Adaptive Current Control with CoolStep and CoolStep+

CoolStep and CoolStep+ drive the motor at the optimum current. They use the StallGuard2 or StallGuard+ load measurement information to adjust the motor current to the minimum amount required in the actual load situation.

CoolStep and CoolStep+ result in energy savings and keep the components cool. Due to driving the motor with the optimum current, CoolStep+ significantly increases motor efficiency compared to the standard operation with approximately 50% torque reserve. CoolStep+ increases efficiency even further.

Benefits

- Highest energy efficiency, power consumption decreased by up to 75% for CoolStep and up to 90% for CoolStep+.
- Motor generates less heat.
- Improved mechanical precision.
- Less or no cooling.
- Improved reliability.
- Use of smaller motor is possible, less torque reserve required.
- Less motor noise due to less excess energy exciting motor resonances.

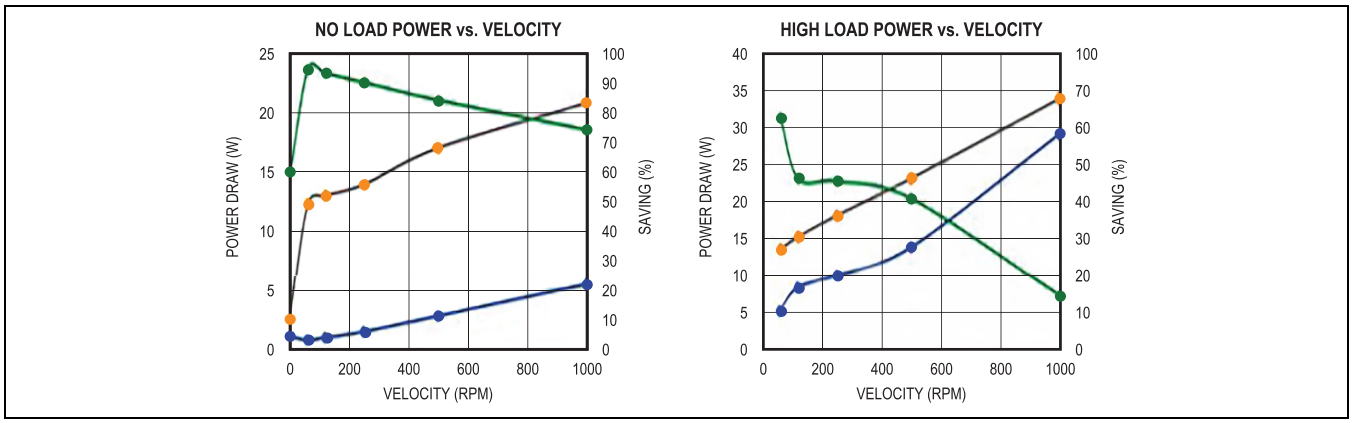


Figure 6. Power Saving with StealthChop+ and CoolStep+ vs. Pure SpreadCycle (No Load and High Load, Motor QSH4218-47-28-040 at 24V Supply)

uDcStep

µDcStep (Micro-DcStep) is a DC-motor like mode of operation to prevent a step loss by allowing a dynamic velocity response and adjustment of the motor to an overload situation, while safeguarding positioning capability.

Real Time On-Chip Scope Interface

One novel function of the TMC5262 is its real time on-chip scope interface (RT-OCSI). Two internal DACs can be mapped to the diagnostic pins and output a broad range of internal registers for tuning support, debugging, and monitoring.

TriCoder

The TriCoder feature is a sensorless standstill steploss detection feature of the TMC5262, which uses the motor back-EMF. It detects position loss during motor standstill while the motor is not powered or allows for teach-in by using the motor as an encoder.

Encoder Interface

The TMC5262 provides an encoder interface for external incremental encoders. The encoder can be used for homing of the motion controller (alternatively to reference switches) with built-in consistency checks between encoder position and

Downloaded from Arrow.com.

ramp generator position. A programmable prescaler allows the adaptation of the encoder resolution to the motor resolution. A 32-bit encoder counter is provided.

Downloaded from Arrow.com.

Serial Peripheral Interface (SPI)

SPI Datagram Structure

The TMC5262 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

<table>
<tr><th colspan="2">MSB (TRANSMITTED FIRST)</th><th colspan="4">40-BIT</th><th colspan="4">LSB (TRANSMITTED LAST)</th></tr>
<tr><td colspan="10">39 ... 0</td></tr>
<tr><td colspan="2">Write: 8-bit address
Read: 8-bit SPI status</td><td colspan="8">Read/write 32-bit data</td></tr>
<tr><td colspan="2">39 ... 32</td><td colspan="8">31 ... 0</td></tr>
<tr><td colspan="2">Write to:
RW + 7-bit address
read from:
8-bit SPI status</td><td colspan="2">8-bit data</td><td colspan="2">8-bit data</td><td colspan="2">8-bit data</td><td colspan="2">8-bit data</td></tr>
<tr><td colspan="2">39 / 38 ... 32</td><td colspan="2">31 ... 24</td><td colspan="2">23 ... 16</td><td colspan="2">15 ... 8</td><td colspan="2">7 ... 0</td></tr>
<tr><td>W</td><td>38...32</td><td>31...28</td><td>27...24</td><td>23...20</td><td>19...16</td><td>15...12</td><td>11...8</td><td>7...4</td><td>3...0</td></tr>
</table>

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 TMC5262, 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 (VACTUAL), the address byte must be set to 0x80 + 0x22 = 0xA2. For read access, the data bit might have any value (-). So, set them 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

Downloaded from Arrow.com.

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 stand still
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 TMC5262 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 TMC5262.

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 7* 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.

Downloaded from Arrow.com.

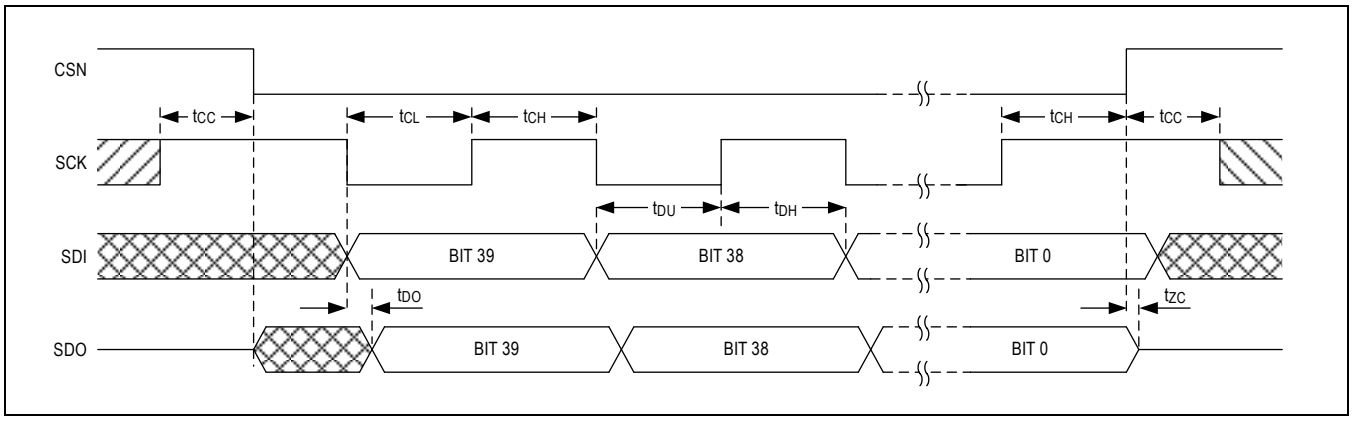


Figure 7. SPI Timing Diagram

Downloaded from Arrow.com.

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.

Enabling the STEP and DIR Interface

The TMC5262 per standard uses its internal motion controller to move the motor. In some applications, it may be desired to either completely work with an external 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.step_dir_enable* allows switching the function of REFL input to STEP function and REFR input to DIR 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 prior to switching to STEP and DIR mode, and vice versa when switching back.

Timing

Figure 8 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.

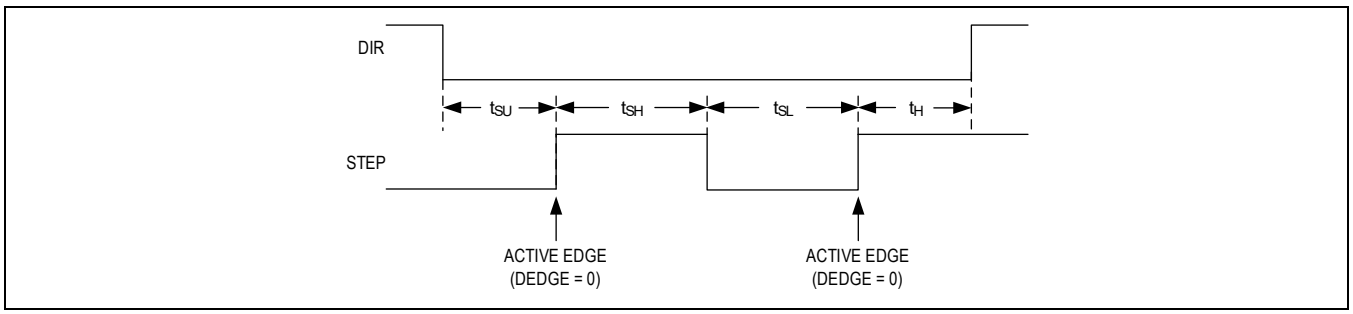


Figure 8. STEP and DIR Signal Timing

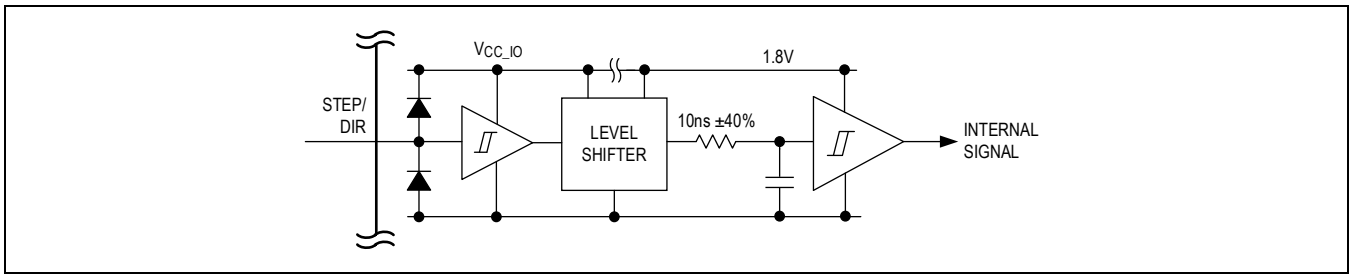


Figure 9. STEP and DIR Signal Input Filter Structure

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 four fullsteps. The microstep resolution setting determines the step width taken within the table. Depending on the DIR input, the microstep counter is increased (DIR = 0) or decreased (DIR = 1) with each STEP pulse by the step width. The microstep resolution determines the increment, 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 for fullstepping. The sequencer has a special provision to allow seamless switching between different microstep rates at any time. When switching to a lower microstep resolution, it calculates the nearest step within the target resolution and reads the current vector at that position. This behavior

Downloaded from Arrow.com.

especially is important for low resolutions like fullstep and half step because any failure in the step sequence leads to asymmetrical run when comparing a motor running clockwise and counterclockwise.

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

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

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

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

Table 6. Fullstep/Half Step Lookup Table Values for Phase A/B Coil Currents

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

MicroPlyer Step Interpolator and Standstill Detection

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

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

The step rate for the interpolated 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 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!

Downloaded from Arrow.com.

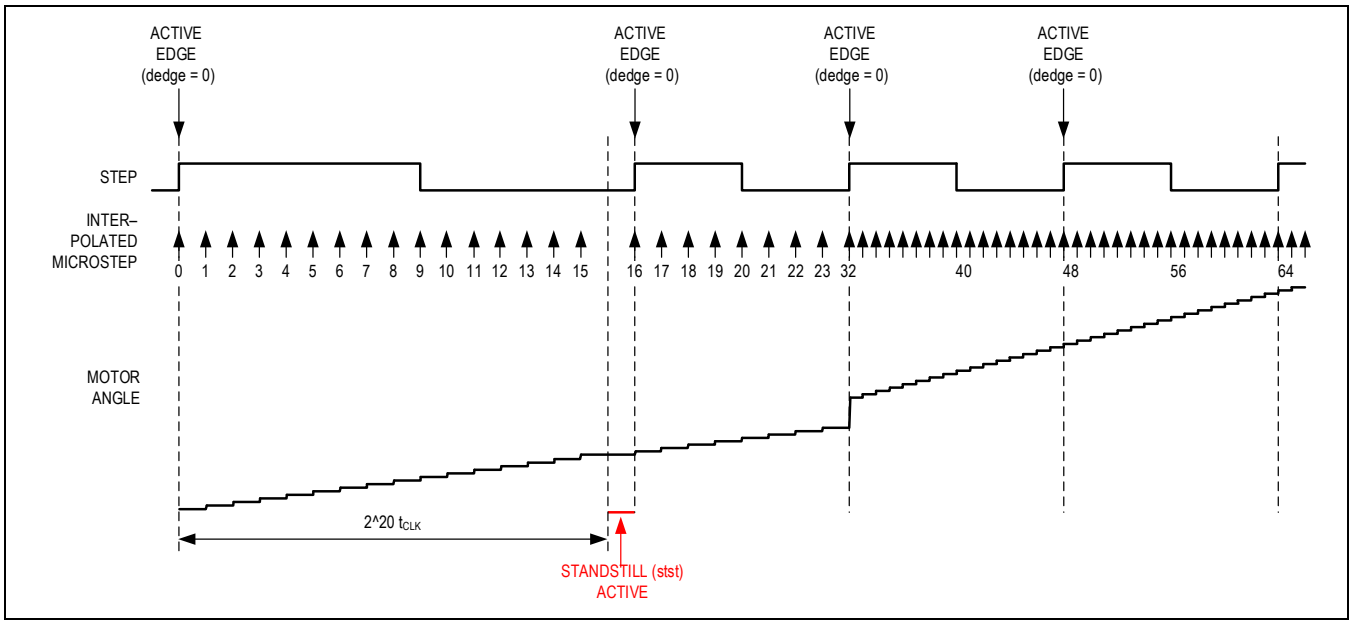


Figure 10. MicroPlyer Microstep Interpolation with Rising STEP Frequency (Example: 16 to 256)

In *Figure 10*, 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.

Downloaded from Arrow.com.

StealthChop+

StealthChop+ is a silent voltage-controlled motor chopper principle integrating highly precise sensorless load measurement, resonance dampening, and premium energy efficiency into its concept, while providing best position precision.

StealthChop+ combines sensorless feedback-based optimization strategies with classical and easy-to-use feed-forward stepper control using ramp generator or Step and Dir input as a stringent set point for the motor position. StealthChop+ automatically determines mechanical load on the motor, adapts the motor current fitting to the mechanical load, and counteracts mechanical resonance by enforcing the electrical angle of the motor current.

The highly efficient driving system allows the use of a smaller, less expensive motor to fit peak torque demand, by keeping the motor cool, which on the other hand gives headroom for overdriving the motor for short times where increased peak power is required. The selection of a smaller motor again reduces relative motor cogging torque and leads to more silent operation.

Deceleration options allow either to efficiently feed motor kinetic energy back to the supply, or to reduce energy feed-back to minimize the demand for overvoltage protection.

Benefits

High efficiency, low noise, low vibration:

- Motor runs smooth and quiet.
- Resonance dampening minimizes mechanical system noise.
- High efficiency allows use of smaller motor.

Reduced system cost:

- Low mean motor current allows overdriving the motor to deliver high peak torque.
- Motor and driver run cool, reducing demand for cooling and supply power.
- Lower resonance allows lighter mechanical system design.

Sensorless load measurement:

- Sensorless load measurement allows sensorless homing and step loss detection.
- Precise load measurement allows predictive maintenance.
- Low velocity stall detection allows use in applications with tiny motion range.

Ease-of-use and reliability:

- Easy and stable, deterministic parameterization using a single parameter set to cover complete motor velocity, current, temperature, and supply voltage range without the need of reparameterization.
- Coil resistance measurement is used for automatic compensation and allows determination of motor temperature.
- Sensorless features provide high dynamic range, typical from a few percent of full velocity to full system velocity (as determined by the motor drive voltage reaching supply voltage level) and even into field-weakening areas.

Downloaded from Arrow.com.

StealthChop+ Principle of Operation

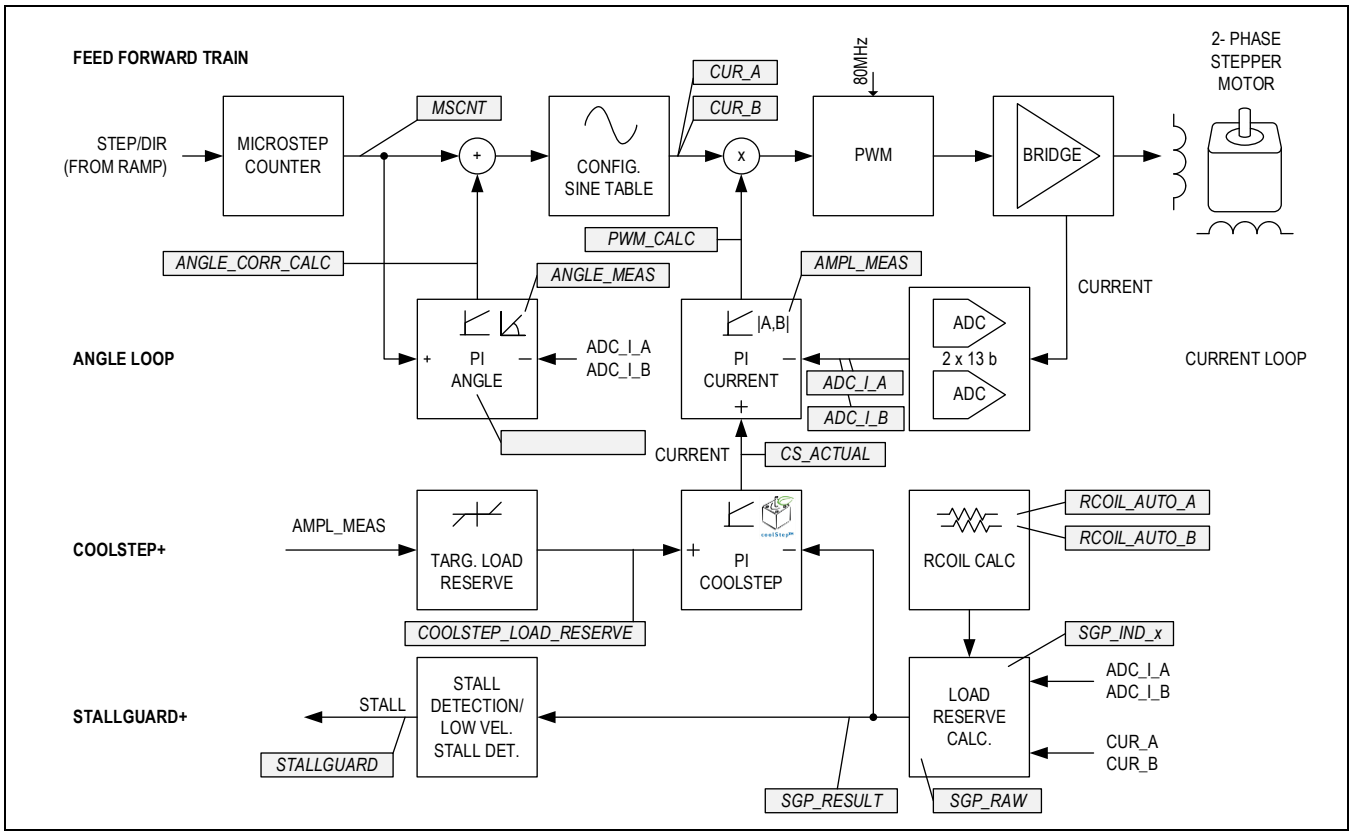


Figure 11. StealthChop+ Regulator Scheme

StealthChop+ uses the classical feed-forward train associated with stepper motor control, to ensure the motor position directly tracks the commanded motion profile.

A current control loop with dual analog-to-digital converters (ADCs) provides perfect motor current tracking, even with fast acceleration and deceleration profiles or upon motor stall.

An angle control loop keeps the motor's stator field aligned to the microstep counter position. This dampens motor oscillations and avoids motor lagging behind the commanded motion by a velocity-dependent phase shift. Further, the angle control loop allows seamless switching to the direct current controlled SpreadCycle chopper.

StallGuard+ keeps track of the mechanical load by calculating a load reserve (a metric describing available unused torque) at the actual motor operation point. It monitors the load reserve to allow sensorless stall detection.

CoolStep+ keeps track of the load reserve and adapts the motor current to fit the actual load demand, while staying in a safe and stable operating point. It allows tremendous energy savings over a classical, pure feed-forward operation.

µDcStep works together with CoolStep+ and is a DC-motor like dynamic adaptation of the motor velocity to prevent step loss. When the motor torque is not sufficient anymore to drive the load, the motor velocity automatically is reduced to a working point with higher torque output due to decreasing back-EMF.

Downloaded from Arrow.com.

Enabling StealthChop+ Procedure

StealthChop+ comes with a set of default parameters, that work with many motors as a starting point. Just a few steps are required to enable StealthChop+.

Table 7. Enabling StealthChop Procedure

NUMBER	TASK	DETAIL
1	Set Motor Current	Configure the motor current settings (*DRVCONF*, *IRUN*) matching the motor's nominal current and set *IHOLD* as desired for standstill torque. Operate in full 256 microstep operation, or with reduced resolution and microstep interpolation enabled. Optionally, optimize the microstep table matching the motor (parameters also can be determined during StealthChop+ operation).
2	Set Motor Coil Inductivity	Enter the coil inductance value as taken from the motor data sheet into *COIL_INDUCT* = <coil inductivity in µH> (example, 8000 for a motor with 8mH coil inductance).
3	Enable StealthChop+	Set *GCONF.en_stealthchop.*
4	Select Velocity Thresholds	Select a lower velocity threshold for CoolStep+ of roughly 0.25 RPS to 1 RPS. For a 1.8° motor, 0.3 RPS roughly is *TCOOLTHRS* = 1000 (1000 clock cycles between two 1/256 steps at 16MHz clock frequency).
5	Select Saving Factor	Select the level of energy saving by programming the current reduction factor *COOL_CUR_DIV* in the range 2 (1/2) to 5 (1/5). (Up to 10 possible using detailed tuning). When selecting *IRUN* and *COOL_CUR_DIV*, consider that scaling down *IRUN* by *COOL_CUR_DIV* results in reduced run currents. The resulting divided current should remain at or above 35 for best results, as the angle regulator per default is disabled for lower currents.
6	Enable Driver	Enable the motor by setting *CHOPCONF.TOFF* = 3 *(*or any value in the range 2 to 15).
7	Initial Motion for Measurement of Coil Resistance	Turn the motor for at least one fullstep at a velocity of maximum 4000 microsteps/second (matching to the default for *T_RCOIL_MEAS* = 4096). This allows the driver to initially measure the motor coil resistance required by StealthChop+. Repeat this step in case *DRV_CONF* settings are modified.
8	Operate Motor within CoolStep Velocity	Accelerate the motor up to >1 RPS, respectively, a velocity of 50000 microsteps/second (or higher) to switch the motor to CoolStep+ operation.

The procedure relies on additional settings not modified here. The default configuration uses conservative values for the best stability of PI regulators. Additional detailed configuration and tuning allows for best motor performance.

Note: Unlike with SpreadCycle, *IRUN* and *IHOLD* values 251 to 255 internally are clipped to 250 to provide sufficient headroom for regulation using the internal AD-converters.

Why Do Detailed Tuning of StealthChop+?

While StealthChop+ comes with workable defaults for many parameters, in certain situations, a tuning of parameters helps to improve the:

- Stability of the motor current and available torque upon load jump.
- Motor efficiency
- Resonance dampening

Downloaded from Arrow.com.

It is important to understand that there is no single “good” setting, but StealthChop+ regulators work fine with a wide range of settings, unless a parameter is close to a functional limit. Therefore, it makes sense to find a valid range for each individual parameter and center the setting in this range.

Configuration of the StealthChop+ Current Regulator

StealthChop+ uses a PI regulator to control the motor current. While current regulation normally works nicely using default values, detailed tuning optimizes current stability. The choice of motor-adapted settings prevents torque reduction due to undershooting upon fast motor acceleration or increased supply current consumption during deceleration due to overshooting of motor current.

The current regulator works based on the actual run current *CS_ACTUAL* (as controlled by *IRUN*, respectively, *IRUN* scaled by *COOL_CUR_DIV*). This value is multiplied by eight and the result is compared to the momentary RMS current amplitude calculated from the ADC converter measurement of both phase currents (*AMPL_MEAS*). A measured amplitude of 2000 corresponds to the upper StealtChop+ *IRUN* value of 250.

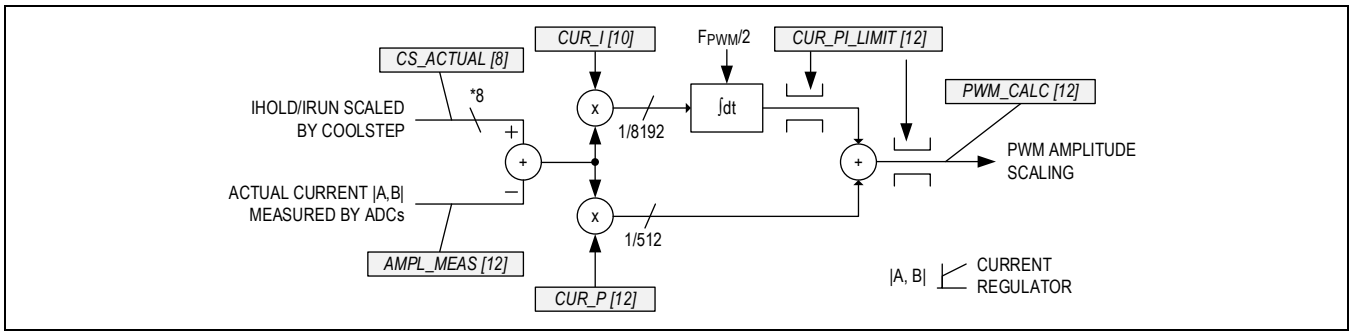


Figure 12. StealthChop+ Current Regulator

The target of the current regulator parameter tuning is to reach a stable regulation with a sufficiently fast response of the motor driver PWM amplitude to the changes in motor velocity, especially upon maximum desired acceleration and deceleration.

Use the desired operating voltage for tuning. In case the voltage can change significantly, check for stability at the maximum desired operating voltage, as the same change in amplitude results in a higher change in motor current at a higher supply voltage. Increase the P-factor *CUR_P* to speed up current step response time. If oscillation occurs, reduce the P factor to half of the critical value or lower. The P-part can, but not necessarily has to, make up for 50% to 95% of the current regulation. The I-factor *CUR_I* is responsible for precisely regulating to the set point in a longer time scale.

A setting of *CUR_P* = 512 corresponds to a factor of 1.0 between the current range and supply voltage range, that is, a 1% deviation of motor current leads to a 1% adaptation of PWM duty cycle. At *CUR_P* = 512, the driver scales up the PWM amplitude to its maximum (96% duty cycle at sine wave peak), when it measures an actual current of 0. The internal PWM duty cycle limit of 96% (using the default sine wave) corresponds to 68% of the supply voltage provided as RMS voltage for the motor (because RMS amplitude and peak for a sine wave differ by a factor of 1/SQRT(2)).

Downloaded from Arrow.com.

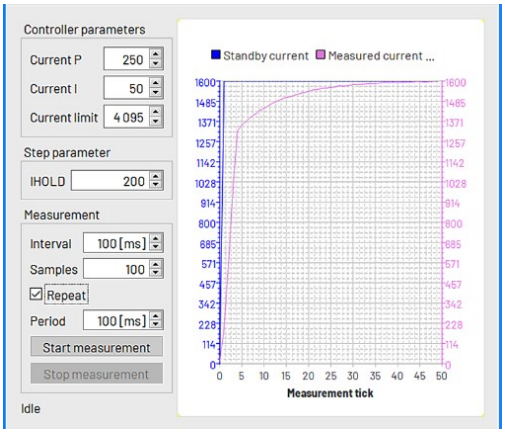


Figure 13. Tuning the Current Regulator by Monitoring the Current Step Response When Switching from 0 to a Higher IHOLD Current.

Current Regulator P Tuning

A first stable setting for *CUR_P* can be calculated based on the motor coil resistance and selected current range. This principle is shown here. However, the optimum setting for an application can be higher to improve response time, or it can be lower because a lower setting reduces regulation ripple caused by measurement noise:

Calculate a 50% P-part regulation as starting value for *CUR_P* optimization:

$$CUR_P_{50} = 512 \times \frac{R_{COIL}[\Omega]}{0.68\ VS[V]} \times CurrentRange[A]RMS$$

where:

- CurrentRange corresponds to the configured current RMS full-scale setting ($\frac{peak\ current}{\sqrt{2}} \approx 0.71 \times peak\ current$) as selected by *CURRENT_RANGE* and *CURRENT_RANGE_SCALE* in register *DRV_CONF*.
- V_S is the motor driver supply voltage (nominal value in application).
- The factor 0.68 results from the microstep sine wave peak of $\frac{248}{256} \times 0.71$.
- R_{COIL} is the nominal motor coil resistance, either taken from the motor data sheet, or measured by R_COIL_AUTO and converted to Ω.

With very fast acceleration and deceleration, the application can use CUR_P_{50}, or up to 5…20 times this value to make up for 80%...95% target current regulation using the P-value. A too high setting causes instability and regulation-noise.

With low or moderate motor acceleration or deceleration, a *CUR_P* lower value in the range of 20% to 100% of CUR_P_{50} may be preferable, as it reduces the influence of the ADC measurement noise. Tune the I-part to make up for most of the regulation.

Current Regulator I Tuning

After determining *CUR_P*, the integral part *CUR_I* must be chosen to get a stable current regulation, far away from oscillations. In case a small P-value is used, a high I-value helps to improve the response of current regulation for high motor acceleration or deceleration.

A good starting value is 25% of the numeric value chosen for the *CUR_P*. When *CUR_P* is chosen to cover most of the current difference (example, 80% or more), optimize *CUR_I* in the range of 10% to 100% of *CUR_P*. When *CUR_P* is chosen with a low setting (example, less than two times the calculated CUR_P_{50}), increase *CUR_I* to take over most of the regulation difference. In this case, set *CUR_I* to up to five times the actual *CUR_P* setting. Consider different numerical ranges and upper limits for *CUR_P* and *CUR_I*.

Downloaded from Arrow.com.

Monitor the current regulation for stability when accelerating or decelerating the motor, or when changing the target current setting using *IRUN* or *IHOLD*. When the current drops more than a few percent during an acceleration phase, increase *CUR_I* to get a faster response. In case oscillation occurs, reduce it to 70% of its critical value or less.

Under special circumstances, the resulting duty cycle can be limited using the *CUR_PI_LIMIT* setting, for example, during experiments that might lead to current ratings exceeding the motor's rating. In this case, set *CUR_PI_LIMIT* to yield the desired maximum fraction of the supply voltage at the motor coils, that is, a setting of 409 limits duty cycle to 10%, and thus, the effective voltage is a maximum 10% of the IC supply voltage. For normal, leave the limit set to the full scale of 4095.

To avoid triggering overcurrent, the regulator I-part automatically is cleared upon chip enable, execution of a hard stop triggered by StallGuard, or by a stop switch or virtual stop event.

Note: For most cases, the choice of the current regulator PI coefficients is uncritical, and a wide range of parameters work. The application of high acceleration (motor acceleration within a few milliseconds to a velocity where significant back-EMF occurs) requires increased PI current regulator coefficients to avoid torque drop resulting from a temporary drop in current. In this case, a tuning of coefficients is mandatory.

An overall low parameter set reduces the IC trying to counteract current distortion caused by a non-sinusoidal-form of the motor's back-EMF. This can affect smoothness with high torque motors or permanent magnet motors in certain velocity ranges. Try out the low parameter setting vs. regular parameter setting.

Rule of Thumb for Current Regulator PI Tuning

$$CUR_P_{50} = 512 \times \frac{R_{COIL}[\Omega]}{0.68\ VS[V]} \times CurrentRange[A\ RMS]$$

Choose *CUR_P*:

For low accelerations, $CUR_P = 20\% \ldots 100\%$ of CUR_P_{50}.

For high accelerations, $CUR_P = 2 \ldots 20 \times CUR_P_{50}$.

Choose *CUR_I*:

A reasonable optimization range is $CUR_I = 10\%\ CUR_P$ to MIN. $(5 \times\ CUR_P_{50};\ 5 \times CUR_P)$:

With $CUR_P\ <\ 2 \times CUR_P_{50}$, choose *CUR_I* in the range of $10\% \ldots 500\%\ CUR_P$.

With $CUR_P\ >\ 2 \times CUR_P_{50}$, choose *CUR_I* in the range of $10\% \ldots 100\%\ CUR_P$.

Example for Current Regulator PI Tuning

A motor with 5Ω coil resistance is operated at 24V in the $1A_{RMS}$ current range.

$$CUR_P_{50} = 512 \times \frac{5\Omega}{0.68 \times 24V} \times 1A = 157$$

With this setting, the P part in the first step compensates any current deviation 1:1, and therefore, regulates to roughly 50% of the target current. This setting is sufficient for many cases.

Possible choices for the example motor:

Slow, low noise current regulation: $CUR_P = ½\ CUR_P_{50} \approx 75;\ CUR_I = CUR_P = 75$

Fast current regulation, using I-part: $CUR_P = CUR_P_{50} = 157;\ CUR_I = 2 \times CUR_P_{50} = 314$

Very fast current/velocity step response: $CUR_P = 10 \times 157 = 1570;\ CUR_I = 0{,}25 \times 1570 = 392$

Downloaded from Arrow.com.

CUR_P and CUR_I Tuning Using a Scope

A scope with current probes effectively shows the operation of the current PI regulator. An iterative approach can be used to determine *CUR_P* and *CUR_I*. The sample plots the resulting coil currents for a NEMA 17 1A motor operated at 700mA RMS/1A peak. Starting from an initial setting, parameters are improved stepwise. The motor quickly accelerates from standstill (with current reduced to a low value) to 120RPM (AMAX = DMAX = 63000) and quickly decelerates again after 15000 microsteps (~58 fullsteps).

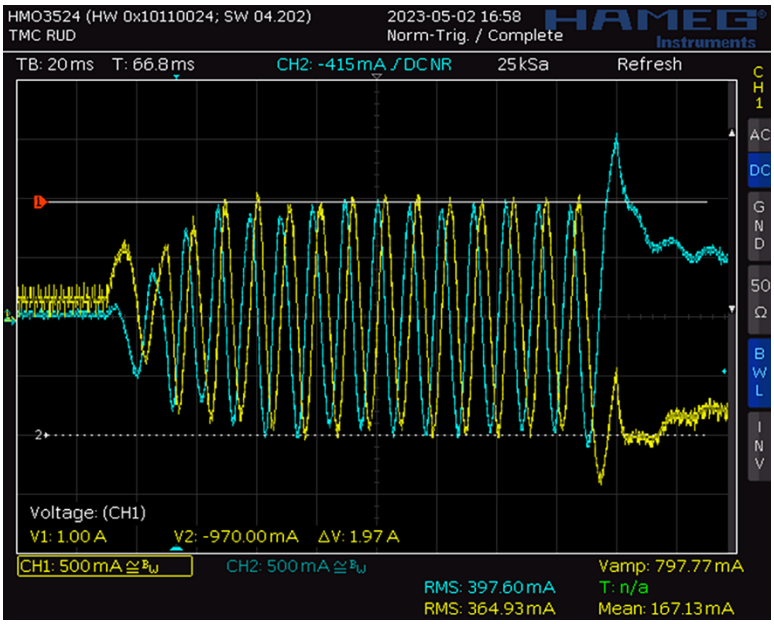


Figure 14. Too Low CUR_P (50% CUR_P$_{50}$) and CUR_I (25% CUR_P$_{50}$): Slow Current Ramp Up, Overshoot Upon Deceleration

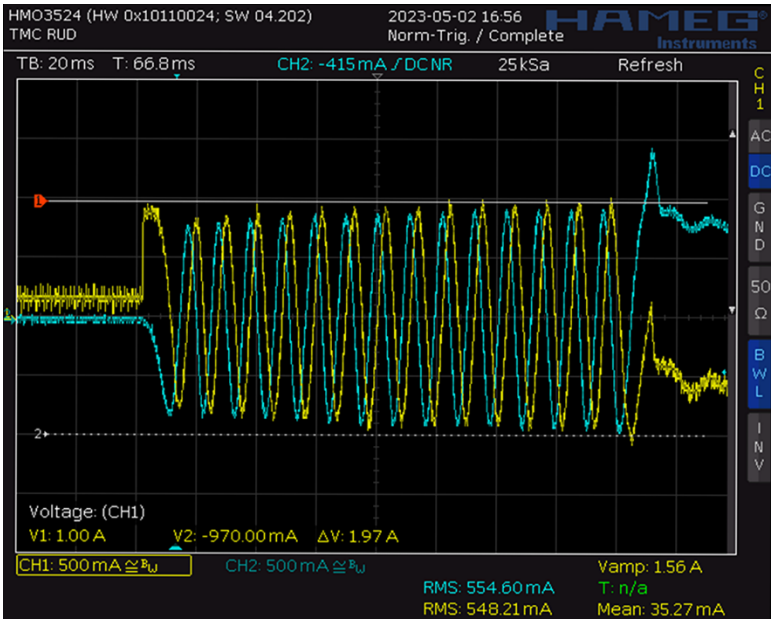


Figure 15. High CUR_P (10 × CUR_P$_{50}$) Gives Fast Response at Start and Stop. Remaining Difference Only Slowly Compensated Due to Low CUR_I (10% of CUR_P)

Downloaded from Arrow.com.

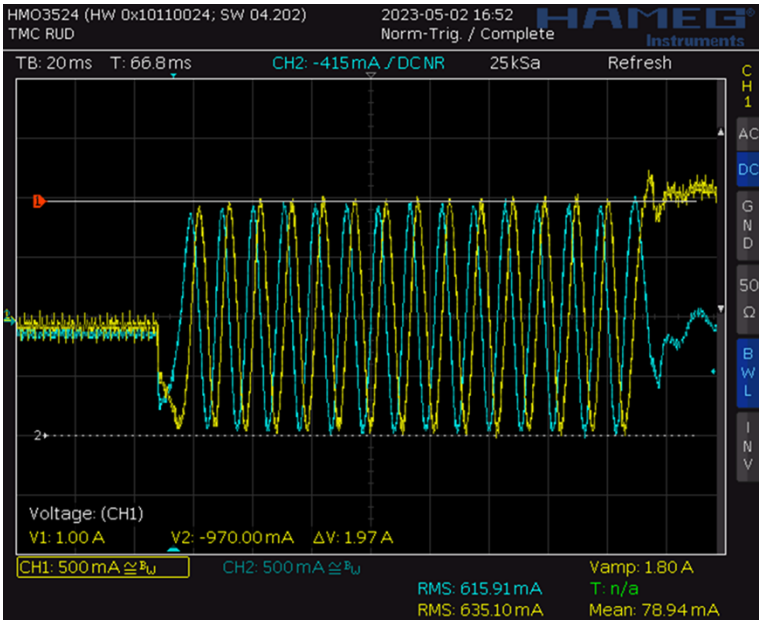


Figure 16. High CUR_P (10 × CUR_P$_{50}$) with Increased CUR_I (25% of CUR_P) Yields Quick Current Regulation to the Set Point.

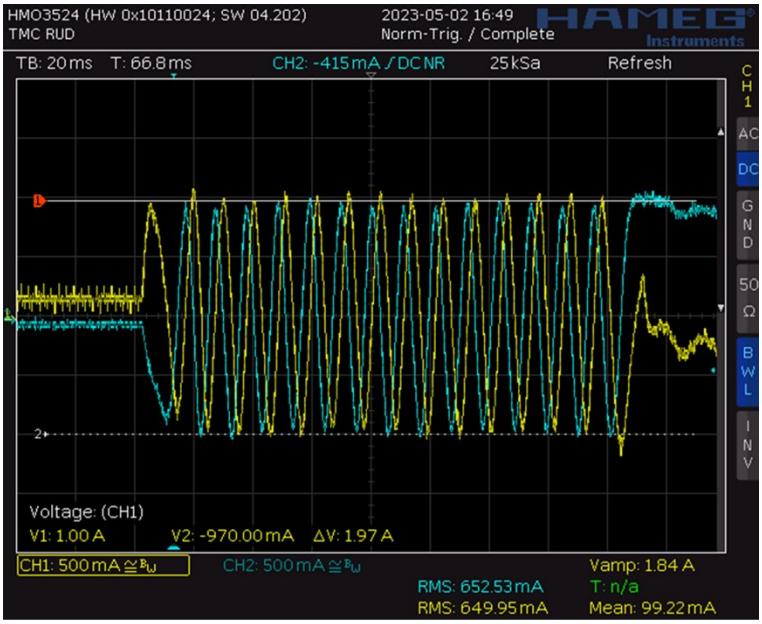


Figure 17. CUR_P = CUR_P$_{50}$ with Increased CUR_I (2 × CUR_P$_{50}$) also Yields Quick Current Regulation to the Set Point with Minor Over- and Undershoot.

Configuring StealthChop+ Angle Regulator

StealthChop+ uses a second PI regulator to control the motor current angle. The angle regulation aligns the current angle to the angle commanded from the microstep counter. The angle control eliminates rotor lagging due to the electrical field phase shift and hinders the rotor field from oscillating around the commanded set point. It is a part of the integrated resonance dampening scheme. The default parameters typically work for aligning the current angle and sufficiently avoiding low frequent oscillation.

Downloaded from Arrow.com.

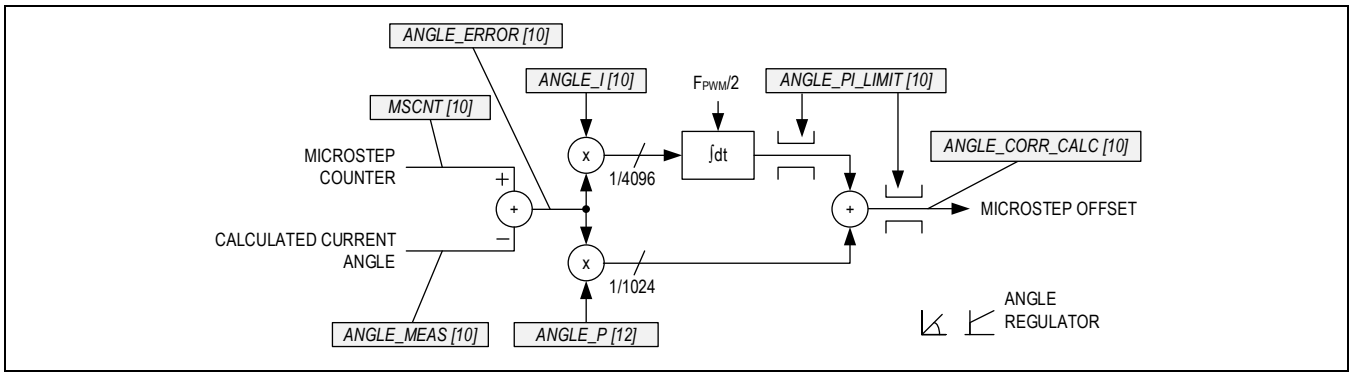


Figure 18. StealthChop+ Angle Regulator

The target of the angle regulator parameter tuning is to reach a stable regulation with a sufficiently fast response of the regulator to changes in the phase shift between the motor coil voltage and coil current caused by a change of velocity, mechanical load, or motor resonance.

Application Range of the Angle Regulator

The angle regulator is activated whenever the velocity exceeds the threshold set by *T_RCOIL_MEAS*. This threshold should be set at a very low velocity, for example, a few RPMs, where the motor mainly shows resistive behavior with negligible effects of BACK-EMF, and thus, negligible angle deviation.

A lower current limit for angle regulation is specified by *ANGLE_LOWER_I_LIMIT*. This allows disabling the angle correction with very low motor currents, as the angle calculation might be imprecise, and thus, lead to jitter when the motor current is too low to precisely determine its electrical angle. When *AMPL_MEAS* falls below *ANGLE_LOWER_I_LIMIT*, the angle regulation is disabled and the microstep offset is frozen to the last valid value. As an *IRUN* of 250 corresponds to an *AMPL_MEAS* of 2000, the default value of *ANGLE_LOWER_I_LIMIT* = 256 corresponds to an *IRUN* of 33. Keep this default value, unless a very high *COOL_CUR_DIV* setting is desired, scaling down *IRUN* to less than roughly 35. The scale is identical to *PWM_MEAS* in ADC resolution.

Due to coil inductance and BACK-EMF, the motor current lags motor voltage by up to 90°. The default setting of 256 for *ANGLE_PI_LIMIT* allows up to 90° of angle correction, which is the largest expected deviation. *ANGLE_CORR_CALC* is a signed value showing the actual correction value calculated.

Depending on the optimization goal, tuning can be executed at a resonance frequency in the application velocity range (this might be a multiple of the first resonance that typically occurs at 50Hz to 150Hz full step frequency with an unloaded motor), or at a high or maximum intended StealthChop+ motor velocity. As the motor current setting as well as its mechanical load shifts resonance frequency, tuning should be done in a realistic application setup for the best results.

Angle P Tuning

Trace *ANGLE_ERROR* showing the actual angle deviation and angle jitter. Check for best effect (lowest motor vibration, most silent operation) and stability using the intended mechanical setup. Increase the P-factor *ANGLE_P* to speed up the load step response time. If oscillation occurs, reduce the P factor to half of the critical value or lower. The I-factor *ANGLE_I* is responsible for precisely regulating to the set point in a longer time scale. The overall response time can be trimmed to a fraction of the mechanical oscillation period, typically in the range of 10ms.

Start with a low *ANGLE_P* setting (for example, 50) and a low *ANGLE_I* setting (for example, 10), and increase *ANGLE_P* until *ANGLE_ERROR* signal amplitude narrows down and does not significantly improve with higher settings. A setting of 1024 for *ANGLE_P* corresponds to a 50% correction of angle error by P-part. High settings up to 3000 can help to dampen the mid-range resonance of motors. As a thumb rule, a setting in the range of 50 to 250 is the most universal. The use of lower values avoids angle calculation noise (resulting from ADC measurement of the motor currents), leading to motor vibration in the low velocity range.

Downloaded from Arrow.com.

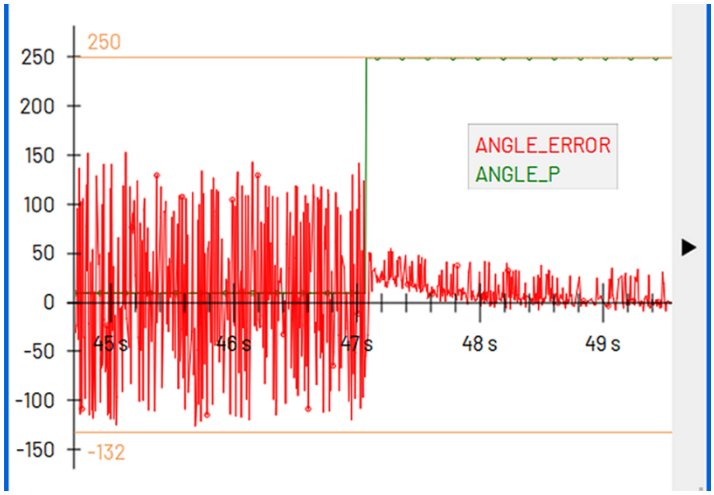


Figure 19. Motor in Mid-Range Resonance (Close to Point of Stalling) Resolved by Increasing ANGLE_P to 250

Angle I Tuning

A setting of 10 to 100 already sufficiently responds to angle errors caused by changing the motor load. Higher settings easily can help to address mid-range resonance but also can cause regulation instability, unless a high *ANGLE_P* is used. Check for stability at the highest application velocity. When running at a high acceleration, a higher *ANGLE_I* improves response time and avoids angle jerks when starting/stopping or reversing the motor, or switching to SpreadCycle operation. Optionally, to a high *ANGLE_P*, the combination of moderate *ANGLE_P* with increased *ANGLE_I* can help to dampen the mid-range resonance.

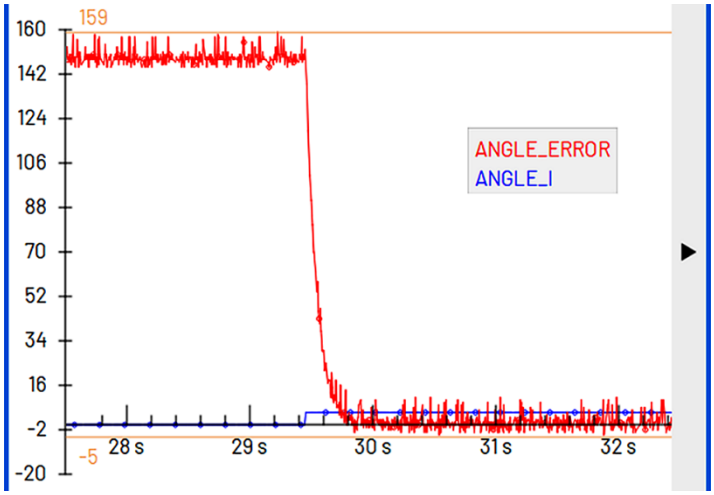


Figure 20. Switching on ANGLE_I from 0 to Any Value Starts Compensating the ANGLE_ERROR to +-0.

Downloaded from Arrow.com.

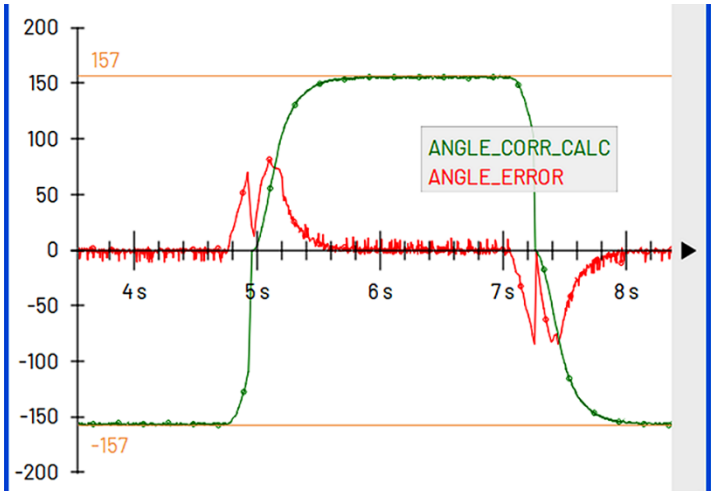


Figure 21. Motor Direction of Rotation Reversed Twice. Reversal with Too Low ANGLE_I Leads to Phase Jerk due to Slow Adaptation. The Dip Shows ANGLE_ERROR Cleared to 0 at a Velocity Below the Limit Given by T_RCOIL_MEAS.

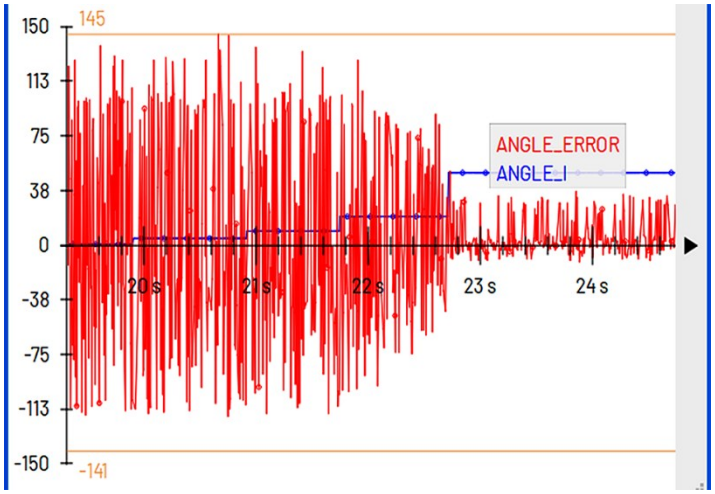


Figure 22. Increasing ANGLE_I from a Low Value up to 50 to Dampen Mid-Range Resonance

Downloaded from Arrow.com.

Rule of Thumb for Angle PI

ANGLE_P = 1024 directly compensates 50% of the angle error. While this gives a fast response, it can lead to harsh motor behavior in some cases.

Choose *ANGLE_P*:

Standard case: Use a value in the range 50 to 250 and optimize for lowest vibration.

Mid-range resonance optimization: Try a setting of 250 up to 3000. Carefully check for stability.

Choose *ANGLE_I*:

Standard case: Set in the range 25 to 250 and carefully check for stability in case of high setting.

Mid-range resonance optimization: Try a setting of 100 to up to the same value as *ANGLE_P*.

Options for the StealthChop+ Chopper

The StealthChop+ chopper offers a handful of additional advanced options.

Chopper Frequency

The chopper frequency can be controlled in eight steps from 20kHz to 60kHz using the PWM_FREQ setting. Step width is roughly 5kHz. As 20kHz already is outside the audible range, it normally is sufficient to operate at this frequency. A higher frequency can be beneficial in case of very low inductive, tiny motors, as it allows reducing current ripple caused by the low inductance. On the other hand, choice of a higher frequency slightly increases dynamic switching losses in the driver due to more switching events. With increased switching frequency, the driver slope control *SLOPE_CONTROL* should be set to the highest value.

SD_ON_MEAS_LO and *SD_ON_MEAS_HI* control the timing of the current measurement by setting a threshold on the chopper duty cycle. Typically, current is measured when both motor coil connections are at GND level, in the so-called slow decay phase. This ensures stable measurement down to very low duty cycles. As the available measurement time becomes lower with increased duty cycle, the driver must switch over to measurement during the motor coil ON state, once the available time falls below the time required for blanking of the current measurement (matching the *TBL* setting in SpreadCycle). This switchover typically only happens at fast motor rotation, where the PWM duty cycle approaches its limit in the sine wave peaks. A high duty cycle threshold (default) is best, as it avoids switching the measurement point of time in low velocity operation, where the best microstep precision is desired. The duty cycle thresholds additionally should be adapted to match higher chopper frequency settings, as a higher frequency reduces measurement time. For cases where the motor current shows a longer excursion or large ripple following each switching event, the motor operation at a PWM duty cycle (*PWM_CALC*) near 4095 may show increased angular ripple, that is, the StallGuard+ signal becomes more instable. This situation can occur with the LC filtering of motor connections and long motor cables. To improve, reduce *SD_ON_MEAS_LO* and *SD_ON_MEAS_HI* by one or two.

Table 8. Selection Options for PWM Frequency

PWM_FREQ	FREQUENCY [kHz]	*SD_ON_MEAS_LO*	*SD_ON_MEAS_HI*
0	19.5	14	15
1	24.4	14	15
2	29.3	14	15
3	34.2	13	14
4	39.1	13	14
5	44.0	13	14
6	48.8	12	13
7	53.7	12	13
8	58.6	12	13

Downloaded from Arrow.com.

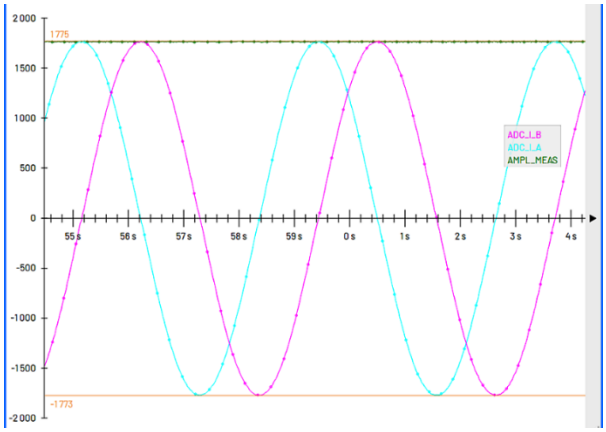


Figure 23. Trace of Motor Current Using ADC Results at Low Velocity in StealthChop+ to Check for Undisturbed Current Measurement

Freewheeling and Passive Braking Modes

A choice of different modes for motor standby is available when setting the motor current to 0 for standstill (*IHOLD* = 0). The default mode (*FREEWHEEL* = 0, normal operation) leaves the current controller on, imprinting minimum current into the coil. To achieve true freewheeling, the driver completely switches off all motor driver bridges (*FREEWHEEL* = 1, freewheeling). The alternatives are passive braking. Passive braking brakes the motor by shorting out its back-EMF using either the driver's low-side or high-side FETs (*FREEWHEEL* = 2 or 3, passive braking low-side or high-side). Freewheeling modes 1 or 2 also can be entered to hot-plug the motor.

Downloaded from Arrow.com.

StallGuard+

StallGuard+ is a sensorless system that precisely measures and monitors actual motor load. Its main output value *SGP_RESULT* is a measure for the motor's load reserve and shows up to what extent the available torque is used. *SGP_RESULT* also is the base for the motor efficiency system CoolStep+. Therefore, a proper configuration is mandatory to take advantage of all benefits.

StallGuard+ mainly uses information available from the measurement of the motor parameters during operation. To precisely determine the motor load, it also must know a few characteristic values of the motor: coil resistance, as well as coil inductivity.

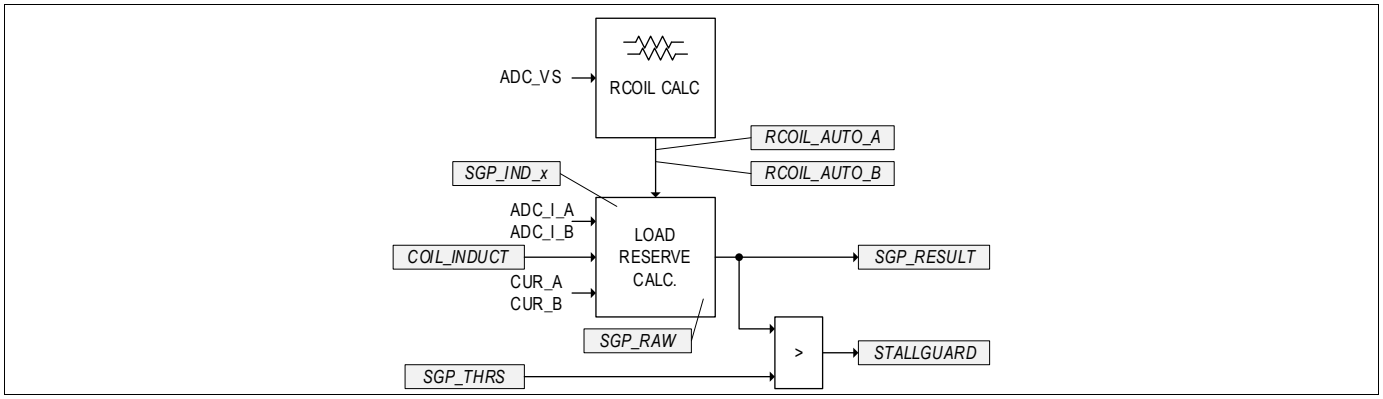


Figure 24. Structure and Basic Parameters of StallGuard+ Load Reserve Calculation

Understanding *SGP_RESULT*

StallGuard+ gives a four-quadrant information describing the actual motor load situation by the load reserve value (*SGP_RESULT*). It determines how much of the available motor torque is used up by the application currently. In stable operating areas, a (small) increase of the motor load leads to an increment of the motor torque turning right the load angle vector (see *Figure 25*) and *SGP_RESULT* sinks. In an overload area, a further increase of the motor load leads to reduced available torque and a step loss of the motor becomes very probable. Near to a load reserve value of 0, there is only minor gain in the motor torque when the load slightly changes, as the torque curve goes with a sine wave function of the load angle repeating each four fullsteps (see *Figure 30*). Due to this, the area near 0 (respectively, near 511 in generatoric operation) is critical, and the motor can easily get into the instable area. Further, a non-perfect parameterization, or a motor parameter stray leads to a certain percentage of uncertainty of the determined load reserve value. Therefore, the motor operating point should be configured to work well outside the uncertain area. However, a motor stall can be safely detected without sacrificing any bit of torque, as the (unfiltered) StallGuard+ value crosses the 0-line in each case upon overload.

Downloaded from Arrow.com.

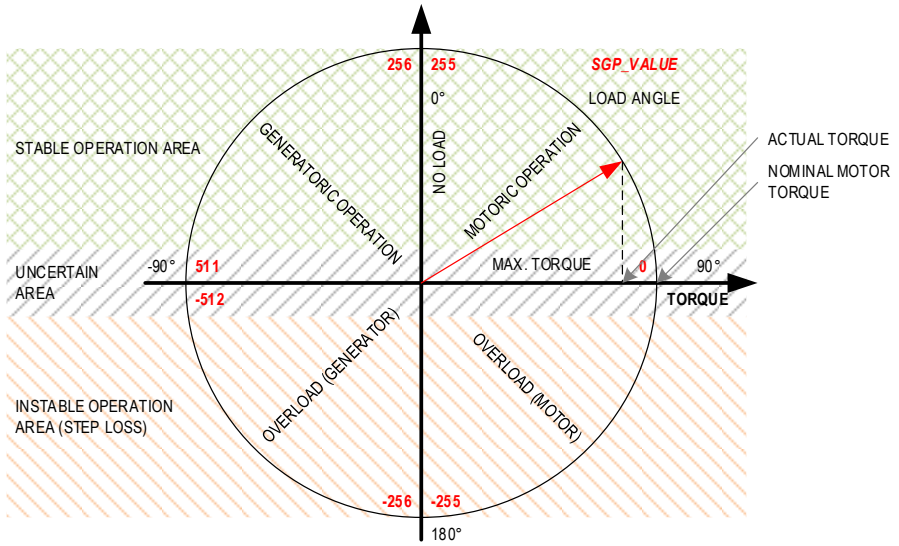


Figure 25. StallGuard+ Value in Different Quadrants of Motor Operation

Figure 25 shows the StallGuard+ value in different quadrants regarding the motor operation. The example vector shows the relative used torque by projecting it to the TORQUE axis corresponding to the *SGP_VALUE* that points to a certain position on the circle. The diameter of the circle varies with motor current. The percentage of the uncertain area depends on the correct configuration, motor current, and motor velocity.

Stall Detection

Stall detection can be used to either detect a stall event, or to prevent the motor from stalling by detecting an overload. Define a load limit using *SGP_THRS.* The StallGuard+ system activates the *stallguard* signal when *SGP_RESULT* falls below the programmed *SGP_THRS.* The motion controller can react to this condition by stopping the motor. It is available as an input flag *RAMP_STAT.status_sg* for the ramp generator stop switch status. Stall detection using the *stallguard* signal is enabled within the CoolStep velocity range above the velocity defined by the threshold *VCOOLTHRS* and up to the upper velocity defined by *TPWMTHRS*. While *THIGH* marks the upper velocity range where CoolStep+ is available, it does not disable stall detection.

Stalling the motor without stopping the ramp generator - an actual stall event leads to (unfiltered) StallGuard+ jitter jumping within the available range.

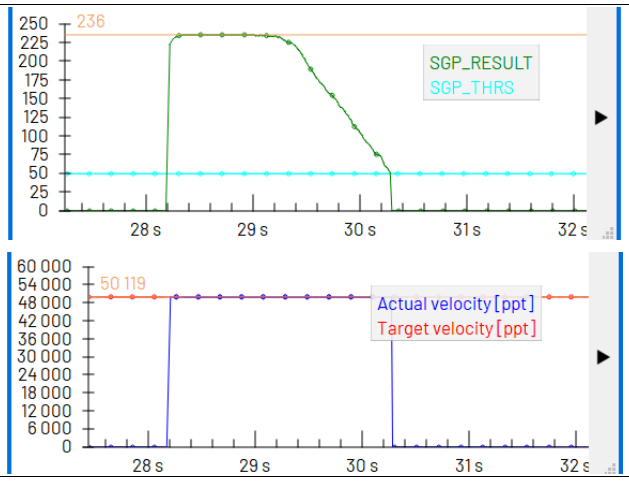


Figure 26. Using SGP_THRS for Stopping the Motor Upon High Load (Plotting Velocity VACTUAL and SGP_RESULT)

Downloaded from Arrow.com.

Low Velocity Stall Detection

Like StallGuard2, StallGuard+ relies on absolute load values for stall detection. With a fixed threshold for stall detection, the relationship between the level and the mechanical load must remain stable. But, at very low motor velocity, the absolute level can float because of the motor heating up or cooling down, and the measurement becomes instable due to a comparatively low back-EMF amplitude being available from the motor.

Low velocity stall detection adds a second system to safely detect mechanical end stops in the low velocity range. The velocity range for this system is defined by *TSGP_LOW_VEL_THRS* (lower velocity) and *TCOOLTHRS* (upper velocity, from which StallGuard+ uses the fixed absolute threshold set by *SGP_THRS*). This second stall detection system monitors *SGP_RESULT* for a falling slope. To program its sensitivity, configure the minimum change level between each two measurements using *SGP_LOW_VEL_SLOPE* and the required number of consecutive fullsteps *SGP_LOW_VEL_CNTS* (0…3: 1...4 events) showing each a *SG_RESULT* reduction above the minimum change level. This system avoids measurement noise triggering stall detection while providing a sensitive detection of suddenly increasing motor load that is typical for a drive hitting a mechanical obstacle. For improved noise reduction, activate the optional filter for *SGP_RESULT* using *SGP_CONF.sgp_filt_en*.

COIL_INDUCT Setting

To precisely calculate motor load, the driver uses a simplified motor model, mainly based on coil resistance and inductivity. While the driver automatically can determine motor resistance, inductivity must be set using the interface. The coil inductivity can be retrieved from the motor data sheet. As the inductivity setting covers several effects, the setting optimally is tuned for the desired motor type using an experimental setup.

Motor coil inductivity *COIL_INDUCT* can be set based on the motor data sheet, like the nominal motor current rating. Enter the coil inductivity in µH, that is, enter 8000 for a motor with 8mH coils. In case any highly-inductive additional filter inductors are used in series with the motor, add 71% of the inductance to this value. As the inductivity of the motor coils fundamentally results from the use of a certain iron core and a certain amount of wire turns, the motor manufacturer can give a good nominal value for a motor type. The temperature variation is small compared to the variation of coil resistance. A well-fitting setting of *COIL_INDUCT* is mandatory to allow StallGuard+ to cover the whole measurement range of 0 (full load) to (near) 255 (no load), especially at increased velocity.

An optional way of determining an initial value for the inductivity of an unknown motor is by tracing the coil current in standstill, when reducing motor current from the nominal value to 0 by setting *CUR_PI_LIMIT* to 0. This allows tracing LR behavior and calculating inductivity from the time needed for *PWM_CALC* to reduce to 1/e = 37% of the original value.

$$L = t_{37} \times R_{COIL}$$

However, precision is limited, as coil resistance, driver and interconnection resistance must be considered for R_{COIL}. Especially, in case the motor already is heated up, it makes sense to use the measured coil resistance rather than the manufacturer's nominal value for the calculation of inductivity.

Note: Depending on the individual motor, StallGuard+ may give best results with an inductivity setting slightly deviating from the data sheet value, for example, 90% or 110% of this. To find the best fitting scaling for a motor type or for an individual motor, do the experimental optimization using the data sheet value as a starting point.

COIL_INDUCT Experimental Optimization

For a broad range of motors and operation scenarios, the coil inductance value given by the motor data sheet results in reliable StallGuard+ behavior. However, moving towards the operational limits tuning of the inductivity value can further improve the StallGuard+ result.

The motor coil inductivity setting in the motor model is used to cover multiple effects. Therefore, the optimum setting not necessarily corresponds exactly to the motor inductivity, but for a certain motor type it can be a few percent higher or lower. Determine the best setting interactively by tracking StallGuard+ response at an increased velocity. Using a low velocity for testing is not recommended, as the thermal effect of coil resistance change affects the result, and the coil temperature can significantly rise or fall within seconds. The optimization test requires a possibility to slowly increase the mechanical load of the motor until it stalls. This procedure gives the best fitting result, as the motor is operated under realistic load conditions.

Set *COIL_INDUCT* to the nominal motor coil inductivity. Move the motor using the intended operating current at a medium velocity. To determine a well-fitting velocity, monitor *PWM_CALC*. Select the velocity by checking *PWM_CALC* to be significantly higher than the reading at the same current in standstill or low velocity, but still not reaching its limit of 4095.

Downloaded from Arrow.com.

Enable StallGuard filtering by setting *sgp_filt_enable*. Now, slowly increase the mechanical load. *SGP_RESULT* reads at roughly 200 to 255 with no load (depending on motor velocity, the upper value sinks with rising velocity), while going down to a positive value close to 0 before the motor stalls. If the motor stalls, before *SGP_RESULT* approaches 0, the actual *COIL_INDUCT* is too low. If it goes below 0 before the motor stalls, *COIL_INDUCT* is too high. Adapt the setting mostly at +-20% until a good value is determined.

A far too high setting results in instable, or completely out of range *SGP_RESULT*.

A far too low setting typically leads to a response in the range of 128 at full load.

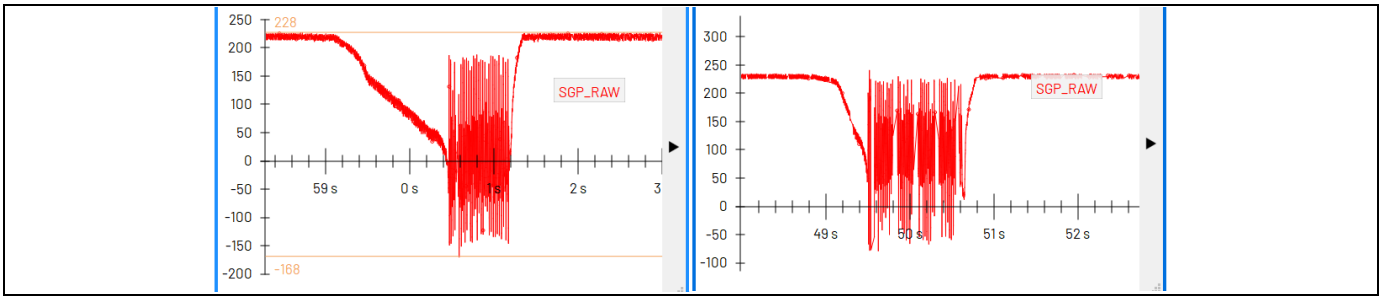


Figure 27. Good Setting – Motor Starts Stalling when SGP_RESULT Comes Close to 0 Versus Too Low Setting (Stall Starts Before Reaching 0)

To compensate for increased motor stray in series production without repeating the experimental optimization process, note the factor between the measured inductance and *COIL_INDUCT* setting and apply it to the individually measured motor inductance.

Note: Motors showing a large amount of resonance and deformed back-EMF may give inferior response leading to a reduced range of *SGP_RESULT* between no-load and maximum load. If the experimental optimization does not give a clear optimum, stay close to the original motor inductivity to ensure a stable result.

R_COIL Measurement

Especially for low velocity operation, a precise knowledge of the motor coil resistance is required to allow StallGuard+ to correctly determine the actual motor load. Strictly speaking, coil resistance is not completely constant, as resistance of the motor coil (copper) has a temperature dependance of 0.39% per Kelvin. Therefore, it can vary up to 40% when the motor heats up from environment temperature to its peak coil temperature (130°C limit for most standard motors). To keep track of the actual resistance, the driver continuously measures actual coil resistance for each coil whenever the motor is at standstill or rotating below the velocity threshold set by *T_RCOIL_MEAS*. This happens automatically and does not require more than 50 to 100 milliseconds of motor standstill or operation at very low velocity motion for a complete update at a minimum of 50% of *IRUN* current per coil.

Coil temperature can change within seconds (much faster than the temperature of the motor metal), and thus, a motor operating at full current may show significant change of its StallGuard+ result value in low velocity operation within a few seconds. Due to this, a recalibration can be required at certain intervals.

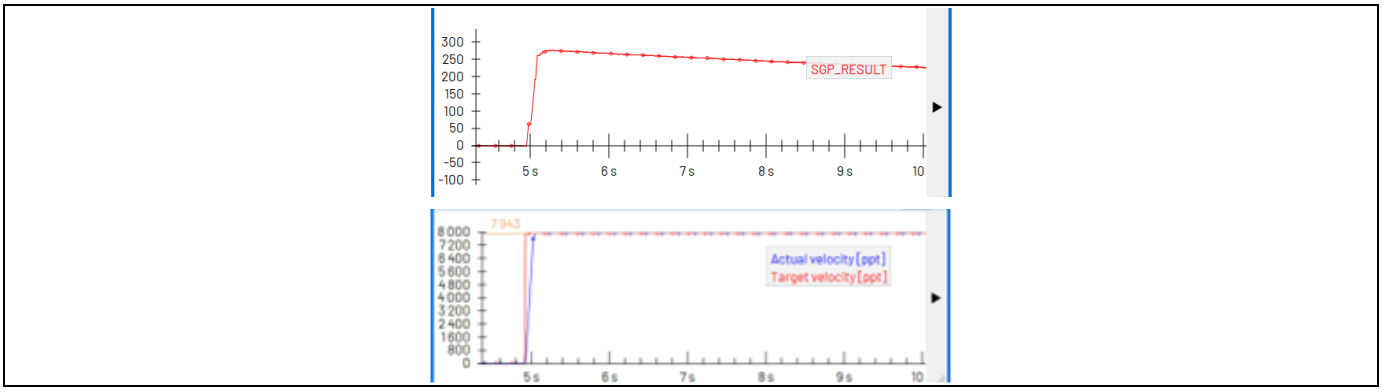


Figure 28. Change in StallGuard+ Result SGP_RESULT due to Coil Heatup when Working at Very Low Velocity

Downloaded from Arrow.com.

Optionally, define the coil resistance by a register individually for each coil (*R_COIL_USER_A*, *R_COIL_USER_B*). Use this option (enable by bit *rcoil_manual*) when the resistance value is well known, for example, when the motor temperature is directly measured or modeled in software. The conversion from resistance in Ohm to register-setting and back depends on the current range used:

Internal representation of resistance:

$$R_COIL_x = 4 \times (R_{LSplusHS} + R_{COIL}[\Omega]) \times 7.1\ [1/V] \times \sqrt{2} \times CurrentRange[A\ RMS]$$

This simplifies to:

$$R_COIL_x \approx 40[\frac{1}{V}] \times \left(R_{LSplusHS} + R_{COIL}\right) \times CurrentRange[A\ RMS]$$

or, to determine the coil resistance from the measured value:

$$R_{COIL}[\Omega] \approx \frac{R_COIL_x}{40 \times CurrentRange[A\ RMS]} - R_{LSplusHS}[\Omega]$$

where:

- R_COIL_x stands either for *R_COIL_AUTO* or the user programmed *R_COIL_USER.*
- $R_{LSplusHS}$ is the output resistance of the power stage plus any wiring and cabling resistance.
- Factor 7.1 results from the driver measuring supply voltage with 7.1 ADC counts per Volt.
- CurrentRange corresponds to the configured current RMS full-scale setting (1/SQRT(2) x peak current) as selected by *CURRENT_RANGE* and *CURRENT_RANGE_SCALE* in register *DRV_CONF:*

$$CurrentRange = 4.24A \times \frac{CURRENT_RANGE + 1}{4} \times \frac{CURRENT_RANGE_SCALE + 1}{4}$$

Note: The internally measured coil resistance value depends on the actual measurement range, as defined by the *CURRENT_RANGE* and *CURRENT_RANGE_SCALE* setting. Do not modify either when the motor is turning.

Consider the desired *CURRENT_RANGE* when setting coil resistance using *R_COIL_USER*.

Example R_COIL_AUTO Interpretation

A motor with 5Ω coil resistance is operated in $1A_{RMS}$ current range. What coil resistance measurement result can be expected?

Ignoring the contribution of cabling, the power stage offers roughly 100mΩ output resistance in the 1A range. Find the exact current range RMS current from the electrical table at 1.06A.

$$R_COIL_AUTO = 40[\frac{1}{V}] \times (5\Omega + 0.1\Omega) \times 1.06A = 216$$

With this, a reading of *R_COIL_AUTO* in the range of 216 can be expected for both coils. Realistically, it is slightly higher because of cabling and potential heat-up of the motor coils. The reading doubles in the 2A operation range and quadruples in the 4A range.

R_COIL Measurement Flow Chart

The internal R_{COIL} coil resistance calculation unit measures and updates *R_COIL_AUTO* whenever a measurement of one or both coils is possible: It checks for a low velocity (standstill, or near standstill), defined by *T_RCOIL_MEAS* and a minimum current of 50% of *IRUN* for the respective coil. In standstill, this is typically satisfied during *TPOWERDOWN* until the motor current is ramped down to *IHOLD*. The threshold velocity *T_RCOIL_MEAS* should be set to a value where the motor back-EMF is negligible compared to the voltage drop in coil resistance (see velocity thresholds), respectively, maximum 10% of the lowest velocity, where StallGuard+ and CoolStep+ shall be used. When both conditions are fulfilled, the measurement is updated using a filter with a time constant of 2048 chopper cycles. Becasue the coils are driven with a sine and a cosine wave, there is a good chance that only one coil can be measured during motor standstill because the

Downloaded from Arrow.com.

second coil has a too low current, for example, when stopping in or near a half step position. To address this, the first measurement for any coil after motor enathe ble is applied to both coils. Further, any change in coil resistance determined for one coil can automatically be applied to estimate the change of the second coil value (set bit *rcoil_thermal_coupling*).

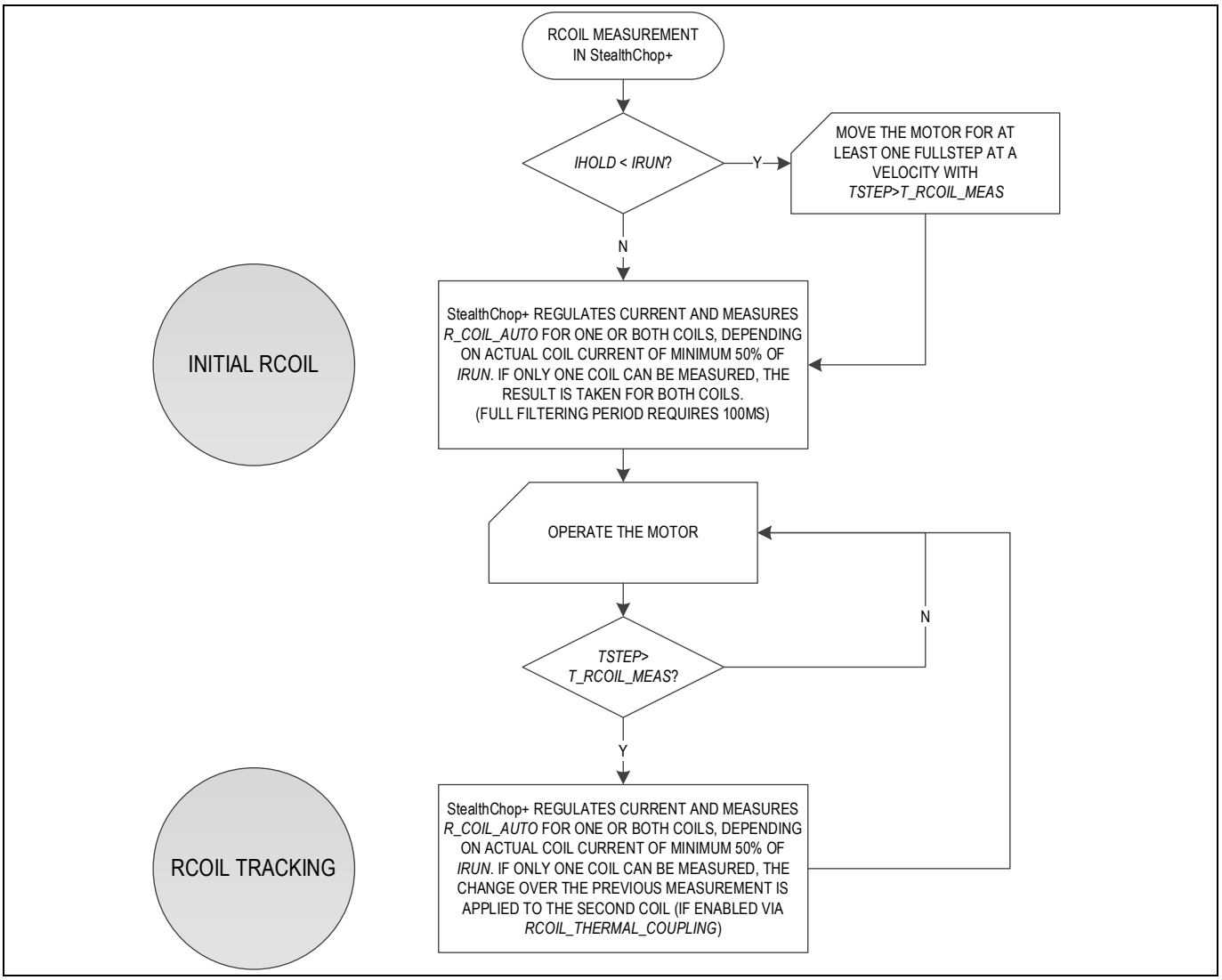


Figure 29. Flowchart for R_{COIL} Resistance Measurement (Following Motor Current Setting and Enabling of StealthChop+)

Use *TPOWERDOWN* to ensure sufficient time in standstill before the motor-current power down, to allow the regular update of the coil resistance measurement. A full measurement requires up to 100ms due to filtering, but any partial cycle also leads to a gradual update. Enable *rcoil_thermal_coupling* to update the resistance measurement for both the coils, unless the motor coils do not heat up in a uniform way. The coil resistance is measured during standstill and in low velocity motion if the current for each coil is min. 50% of the *IRUN* setting and current scale *CS_ACTUAL* minimum 50. The available measurement time, and with this measurement precision, can be increased by the *TPOWERDOWN* setting or by allowing measurement at very low velocity motion, defined by *T_RCOIL_MEAS* (see the velocity thresholds to select the proper velocity).

Note: Monitoring the motor temperature.

Monitor the coil resistance to calculate the motor coil temperature. As copper has a thermal coefficient of roughly 0.0039/K, the motor coil temperature can be estimated based on the resistance change over an initial measurement of the coil resistance at room temperature. A heat-up from 20°C to 120°C causes coil resistance to significantly increase to

Downloaded from Arrow.com.

139% of its initial value. Absolute accuracy for low-resistive motors (below a few Ohms) is reduced due to the contribution of other factors like temperature dependance of the output resistance of the IC itself.

Downloaded from Arrow.com.

CoolStep+

An electric motor has its best efficiency when operating at or near the 90° load angle. At 90° load angle, its magnetic stator field is offset by 90° to its rotor field. The motor torque rises with the load angle in a sine wave shape (see *Figure 30*). At 90° offset, the strength of the stator field required to yield the actual torque is minimum.

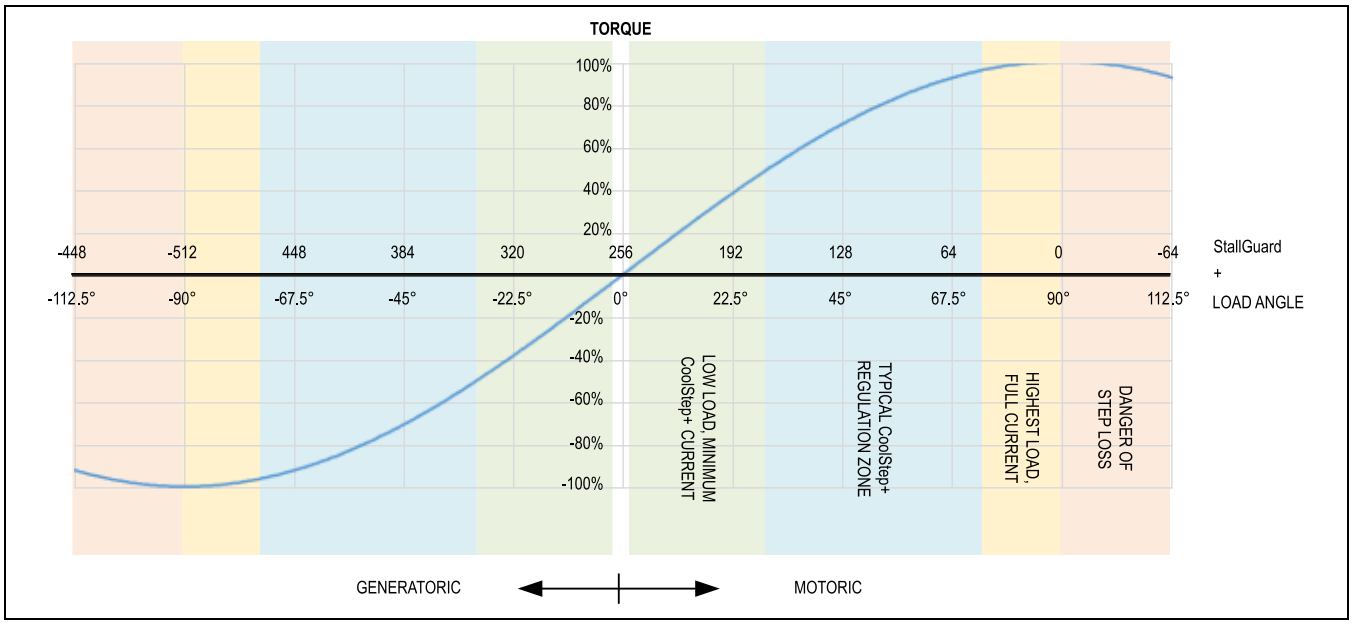


Figure 30. Motor Torque vs. Load Angle

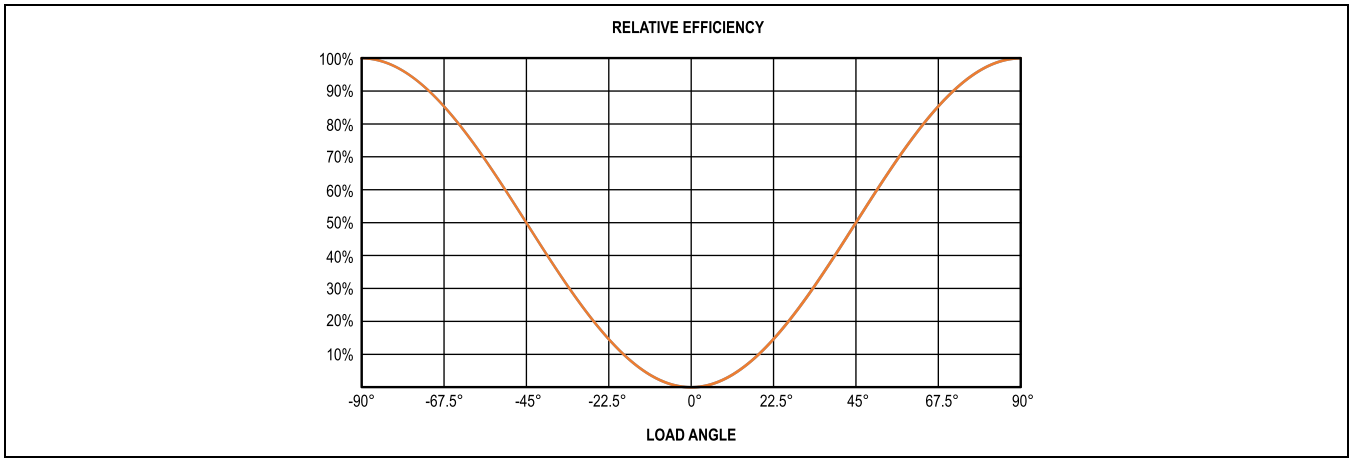


Figure 31. Relative Efficiency vs. Load Angle

Minimizing the strength of the stator field to this necessary minimum means reducing the motor current to the minimum required current. Keeping in mind that stepper motors classically are over-dimensioned and run at full current to safely provide the peak torque ever expected in the application, there is lot of optimization potential when adapting the current to what really is required at each point of time. As the motor power dissipation rises with the square of the motor current, while torque just rises linear with current, the adaptive motor current can save lots of energy. The efficiency graph (see *Figure 31*) shows the efficiency going with a shifted sine function with the load angle. CoolStep+ brings the load angle near to its optimum just below 90°, but within a safety zone of some 10°, ensuring intrinsic stepper motor stability against a change of load. This method combines optimized efficiency with the conventional operation's stability.

Downloaded from Arrow.com.

Note: CoolStep+ relies on the StallGuard+ result to supply the actual load reserve of the motor (*SGP_RAW*) describing the load angle deviation from 90°. Therefore, StallGuard+ must be properly configured before using CoolStep+.

Procedure to Configure CoolStep+

CoolStep+ comes with default settings for most parameters. However, detailed tuning allows to adapt the driver to the mechanical load profile, giving the best safety margin, while providing the highest possible level of energy saving.

Table 9. Procedure to Configure CoolStep+

NUMBER	TASK	DETAIL
1	Basic Motor Settings	Configure StallGuard+ by setting the motor inductivity and running the automatic determination of coil resistance.
2	Load Reserve Settings	Determine the load reserve values per guideline or leave default values.
3	Select Saving Factor	COOL_CUR_DIV setting 2 to 10 controls the lower current between 1/2 of IRUN to 1/10th of IRUN. A setting of 1 disables CoolStep+. Reduction to 1/4 or to 1/5 gives best stability and a high degree of energy saving for most motors. Start with COOL_CUR_DIV = 4 and recheck energy saving vs. stability after optimizing additional settings. For IRUN/COOL_CUR_DIV < 35, optionally adapt ANGLE_LOWER_I_LIMIT down to roughly 150 to keep the angle regulation activated at low current. (Default value is 256).
4	Determine Lower Velocity Threshold	Determine a lower velocity where StallGuard+ gives a stable value and correctly responds to a change in the motor load. Set TCOOLTHRS identical to TSTEP at this velocity.
5	Determine CoolStep PI Regulator Settings	Increase COOLSTEP_P to not more than 256 for initial fast response to a load step (first few milliseconds, if required), and COOLSTEP_I to a setting of 10 to 128 to give a quick response to the increasing motor load. Check that the load reserve does not undershoot upon an expected maximum load raise.
6	Determine Current Reduction Speed	Increase COOL_PI_DOWN_LIMIT to speed up the current reduction up to a value where it optimally fits to load change in the application but does not get instable. A setting of 330 gives a reaction time of roughly 100ms to 200ms. Current reduction shall be significantly slower than current increment.
7	Determine Current Ramp Up Speed	Switch on/off CoolStep+, for example, by leaving the velocity range (lower limit defined by TCOOLTHRS and upper limit by THIGH). Set COOL_PI_OFF_SPEED to a value giving fast but jerk-free increment of current to IRUN. Start with 10% of the value set for COOL_LOW_LOAD_RESERVE and increase up to 50% of this value. This way, leaving the velocity range causes the current incrementing in the same way as in the situation where SGP_RAW falls below the limit defined by LOW_LOAD_RESERVE.

Tuning CoolStep+

CoolStep+ is an advanced two-quadrant version of CoolStep: it adapts the motor current to the actual motor load by evaluating the actual StallGuard+ value *SGP_RAW*, which in turn represents the amount of unused, available motor torque or “load reserve”. Its target is to minimize the load reserve in a way that the motor in each situation has just sufficient torque to safely operate the application.

Target Load Reserve Generator

The target load reserve generator supplies the target value for the load reserve, as determined by the actual motor current.

In a low current situation, the available motor torque is low, and even a small increase of mechanical load can easily exceed the motor’s torque capability. Further, the reaction to an increased load angle requires some time for the PI regulator to smoothly ramp up current. Therefore, in a low current situation, the motor must not be operated as close to its torque limit as in a high current situation. In a high current situation, the motor’s available torque is high, and the same

Downloaded from Arrow.com.

increase in load has a comparatively lower impact on the motor load angle. Therefore, the motor in a high current situation can be operated with a smaller percentage of load reserve near to its optimum load angle of 90°. Considering motor power dissipation rises with the square of the motor current, this is a good fit, as operation at a higher motor torque offers much more possibility to save energy by going to the optimum load angle. This means that in a high torque, high motor current situation, the percentage of load reserve and with this unused, the available torque shall be minimized.

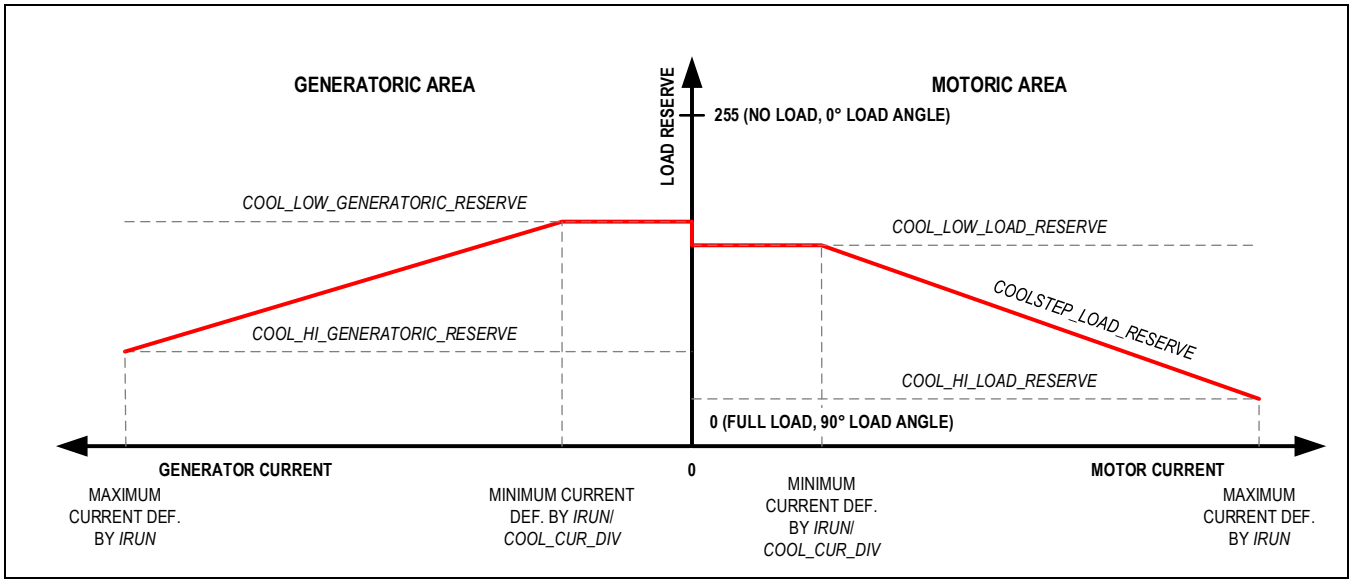


Figure 32. Function of Target Load Reserve Generator Showing Asymmetric Setting with Reduced Efficiency in Generatoric Operation

The motor, in certain situations, also acts as a generator: It feeds back energy whenever it is commanded to quickly decelerate, or the load pulls into the same direction as the rotation. In generatoric operation, CoolStep+ uses the same logic as in normal operation to generate a load reserve. For CoolStep angle regulation, load reserve values (*SGP_RAW*) 256 to 511 are mapped to 255 down to 0 and the driver calculates *COOLSTEP_LOAD_RESERVE* accordingly based on the generatoric reserve parameters.

As generatoric operation feeds back energy from the mechanical load to the power supply, keeping the voltage in safe limits can be a challenge for the power supply system, which typically supports single quadrant operation only. The capability to take fed back energy is limited by the supply capacitors and energy consumption of the stepper motor itself, and other loads. Thus, overvoltage protection mechanisms may have to be integrated. To address this problem, StealthChop+ allows to separately specify the load reserve function for generatoric operation. Choosing generatoric reserve values nearer to 0° load angle increases power dissipation in the stepper motor during generatoric operation. This way, less or no energy goes back to the supply. However, as the current does not increase beyond the specified *IRUN* value, the effectiveness of this measure must be proven on a case-by-case base.

Rule of Thumb for Load Reserve

Motor efficiency is best when the load reserve is 0. But, as a stepper motor is a pure feed-forward controlled system, it needs a safety margin of some minimum positive value of load reserve. Further, this margin allows compensating for measurement inaccuracy (for example, due to deviation of the inductance versus inductance setting or due to mismatch of resistance value).

Choosing *COOL_HI_LOAD_RESERVE:*

This is the load reserve value for a highly loaded motor operating at maximum current (*IRUN*), and therefore, it is most critical for efficient operation. To assess the required safety margin, check the bandwidth of the minimum (positive) *SGP_RESULT* within the application limits before a motor stall occurs. Check for corner case stability by varying the

Downloaded from Arrow.com.

motor temperature and inductance setting (for example, by ±10%). Apply additional margin of 50% to 100%, or minimum an offset of 10, to allow the CoolStep regulator to react upon undershooting.

As a result, *COOL_HI_LOAD_RESERVE* should be in the range of 25 to 80.

Choosing *COOL_LOW_LOAD_RESERVE:*

The low load reserve setting should be higher than the high load reserve, as a certain load change relatively has a higher impact, and thus, the driver needs more headroom to react, especially when it is intended to reduce the motor current by more than a factor of two in low load situations.

- Low reduction case: With *COOL_CUR_DIV* = 2, use the same or a value slightly above *COOL_HI_LOAD_RESERVE.*
- Standard case: Use a value of roughly 64 to 128 above *COOL_HI_LOAD_RESERVE* to increase headroom.

Choosing *COOL_HI_GENERATORIC_RESERVE* and *COOL_LOW_GENERATORIC_RESERVE:*

- Standard case: Use the same values as for *COOL_HI_LOAD_RESERVE* and *COOL_LOW_LOAD_RESERVE.*
- For reduced generatoric feedback to power supply: Use increased values compared to those for motoric operation. Do not exceed roughly 200.

CoolStep+ Regulator

CoolStep+ current regulation aims to use the minimum current to safely move the motor. To ensure stable operation upon load change, it quickly reacts to the rising load by increasing current to prevent step loss but exhibits a limited slope with falling load.

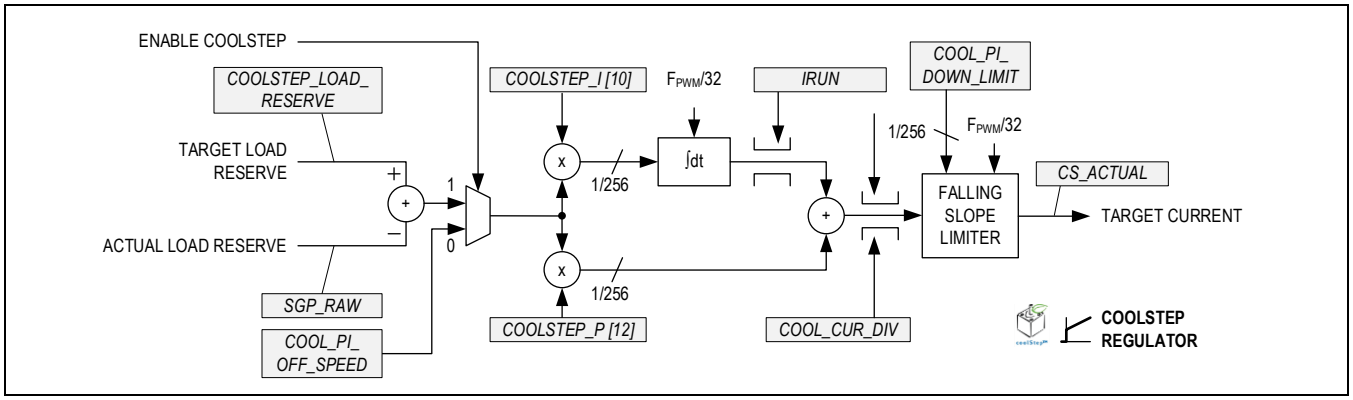


Figure 33. Structure and Basic Parameters of CoolStep+ Regulator

CoolStep+ employs a PI-regulator for the target motor current, like the current and angle PI regulators. The regulator quickly reacts to a change in motor load, especially when the load increases. This is mandatory to avoid step loss upon sudden load increase. As a result, it adapts the motor target current (*CS_ACTUAL*) by regulating it between the upper limit set by *IRUN*, respectively, a lower limit defined as a fraction of *IRUN* selected by *COOL_CUR_DIV* to 1/2, 1/3, 1/4 … down to 1/10 of *IRUN*.

The regulator is enhanced by a falling current slope limitation (*COOL_PI_DOWN_LIMIT*). This limitation sets the maximum target current change in 1/256 steps per each 32 PWM periods. This allows the motor and the regulator cascade to react while CoolStep+ ramps down the motor current.

An additional slope limiter ensures a smooth transition when CoolStep is disabled, for example, when leaving its velocity range by decelerating the motor or exceeding the upper velocity threshold. *COOL_PI_OFF_SPEED* sets the slope used in this case. It acts the same way as a regulation difference by feeding a certain value into the regulator and using its P and especially its I part to react to the difference. However, it should not be set too slow, as the current is momentarily forced to the *IRUN* setting in the moment where the velocity set by *TPWMTHRS* is exceeded, or whenever the *standstill* flag goes active. *COOL_PI_OFF_SPEED* concerns current increment, while *COOL_PI_DOWN_LIMIT* concerns current ramp down.

Downloaded from Arrow.com.

Note: Tune the current regulator before tuning the CoolStep+ PI regulator, as CoolStep+ relies on the motor current regulator to react sufficiently fast to a change in the motor target current.

CoolStep+ P Tuning

The CoolStep+ P-part instantly reacts to an undershoot in the load reserve. A *COOLSTEP_P* = 256 gives a 1:1 response. For example, undershooting the target load reserve by 1 leads to a current increment of one step. A P-regulator, however, directly forwards the jitter of the measured signal. Therefore, the coefficient should not be chosen too large. Trace *SGP_RAW* to assess the jitter of the load measurement within the application conditions. Therefore, the P part should not be set to high values, unless a very quick load response is required. Select *COOLSTEP_P* in the range of a few 10 up to 256. Unless mechanical load is extremely abrupt, response by the I-part can take over most of the reaction. Always select the filtered raw value (*load_filt_en* = 1) to reduce the current jitter feed through, especially when using a large P-part. The filter calculates the mean value of the previous 16 load reserve measurements taken within the previous 32 chopper cycles, and thus, does not introduce significant delay.

CoolStep+ I Tuning

The CoolStep+ I-part uses the same scaling as the P-part. Therefore, its setting should be below the P-part. As a rule of thumb, select *COOLSTEP_I* in the range of 5 to 128.

At 25kHz chopper frequency, CoolStep+ executes roughly 750 times per second, which is roughly ten times faster than a typical mechanical time constant. This means, the I-regulator typically responds within several cycles to a change (increment) in the mechanical load.

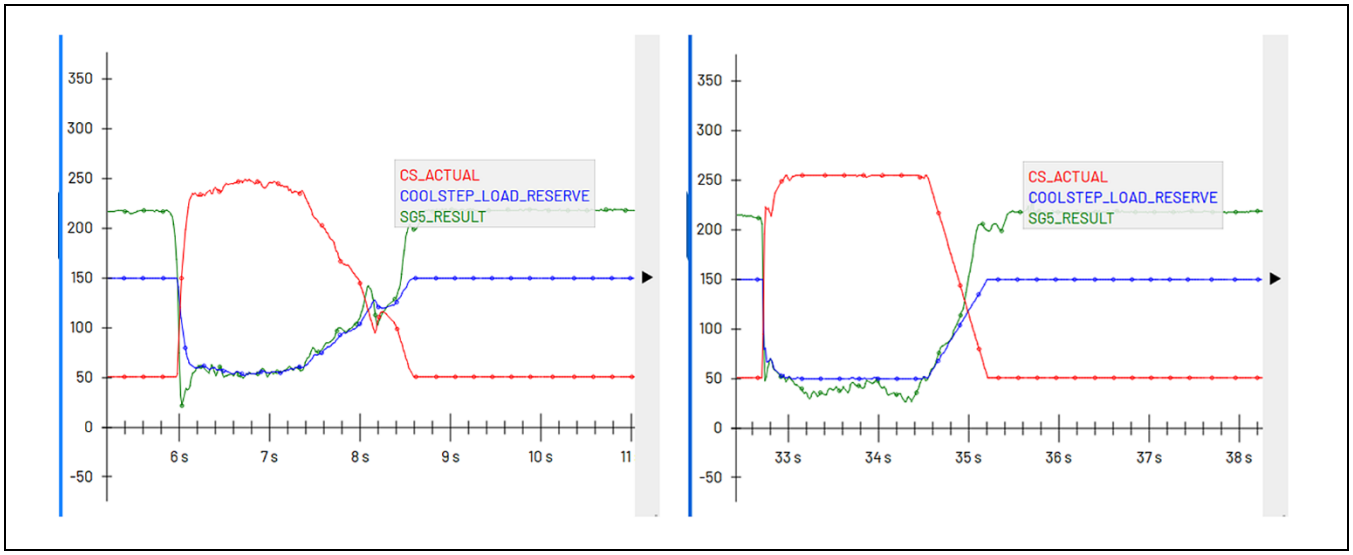


Figure 34. PI Regulator Setting too Small – SGP_RESULT (Green) Undershoots Upon Load Increment vs. Sufficiently High Setting

COOL_PI_DOWN_LIMIT Tuning

As the CoolStep PI regulator must be parameterized to quickly respond to an increasing load, the same response time for a falling load easily can lead to an instable regulation, exciting mechanical oscillation. The falling slope limiter addresses this by limiting the maximum current reduction per regulation cycle (once each 32 chopper cycles). Increase *COOL_PI_DOWN_LIMIT* until the current ramps down as fast as the motor load decreases in the application to yield maximum energy saving. For optimum energy efficiency in applications with abrupt load change, use a value that sufficiently allows current reduction to cover most of the times of low load.

$$COOL_PI_DOWN_LIMIT = \frac{IRUN - \frac{IRUN}{COOL_CUR_DIV}}{t_{SLOPE} \times f_{CHOP}} \times 8192$$

where:

- t_{SLOPE} describes the intended time for current reduction from IRUN down to the fraction programmed by *COOL_CUR_DIV.*
- f_{CHOP} is the selected chopper frequency.

Downloaded from Arrow.com.

Example: At 25kHz chopper frequency, the *COOL_PI_DOWN_LIMIT* setting of 330 corresponds to a current decrease by 100 steps within 100ms. This means, a reduction from *IRUN* = 250 to *IRUN* = 50 requires 200ms.

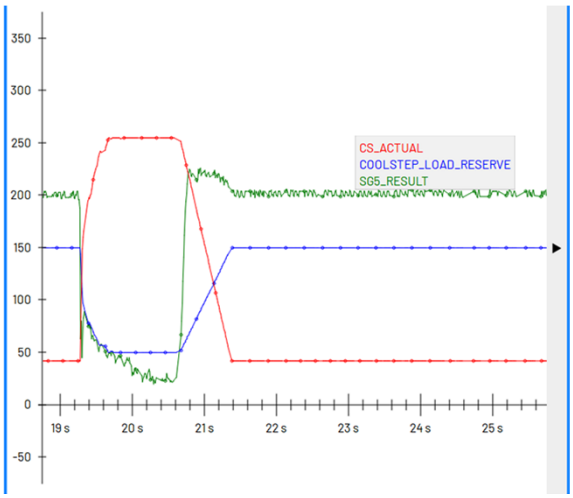


Figure 35. Limitation of Falling Slope Controlled by COOL_PI_DOWN_LIMIT

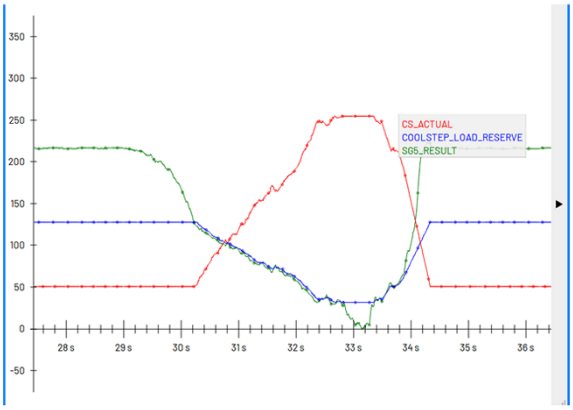


Figure 36. SGP_RESULT Decreasing with Rising Mechanical Load and Tracking COOLSTEP_LOAD_RESERVE Until Maximum Current (CS_ACTUAL = 255) is Reached. Limited Slope for Current Reduction (COOL_PI_DOWN_LIMIT) Upon Load Decreasing

Rule of Thumb for CoolStep+ PI and Slope Limits

Operate the motor in the center of the desired velocity range for tuning.

COOLSTEP_P = 256 provides a 1:1 compensation of motor current with a mismatch of load reserve. As a larger value leads to increased current ripple, a typical setting is lower than this.

Choosing *COOLSTEP_P*:

- Select *COOLSTEP_P* in the range of 25 up to 256. Tune *COOLSTEP_I* for quick reaction.

Choosing *COOLSTEP_I*:

Downloaded from Arrow.com.

- Set in the range 5 to 128 and carefully check for sufficiently fast reaction to load increase. A high value allows faster reaction speed but risks regulation overshooting. Increase *COOLSTEP_P* if initial reaction is still too slow at a high setting, or optionally, increase the load reserve. Best stability is with *COOLSTEP_I* < ½ *COOLSTEP_P*.
- Check for stability at the highest and lowest applicable velocity. In case the regulation is instable and overshoots at low currents, consider a reduced CoolStep+ current division factor, for example, *COOL_CUR_DIV* = 4 or try reducing *COOL_PI_DOWN_LIMIT*. In case stability at low velocities cannot be reached with the same parameter set required for a sufficiently quick reaction at a higher velocity, consider decreasing *TCOOLTHRS*.

Choosing *COOL_PI_DOWN_LIMIT*:

- Starting from a low setting with a reaction of a few 100ms increase, until current saving in low load periods is optimized. Recheck the stability at medium load, when approaching the mechanical reaction time in the range of a few 10ms.

Choosing *COOL_PI_OFF_SPEED*:

- A setting in the range of 10% to 50% of *COOL_LOW_LOAD_RESERVE* normally is sufficient, as it simulates a load increment. Accelerate and decelerate the motor at the desired acceleration from/to a velocity range, where CoolStep is off, for exmaple, by stopping the motor. Decrease *COOL_PI_OFF_SPEED* to reduce the jerk caused by the sudden rise of current. When switching to SpreadCycle at higher velocity using *TPWMTHRS*, tune together with the velocity difference defined by *THIGH* (a higher velocity where CoolStep+ is switched off) and *TPWMTHRS* (velocity where the driver switches over to SpreadCycle). Increase *COOL_PI_OFF_SPEED* or increase velocity difference if a jerk occurs when reaching *TPWMTHRS* at the highest application acceleration.

Attention: CoolStep+ PI regulator tuning can be a tradeoff between reaction time and stability. While high motor velocities often demand a short reaction time, operation at low velocity may result in reduced stability with the same parameter. Consider raising the lower velocity threshold as controlled by *TCOOLTHRS* for operation in the normal (medium and high) application velocity range and loading a second parameter set when working for extended times at low velocities.

Velocity Thresholds Relevant for StealthChop+

StealthChop+ advanced features are dependent on the availability of sufficient motor back-EMF (BEMF) signal (voltage induced in the motor's stator coils by rotation of the rotor). A certain BEMF is required to correctly determine the motor load and load angle. Therefore, several velocity thresholds are provided to seamlessly enable/disable the functionality. The BEMF is dependent on the motor type and its velocity.

Table 10. Velocity Thresholds Relevant for StealthChop+

PARAMETER	DESCRIPTION	CONDITION
T_RCOIL_MEAS	Upper velocity threshold for the automatic measurement of coil resistance. Lower velocity threshold for operation of the angle regulator and for low velocity StallGuard+ stall detection system.	Determine the velocity near to standstill, where back-EMF is negligible. Typical range: 1 RPM to 10 RPM. Below this velocity, StealthChop+ treats the motor as a pure ohmic load and determines its coil resistance. Proposed typical condition for threshold velocity: $U_{BEMF} < 3\% \times U_{COIL}$ In this condition, the back-EMF is less than 3% of the resistive voltage drop. $T_RCOIL_MEAS \geq 4 \times TSGP_LOW_VEL_THRS$
TSGP_LOW_VEL_THRS	Upper velocity threshold for freezing/zeroing *SGP_VALUE*. Lower velocity threshold for low velocity StallGuard.	Select the freezing of *SGP_VALUE* below the selected velocity by *sgp_low_vel_freeze* = 1. When 0, *SGP_VALUE* is zeroed out instead. Proposed typical condition for threshold velocity: $U_{BEMF} < 3\% \times U_R$

Downloaded from Arrow.com.

		$TSGP_LOW_VEL_THRS \leq 1/4 \times T_RCOIL_MEAS$ Further, a sufficient BEMF voltage is required. $BEMF_ABS \geq 30$
TCOOLTHRS	Lower velocity threshold for CoolStep+ and StallGuard+ operation; upper threshold for low velocity StallGuard+ stall detection system.	Determine the velocity, where there is sufficiently noise-free StallGuard result. Also check the stability of the absolute value with regards to the typical heat up of motor coils during an operation cycle. A new operating cycle starts after the motor standstill with an update of the coil resistance measurement whenever the motor accelerates beyond *T_RCOIL_MEAS*. Ensure *TSGP_LOW_VEL_THRS* > *TCOOLTHRS* Proposed typical condition for threshold velocity: $U_{BEMF} > 30\% \times U_R$ Further, a sufficient BEMF voltage is required. $BEMF_ABS \geq 100$
TUDCSTEP	Lower velocity threshold to enable µDcStep and minimum velocity to which µDcStep decelerates.	Select *TUDCSTEP* < *TCOOLTHRS* with a sufficient velocity difference to allow CoolStep to ramp down current while accelerating from CoolStep enable velocity to µDcStep enable velocity.
THIGH	Upper velocity threshold for CoolStep+. The current is increased up to *IRUN* beyond this velocity.	Use this velocity threshold when combining StealthChop+ operation with SpreadCycle as determined by *TPWMTHRS*. The velocity difference between *THIGH* and *TPWMTHRS* allows sufficient time to smoothly ramp up the current to *IRUN* when leaving CoolStep+ velocity area as defined by *COOL_PI_OFF_SPEED*. This avoids a potential jerk caused by the current difference. Proposed typical condition for threshold velocity: 110% to 120% of *TPWMTHRS*
TPWMTHRS	Upper velocity threshold for StealthChop+ operation. The IC defaults back to SpreadCycle above this velocity and switches over stall detection to StallGuard2 and enables CoolStep2, if configured.	StealthChop+ can operate the motor within the velocity range standstill up to 100%…150% of the velocity where the motor voltage reaches the supply voltage level, characterized by *PWM_CALC* reaching its limit of 4095. Beyond this velocity, the motor goes into a field-weakening operation and stable StealthChop+ operation may not be possible. This typically also is the point where the available motor torque starts reducing significantly. SpreadCycle offers the ability to work far into the field-weakening velocity. However, the motor torque is reduced, which might be a sign that the selected motor is not optimal for the application. Probably, the motor is too strong if operation in this area is required. Choose a motor with lower coil resistance and higher operating current to extend StealthChop+ operating range to higher velocities or increase supply voltage. Proposed condition for threshold velocity: 100% to 150% of the velocity, where: • The required motor drive voltage reaches V_S. • Where *PWM_CALC* reaches 4095 when running at the desired (maximum) current and with the mechanical load attached.

Understanding the Back-EMF Constant of a Motor

The back-EMF constant is the voltage a motor generates when turned with a certain velocity. Often, motor data sheets do not specify this value, as it can be deducted from the motor torque and coil current rating. Within SI units, the back-EMF constant C_{BEMF} has the same numeric value as the torque constant. For example, a motor with a torque constant of 1 Nm/A would have a C_{BEMF} of 1V/rad/s. Turning such a motor with 1 rps (1 rps = 1 revolution per second = 6.28 rad/s) generates a back-EMF voltage of 6.28V. Thus, the back-EMF constant can be calculated as:

Downloaded from Arrow.com.

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

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

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

$$U_{BEMF} = C_{BEMF}\left[\frac{V}{\frac{rad}{s}}\right] \times 2\pi \times \frac{f_{CLK} \times VACTUAL}{2^{24} \times MSPR}$$

The B_{EMF} voltage U_{BEMF} is valid as RMS voltage per coil. Thus, the nominal current has a factor of two in this formula.

MSPR is the number of microsteps per rotation, example, 51200 = 256µsteps multiplied by 200 full steps for a 1.8° motor.

VACTUAL is the actual velocity as commanded by the internal ramp generator.

StallGuard+ calculates U_{BEMF} to determine the motor load. As U_{BEMF} cannot be measured directly but has to be deducted from the motor current measurement resulting from the application of a certain drive voltage, the overlaying resistive voltage U_{COIL} drop plays a significant role for the exactness of this calculation, and therefore, must be known as precisely as possible. U_R describes the voltage drop in the motor coil:

$$U_R = R_{COIL} \times I_{COIL_IRUN}$$

R_{COIL} is the ohmic resistance of the coil.

I_{COIL_IRUN} is the RMS coil current at the *IRUN* setting, that is, typically, the motor's specified RMS current, unless *IRUN* is deliberately set to a lower or higher value.

Having understood the source of U_{BEMF} and U_R, it is clear that StallGuard+ and CoolStep+ only work reliably in a velocity range where U_{BEMF} sufficiently exceeds the height of the error that occurs when calculating the resistive voltage drop U_R. As a thumb rule for CoolStep+, it can work nicely in a range where U_{BEMF} >30% × U_R. This considers an uncertainty of the determined coil resistance due to the temperature change of the motor during operation.

Overview of Velocity Dependent Features

The superscript options 1 to 4 are explained in *Table 11*.

Table 11. Overview of Velocity Dependent StealthChop+ Features

(Lower) Threshold Velocity (Ascending)	RCOIL Calibration	Low Velocity StallGuard+	StallGuard+	CoolStep+	µDcStep	Angle Regulator	StealthChop+	SpreadCycle	StallGuard2	CoolStep
Standstill	Y						Y			
T_RCOIL_MEAS						Y	Y			
TSGP_LOW_VEL_THRS		Y				Y	Y			
TCOOLTHRS1			Y	Y		Y	Y			
TUDCSTEP			Y	Y	Y	Y	Y			
THIGH3			Y			Y	Y			
TPWMTHRS								Y	Y1N2	Y1N3
TCOOLTHRS2								Y	Y	Y4N3
THIGH4								Y	Y	N

Downloaded from Arrow.com.

Option 1: TCOOLTHRS(1) either can be used to control CoolStep+ (recommended in combination with StealthChop+) and then should be set to a velocity in StealthChop+ range, or

Option 2: TCOOLTHRS(2) can be used to control the activation of CoolStep in combination with StallGuard2.

Option 3: THIGH(3) can either be used to disable CoolStep+ and ramp up the current before transitioning to SpreadCycle (recommended when switching to SpreadCycle with CoolStep not in used), or

Option 4: THIGH(4) can be used to disable CoolStep at high velocities.

Downloaded from Arrow.com.

Open Load Detection

StealthChop+ allows precise feedback over the actual operating state of the motor using the ADC measurements, respectively, the current amplitude calculated from it (*AMPL_MEAS*) and the PWM value required to yield the actual current (*PWM_CALC*). Also, the calculated coil resistances (*RCOIL_AUTO_A* and *RCOIL_AUTO_B*) allow a precise check of each motor coil and its connection. In case the motor connection is lost during operation, a simple integrated open load test is available in *DRV_STATUS* through open load flags *OLA* and *OLB*. Open load detection is just informative, and the IC does not trigger any action upon detection.

Open load detection during motor rotation:

During motor rotation, open load detection signals if at least 75%/50%/25%, respectively, 12.5% (select by *OL_THRSH* in *PWM_CONF*) of the requested peak current is reached minimum one time within a half electrical rotation (indicated by each change of the MSB of *MSCNT*). The first update is after one rotation in a single direction. Therefore, open load also shows when the motor cannot reach its target current in situations where the velocity requires a drive voltage significantly above the actual supply voltage level. In these cases, *PWM_CALC* typically has reached its maximum of 4095. However, this method also can give spurious open load detection in case the current regulator cannot keep up with the acceleration settings.

Note: Check for open load (*OLA* and *OLB*) during the slow rotation of the motor for at least one electrical rotation.

Open load detection during motor standstill:

Open load detection also is updated in standstill for each coil whenever the requested coil current is at least 80 ADC counts. Therefore, the update depends on the current setting and microstep position. Open load is signaled in case the coil reaches less than half of the target current. As it monitors a coil whenever it carries significant current, it safely indicates a major loss of holding torque.

Note: Optimally, check for open load (*OLA* and *OLB*) during the slow rotation of the motor for at least one electrical rotation.

Downloaded from Arrow.com.

µDcStep

µDcStep adds on top of CoolStep+ and is a DC-motor-like mode of operation intended to prevent a step loss by allowing a dynamic response of the motor velocity to an overload situation. It improves applications where mechanical load at high velocities can increase beyond the available motor torque and where a load dependent change of motor velocity can be tolerated. The system on the other hand opens the possibility to move as fast as possible (as fast as the motor can turn the mechanics under given circumstances).

µDcStep monitors the current CoolStep+ is using (*CS_ACTUAL*) to operate the motor. Once CoolStep+ reaches a user-defined upper current limit (*DECEL_THRS*, limit programmed in *DECEL_THRS*/16 of *IRUN*) in response to increased mechanical load, µDcStep starts decelerating the motor until the motor load has decreased sufficiently to let CoolStep+ ramp down the current below the current limit again. The lowest velocity to which µDcStep decelerates is defined by its activation threshold (define by *TUDCSTEP*) and must lie above the CoolStep+ activation threshold. The motor remains at the reduced velocity until CoolStep+ current sinks below a second user-defined limit (*ACCEL_THRS*). This limit can be identical to or lower than the upper current limit (*DECEL_THRS*). The driver then follows the acceleration ramp to reaccelerate.

The effectiveness of µDcStep relies on the flywheel mass of the motor providing a short time higher torque as well as mechanical load typically reducing with lower velocity, while the available motor torque increases (slightly). To give a sufficient effect of the flywheel mass of the motor, program a fast deceleration ramp setting.

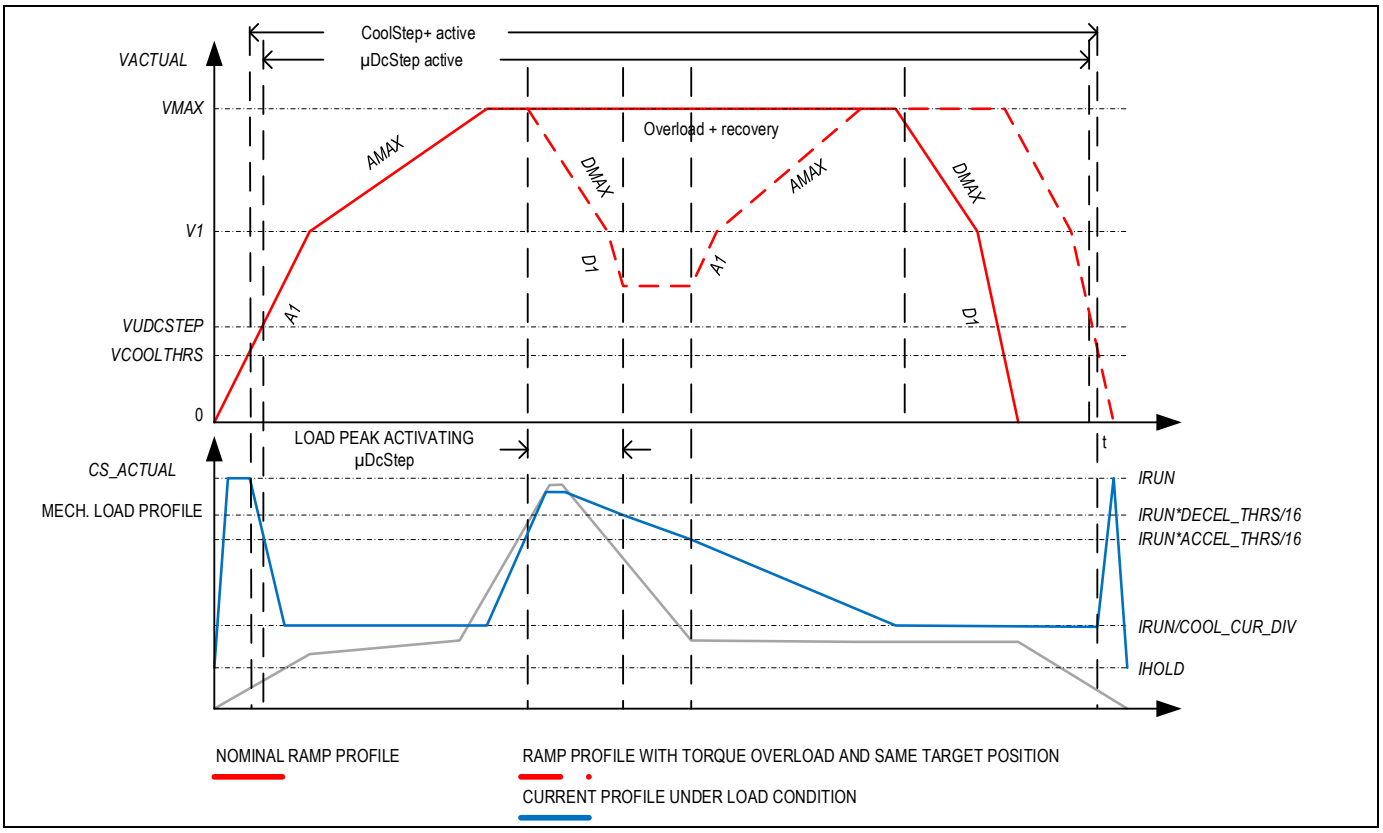


Figure 37. Example for Velocity Profile with µDcStep

Downloaded from Arrow.com.

Enabling µDcStep

Once CoolStep+ is correctly configured, µDcStep can be enabled using a few additional settings.

Table 12. µDcStep Enable Procedure

NUMBER	TASK	DETAIL
1	Configure StallGuard+ and CoolStep+	Tune CoolStep+ to respond quickly to both a rising load and a falling load.
2	Set Upper Current Limit for Velocity Reduction	Configure an upper current limit to activate µDcStep in the range of typical 12 to15, for example, by programming *DECEL_THRS* to 14. The current limit, this way, is programmed to *DECEL_THRS*/16 of *IRUN.*
3	Set Reacceleration Current Level	Set *ACCEL_THRS* to an identical current limit or typical 1 to 4 counts lower than *DECEL_THRS*. This is the current limit for reacceleration. The difference between *DECEL_THRS* and *ACCEL_THRS* improves the velocity stability when the load stays near the deceleration threshold.
4	Enable µDcStep	Enable µDcStep by setting *DCU_CONF.dcu_enable* flag. This disables the virtual stop switch functionality and selects µDcStep as the source for the ramp generator instead.
5	Configure Deceleration Mechanism	Now, set *en_softstop*, *en_virtual_stop_l* and *en_virtual_stop_r* in SW_MODE to allow µDcStep to influence the ramp generator velocity like a virtual stop switch.
6	Select Lower Velocity Threshold	*TUDCSTEP* selects a lower velocity threshold for activation and at the same time the lowest velocity to which µDcStep decelerates. However, the velocity slightly may undershoot this value depending on the deceleration settings. Select *TUDCSTEP* with a certain distance to describe a higher velocity than *TCOOLTHRS*, to allow CoolStep+ to start ramping down the motor current while accelerating to the µDcStep threshold velocity.
7	Configure Deceleration Ramp	Select high deceleration settings for *DMAX*, *D1,* and *D2* (whichever is used, as defined by velocity thresholds *V1* and *V2*) to allow sufficiently quick deceleration upon a load impact. Set *TVMAX* to 0 or select *TVMAX* to control the minimum delay time for reacceleration.

Downloaded from Arrow.com.

SpreadCycle and Classic Chopper

The normal mode of operation is StealthChop+ chopper. However, in some situations, a more classic chopper principle may be beneficial, for example, for the high velocity operation of a motor far beyond the point where the motor's back-EMF reaches the supply voltage level.

While StealthChop+ is a voltage-mode, PWM-controlled chopper, SpreadCycle is a cycle-by-cycle current control. Therefore, it can react extremely fast to changes in the motor velocity or motor load. The currents through both the motor coils are controlled using choppers. The choppers work independently of each other. *Figure 38* shows the different chopper phases.

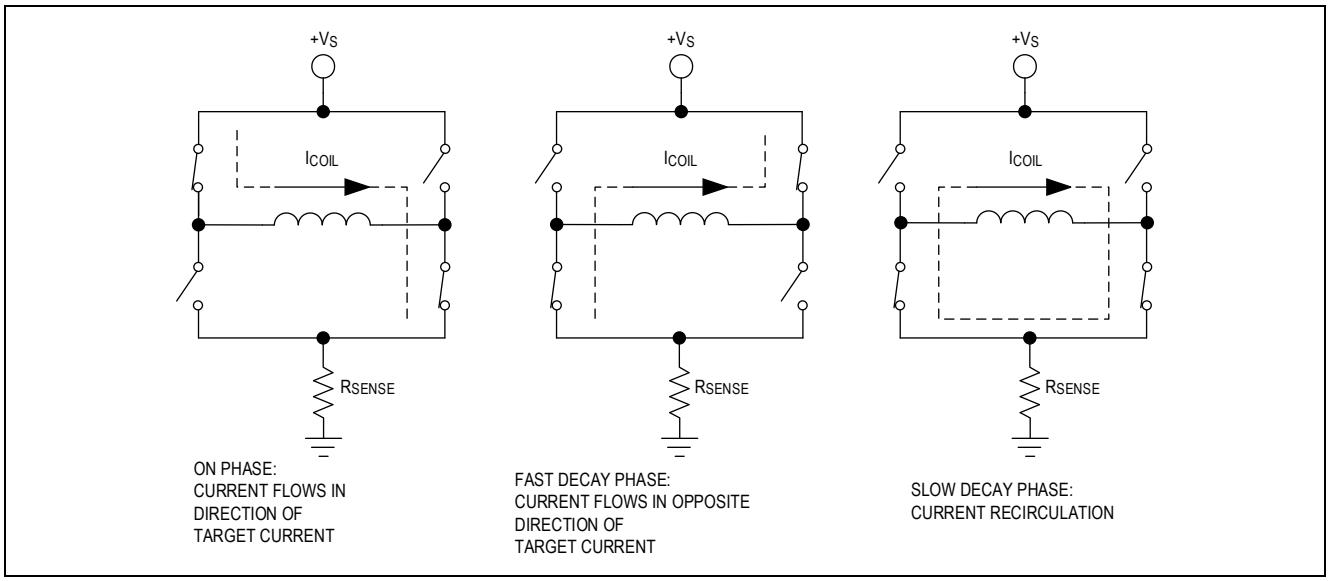


Figure 38. Typical Chopper Decay Phases (Sense Resistor is Only Symbolic for Current Measurement in Low-Side Path)

Although the current can be regulated using only 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.

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

The chopper frequency is an important parameter for a chopped motor driver. A too low frequency might generate audible noise. A higher frequency reduces 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. Therefore, a compromise must be found. Most motors optimally work in a frequency range of 20kHz to 40kHz. The chopper frequency is influenced by a number of parameter settings as well as by the motor inductivity and supply voltage.

Hint: A chopper frequency in the range of 20kHz to 40kHz gives a good result for most motors when using SpreadCycle. A higher frequency leads to increased switching losses.

Downloaded from Arrow.com.

Table 13. Parameters Controlling SpreadCycle and Classic Constant Off-Time Chopper

PARAMETER	DESCRIPTION	SETTING	COMMENT
TOFF	Sets the slow decay time (off time). This setting also limits the maximum chopper frequency. For operation with StealthChop+, this parameter is not used, but it is required to enable the motor. In case of operation with StealthChop+ only, any setting is OK. 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 24 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}
chm	Selection of the chopper mode. Default = 0	0	SpreadCycle
		1	Classic constant off time

SpreadCycle Chopper

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

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

Example for determining an initial value for *TOFF*:

Target chopper frequency: 25kHz

Assumption: Two slow decay cycles make up 50% of the overall chopper cycle time.

→ tOFF = 1/25kHz × 50/100 × 1/2 = 10 µs

For the *TOFF* setting, this means: *TOFF* = (tOFF × f_{CLK} − 24)/32.

With 16MHz clock f_{CLK}, this results in *TOFF* = 4.25, which requires a setting of *TOFF* = 4.

The hysteresis start setting forces the driver to introduce a minimum amount of current ripple into the motor coils. The current ripple must be higher than the current ripple, which is caused by resistive losses in the motor to give the best microstepping results. This allows the chopper to precisely regulate the current for both the rising and falling target current. The time required to introduce the current ripple into the motor coil also reduces the chopper frequency. Therefore, a higher hysteresis setting leads to a lower chopper frequency. The motor inductance limits the ability of the chopper to

Downloaded from Arrow.com.

follow a changing motor current. Further, the duration of the on-phase and the fast decay must be longer than the blanking time, because the current comparator is disabled during blanking.

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

A low to medium default value for the hysteresis (for example, effective hysteresis = 4) often already gives good results. The setting can be optimized by experimenting with the motor: a too low setting results in reduced microstep accuracy, while a too high setting leads to more chopper noise and motor power dissipation. When the fast decay time is slightly longer than the blanking time, the setting is optimum. Reduce the off-time setting if this is hard to reach.

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

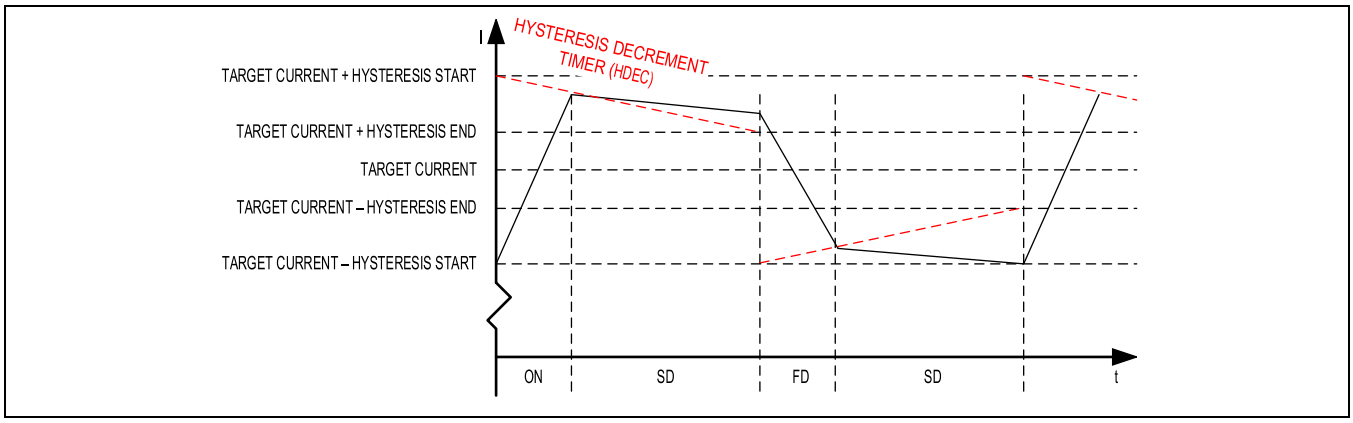


Figure 39. SpreadCycle Chopper Scheme Showing the Coil Current During a Chopper Cycle

Table 14. SpreadCycle Mode Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
HSTRT	Hysteresis start setting. This value is an offset from the hysteresis end value *HEND*. Default = 5	0…7	*HSTRT* = 1…8 This value adds to *HEND*.
HEND	Hysteresis end setting. Sets the hysteresis end value after a number of decrements. The sum *HSTRT* + *HEND* must be ≤16. At a current setting of maximum 251 (amplitude reduced to 245), the sum is not limited. When using a modified sine wave, make sure the sine wave peak value, scaled by (*IRUN* + 1)/256, does not exceed 256 - (*HSTRT* + *HEND*). This can either be ensured by scaling down the sine wave to a peak of 248, like the default wave, or by limiting *IRUN* and *IHOLD* accordingly. Default = 2	0…2	-3…-1: negative *HEND*
		3	0: zero *HEND*
		4…15	1…12: positive *HEND*

Downloaded from Arrow.com.

Example:

A hysteresis of 4 is chosen. First fitting setting:

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

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

To take advantage of the variable hysteresis, set most values to the *HSTRT*, example, 4, and the remaining 1 to hysteresis end. The resulting configuration register values are as follows:

HEND = 4 (sets an effective end value of 1).

HSTRT = 2 (sets an effective start value of hysteresis end +3: 1 + 3 = 4).

Direct Mode

TMC5262 offers a special direct motor phase current control mode called the direct mode. This mode allows to individually command dedicated phase current values per each motor phase using SPI.

Setting *GCONF* register (0x0) bit-6 (*direct_mode*) switches into the direct mode.

Register XTARGET (0x2D) specifies the signed coil A current (bits 8...0) and coil B current (bits 24...16). In this mode, the current is scaled by the I_{HOLD} setting.

This mode must be used with SpreadCycle chopper only and does not support StealthChop+ operation.

Classic Constant Off-Time Chopper

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

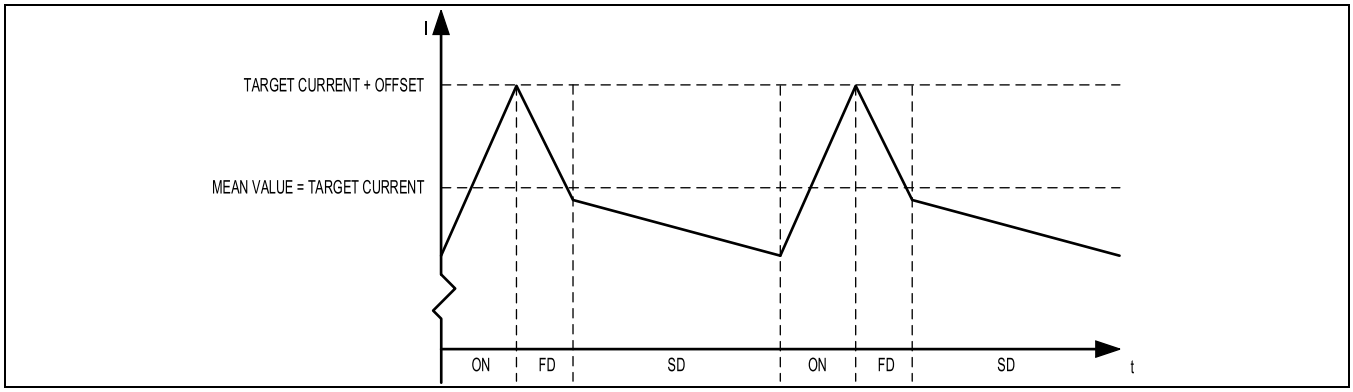


Figure 40. Classic Constant Off-Time Chopper with Offset Showing Coil Current

After tuning the fast decay time, the offset should be tuned for a smooth zero crossing. This is necessary because the fast decay phase makes the absolute value of the motor current lower than the target current (see *Figure 41*). If the zero offset is too low, the motor stands still for a short moment during the current zero crossing. If it is set too high, it makes a larger microstep. Typically, a positive offset setting is required for the smoothest operation.

Downloaded from Arrow.com.

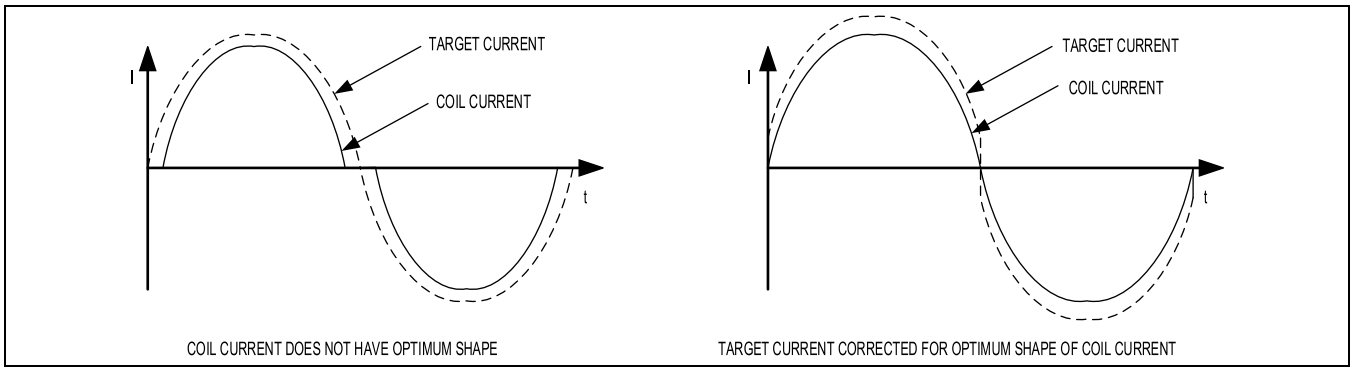


Figure 41. Zero Crossing with Classic Chopper and Correction Using Sine Wave Offset

Table 15. Parameters Controlling Constant Off-Time Chopper Mode

PARAMETER	DESCRIPTION	SETTING	COMMENT
TFD (*fd3* and *HSTRT*)	Fast decay time setting. With *chm* = 1, these bits control the portion of fast decay for each chopper cycle. Default = 5	0	Slow decay only
		1…15	Duration of fast decay phase
OFFSET (*HEND*)	Sine wave offset. With *chm* = 1, these bits control the sine wave offset. A positive offset corrects for the zero-crossing error. Default = 2	0…2	Negative offset: -3…-1
		3	No offset: 0
		4…15	Positive offset 1…12
disfdcc	Selects use of the current comparator to terminate the fast decay cycle. If the current comparator is enabled, it terminates the fast decay cycle if the current reaches a higher negative value than the actual positive value. Default = 0	0	Enable comparator termination of fast decay cycle
		1	End by time only

StallGuard2 Load Measurement

To also complement the SpreadCycle operation with a sensorless feedback system, the TMC5262 offers StallGuard2. StallGuard2 works in the SpreadCycle mode, while StallGuard+ is bound to the StealthChop+ operation.

Like StallGuard+, 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 using the *SGT* setting!

Downloaded from Arrow.com.

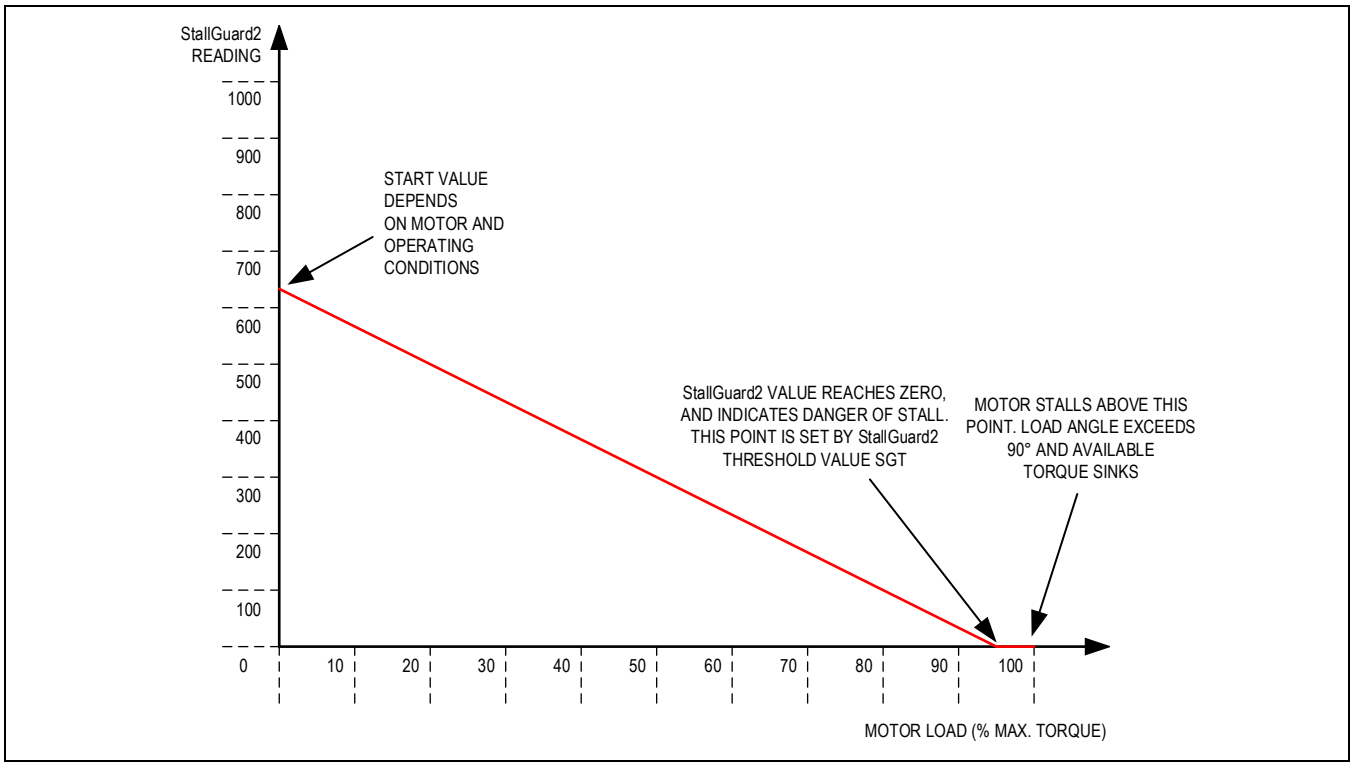


Figure 42. Function Principle of StallGuard2

Table 16. 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 (four fullsteps).	0	Standard mode (best for stall detection with hard stall)
		1	Filtered mode
STATUS WORD	**DESCRIPTION**	**RANGE**	**COMMENT**
SG_RESULT	This is the StallGuard2 result. A higher reading indicates less mechanical load. A lower reading indicates a higher load, and thus, a higher load angle. Tune the *SGT* setting to show a *SG_RESULT* reading of roughly 0 to 100 at 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 17. Procedure to Tune StallGuard

Downloaded from Arrow.com.

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*
2	Apply Load and Optimize *SGT*	Apply slowly increasing mechanical load to the motor. If the motor stalls before *SG_RESULT* reaches zero, decrease *SGT*. If *SG_RESULT* reaches zero before the motor stalls, increase *SGT*. A good *SGT* starting value is zero. *SGT* is signed, so it can have negative or positive values.
3	Enable Stop on Stall	Now enable *sg_stop* and make sure the motor is stopped safely whenever it is stalled. Increase *SGT* if the motor is stopped before a stall occurs. Restart the motor by disabling *sg_stop* or by clearing *event_stop_sg* in the *RAMP_STAT* register (write to clear).
4	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 temperature).

Optional procedure allowing the automatic tuning of *SGT*:

The basic idea behind the *SGT* setting is a factor, which compensates the StallGuard measurement for resistive losses inside the motor. At standstill and very low velocities, resistive losses are the main factor for the balance of energy in the motor, because mechanical power is zero or near to zero. This way, *SGT* can be set to an optimum at near zero velocity. This algorithm is especially useful for tuning *SGT* automatically within the application to give the best result independent of environment conditions, motor stray, etc.

Table 18. Procedure to Tune StallGuard Low Velocity

NUMBER	TASK	DETAIL
1	Operate Motor	Operate the motor at low velocity < 10 RPM (that is, a few fullsteps per second) and target operation current and supply voltage. In this velocity range, there is not much dependence of *SG_RESULT* on the motor load, because the motor does not generate significant back-EMF. Therefore, mechanical load does not make a big difference on the result.
2	Find Sensitive *SGT* Setting	Switch on *sfilt*. Now increase *SGT* starting from 0 to a value, where *SG_RESULT* starts rising. With a high *SGT*, *SG_RESULT* rises to the maximum value. Reduce again to the highest value, where *SG_RESULT* stays at 0. Now the SGT value is set as sensibly as possible. When you see *SG_RESULT* increasing at higher velocities, there is useful stall detection.
3	Enable Stop on Stall	*SG_RESULT* goes to zero when the motor stalls and the ramp generator can be programmed to stop the motor upon a stall event by enabling sg_stop in *SW_MODE*. Set *TCOOLTHRS* to match the lower velocity threshold where StallGuard delivers a good result to use *sg_stop*.

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

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.

Downloaded from Arrow.com.

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

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

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

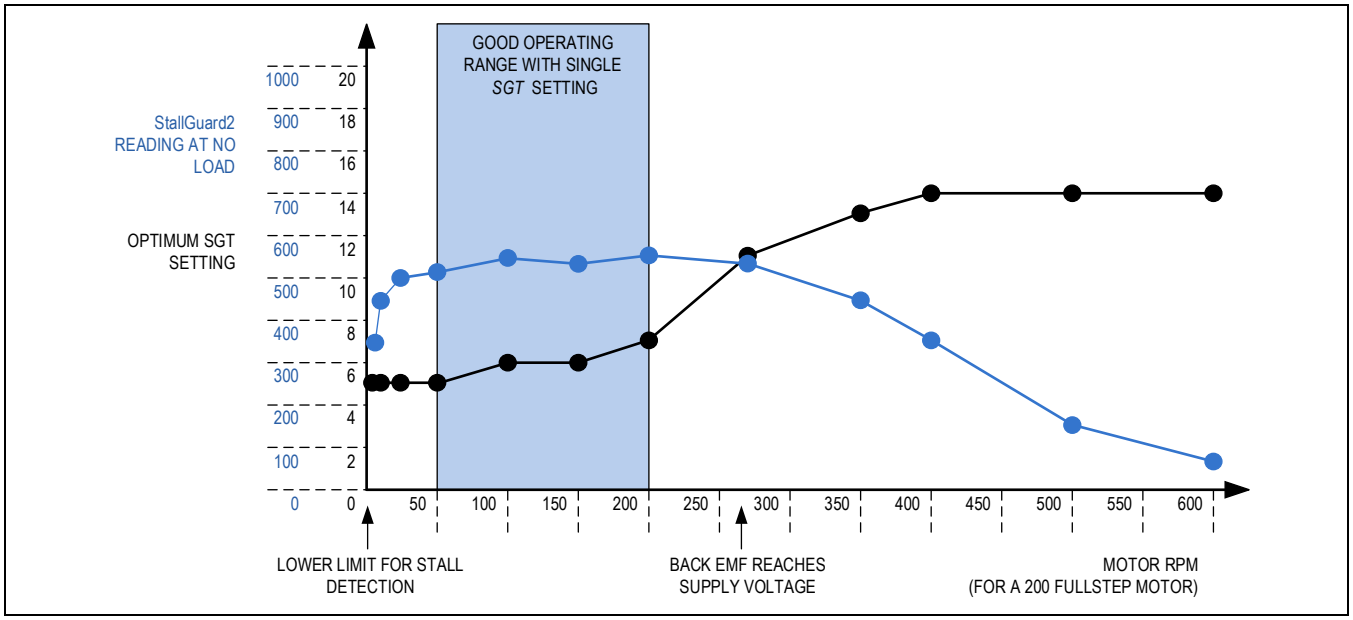


Figure 43. Example: Optimum SGT Setting and StallGuard2 Reading with an Example Motor

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

Downloaded from Arrow.com.

The filter should be disabled when a rapid response to increasing load is required and for best results of the sensorless homing using StallGuard.

Detecting a Motor Stall

For best stall detection, work without StallGuard2 filtering (*sfilt* = 0). To safely detect a motor stall, the stall threshold must be determined using a specific *SGT* setting. Therefore, the maximum load 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, etc.). The automatic tuning procedure described earlier can partially compensate for these effects. Other conditions also lead to the extreme settings of *SGT* and poor response of the measurement value *SG_RESULT* to the motor load. Consider operation in StealthChop+ if a stable StallGuard result is required over an increased operation range.

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. Depending on the actual chopper mode, CoolStep, respectively, CoolStep+ automatically uses StallGuard+ load measurement result in StealthChop+, or StallGuard2 in SpreadCycle mode. Coolstep requires StallGuard2 to be tuned before use. Like StallGuard2, CoolStep is limited to operation at a single target velocity, 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 19. CoolStep Critical Parameters

PARAMETER	DESCRIPTION	RANGE	COMMENT
SEMIN	4-bit unsigned integer that sets a lower threshold. If *SG_RESULT* goes below this threshold (indicating a large load), CoolStep increases the current to both coils. The 4-bit *SEMIN* value is scaled by 32 to cover the lower half of the range of the 10-bit *SG_RESULT* value. (The name of this parameter is derived from smartEnergy, which is an earlier name for CoolStep.)	0	Disable CoolStep
		1…15	Threshold is *SEMIN* x 32. 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	0…15	Threshold is (*SEMIN* + *SEMAX* + 1) x 32

Downloaded from Arrow.com.

	the current to both coils. The upper threshold is (*SEMIN* + *SEMAX* + 1) x 32.		Applications often work best with a setting of 0 or 1 (small hysteresis).

Figure 44 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* x 32, and CoolStep increases the current. When the load decreases, *SG_RESULT* rises above (*SEMIN* + *SEMAX* + 1) x 32, and the current is reduced.

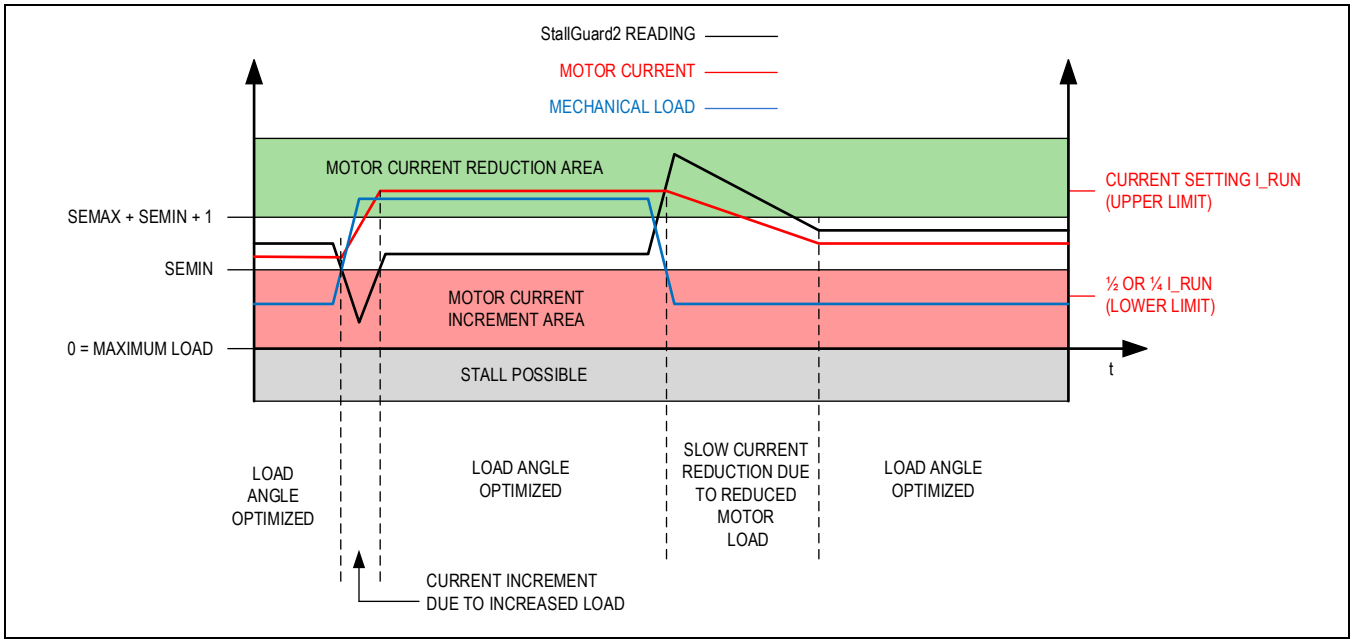


Figure 44. CoolStep Adapts Motor Current to the Load

Table 20. 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.	1… 2^{20} - 1	Specifies lower CoolStep velocity by comparing the threshold value to *TSTEP*.

Downloaded from Arrow.com.

	Hint: May be adapted to disable CoolStep during the acceleration and deceleration phases by setting the threshold velocity just slightly below *VMAX*.		
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 I_{RUN} 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 four. **Advice:** the most common and most beneficial use is to adapt CoolStep for operation at the typical system target operation velocity and to set the velocity thresholds accordingly. As acceleration and 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.

Downloaded from Arrow.com.

Integrated Current Sense

Non-dissipative current sensing is integrated in the TMC5262 (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*. With StealthChop+ operation, *CURRENT_RANGE_SCALE* is set to 3 (100%) when working with *CURRENT_RANGE* above 0 (more than 1A RMS). 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 and 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 21 shows the available maximum full-scale ranges with a typical R_{REF} of 12kΩ. It is not recommended to work with different values for R_{REF}.

Table 21. 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
1	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	

These values are valid for StealthChop+. SpreadCycle current is slightly lower with the same setting due to the default sine wave amplitude limited to 248/256 = 97% and because of the mean current in a hysteresis chopper that slightly lies below the target current due to the drop of the current in slow decay time. A more precise calculation for SpreadCycle operation is given in the following equation. It considers the sine wave amplitude, but generally using the formula for StealthChop+ is fine for applications combining both the chopper principles.

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

Downloaded from Arrow.com.

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

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

Attention: *IRUN* limitation: Selecting *IRUN* directly affects the use of the internal ADC range. The full range is used when working with *IRUN* ≥ 250 (StealthChop+), respectively, *IRUN* = 255 (SpreadCycle). A low current scaling (*IRUN*, respectively, *IRUN* scaled down by *COOL_CUR_DIV*) proportionally reduces the ADC reading, and thus, the range available for the current and angle regulation, respectively, for StallGuard measurement. To achieve the best possible performance, choose the lowest *CURRENT_RANGE* and with this the highest setting of *IRUN* delivering the required motor current. Avoid scaling down the current to less than 10% of the range.

The parameters given in *Table 22* allow to adapt the current scaling as well as the current ramp up and down.

Table 22. 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. For StealthChop+, the best results are yielded when keeping a 100% setting in combination with *CURRENT_RANGE* > 0.	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 250 (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(+). Default = 240	0…250 (0…255)	Scaling factor 0/256, 1/256, … 250/256, respectively, 1/256…256/256 for SpreadCycle.
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 x 200 x 2^{11} clock cycles, example, 100ms at 16MHz. Default = 7	0	Instant power down to I_{HOLD}.
		1…255	1 x 2^{11} … 15 x 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	0	Instant power up to IRUN.
		1...15	Delay per current increment step in multiples of

Downloaded from Arrow.com.

	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 down time of 4 × 200 × 64 clock cycles, example, 3.2ms at 16MHz. Default = 4		*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 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 quick *TPOWERDOWN* timing. It corresponds to a velocity of 64.	0	2^{20} clocks
		1	2^{18} clocks

Interpreting ADC Values

The internal current measurement in StealthChop+ uses ADCs to assess both motor coils' currents. The actual current can be read using the *ADC_I_A* and *ADC_I_B* registers as well as the RMS current per coil calculated from both given by *AMPL_MEAS*. The ADC scaling, and with it the conversion of the ADC values, depends on the current settings *CURRENT_RANGE* and *CURRENT_RANGE_SCALE*. These values are NOT available during SpreadCycle operation. The currents assessed for *ADC_I_A* and *ADC_I_B* are the momentary currents with a peak value exceeding the RMS value by SQRT(2).

$$RMSCurrent = 4.24A \times \frac{CURRENT_RANGE + 1}{4} \times \frac{CURRENT_RANGE_SCALE + 1}{4} \times \frac{AMPL_MEAS}{2000}$$

$$ADCCurrent = 6.0A \times \frac{CURRENT_{RANGE} + 1}{4} \times \frac{CURRENT_{RANGE_{SCALE}} + 1}{4} \times \frac{ADC_I_A \mid ADC_I_B}{2000}$$

Slope Control

The TMC5262 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 ringing of supply voltages and coupling to cables adjacent to the motor cables. The slowest settings with <400V/µs should be used in low operating voltage scenarios only. Especially when operating at voltage 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 in case slope-related issues cannot be solved by different means like the optimization of the PCB layout or cabling.

Downloaded from Arrow.com.

Velocity-Based Mode Control

The TMC5262 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, CoolStep+, StallGuard2, or StallGuard+ 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.

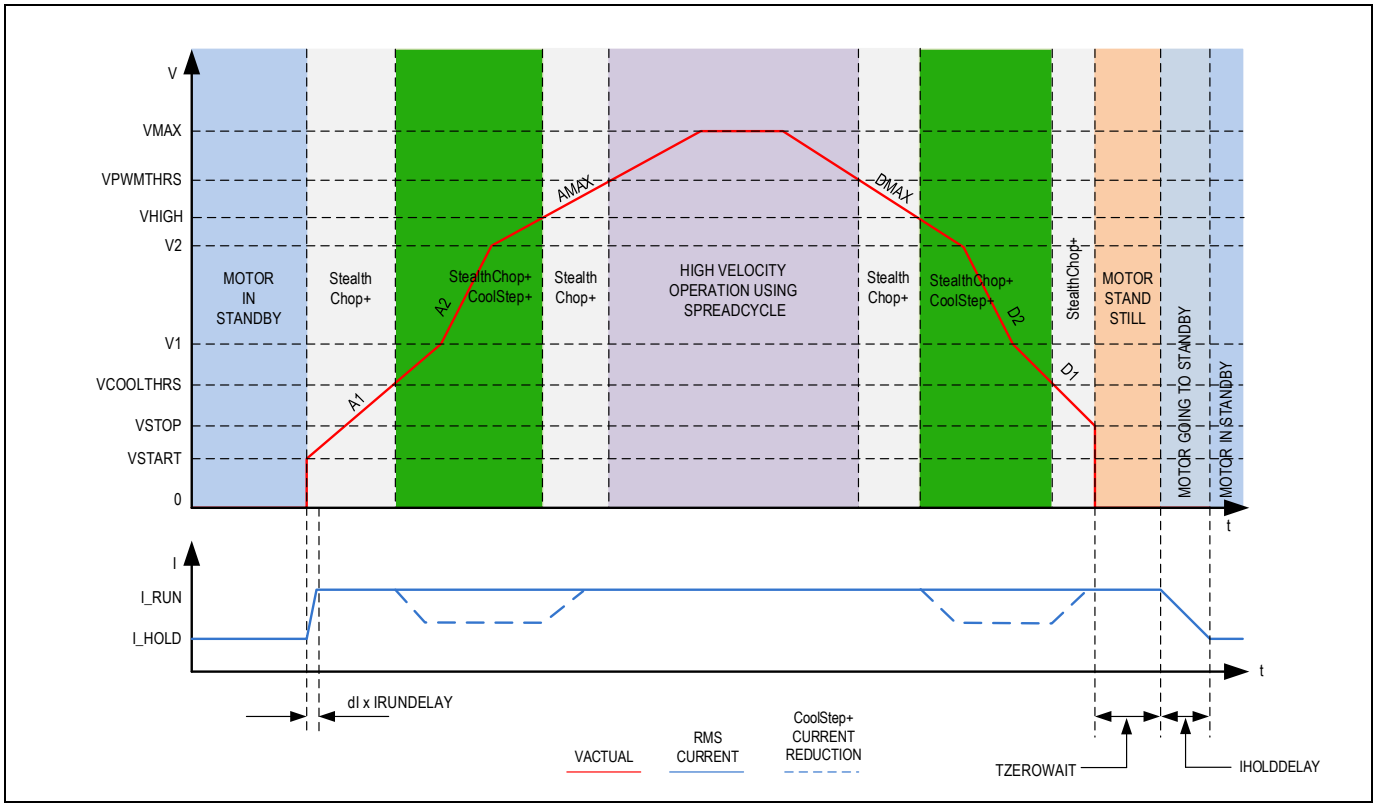


Figure 45. Choice of Velocity-Dependent Modes

Figure 45 shows the most important thresholds and required ordering. VPWMTHRS, VHIGH, and VCOOLTHRS are determined by the settings TPWMTHRS, THIGH, and TCOOLTHRS. The velocity is described by the time interval TSTEP between each 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 its 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 23. 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*).	0/1	Status bit, read-only

Downloaded from Arrow.com.

	Default/reset: 0		
TPOWERDOWN	This is the delay time after standstill detection (stst) of the motor--to-motor current power down. The time range is about 0 to 4 seconds (with f_{CLK} = 16MHz). Setting 0 is no delay, 1 is one clock cycle delay. Further increment is in discrete steps of 2^{18} clock cycles. Default: 10	0…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}.
TPWMTHRS	*TSTEP* < *TPWMTHRS* • StealthChop+ is disabled and driver switches over to SpreadCycle. Default: 0	0	StealthChop+ available in whole velocity range.
		1… 1048575	Setting to control the upper velocity threshold for operation in StealthChop2.
TCOOLTHRS	*TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: • StallGuard2 and CoolStep are enabled, if configured, respectively, StallGuard+ and CoolStep+. *TCOOLTHRS* ≥ *TSTEP* • Stop-on-stall is enabled and StallGuard(+) stall output signal is enabled (if configured). Default: 0	0	CoolStep/CoolStep+ off
		1… 1048575	Setting to control the lower velocity threshold for operation with CoolStep and StallGuard.
THIGH	*TSTEP* ≤ *THIGH*: • CoolStep, respectively, 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.
en_stealthchop	StealthChop+ enable flag (depending on velocity thresholds). Switch from off to on state while in standstill only. Default: 0	0	No StealthChop+.
		1	StealthChop+ active, if configured, and *TSTEP* > *TPWMTHRS*.

Downloaded from Arrow.com.

Ramp Generator

The ramp generator allows motion based on the target position or target velocity. It automatically calculates the motion profile, taking into account acceleration and velocity settings. The TMC5262 integrates a new type of ramp generator, which offers faster machine operation compared to the classical linear acceleration ramps. The EightPoint ramp generator allows adapting the acceleration ramps to the torque curves of a stepper motor. It uses three different acceleration settings, each for the acceleration phase and deceleration phase to allow for jerk minimized ramps.

Real-World Unit Conversion

The TMC5262 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[TMC5262] and a[TMC5262] are internal units of TMC5262. These are the values that must be written to the velocity/acceleration registers of TMC5262.

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

Table 24. Ramp Generator Parameters vs. Units

PARAMETER/SYMBOL	UNIT	DESCRIPTION
f_{CLK}	[Hz]	Clock frequency of the TMC5262; typically 16MHz.
s	[s]	Second
US	Microstep	
FS	Fullstep	
USC - microstep count	-	Microstep resolution in number of microsteps (refers to the number of microsteps between two fullsteps – normally 256).
FSC - fullstep count	-	Motor fullsteps per rotation, for example, 200
µstep velocity v[Hz]	Microsteps/s	v[Hz] = v[TMC5262] × (f_{CLK}[Hz]/2^{24})
µstep acceleration a[Hz/s]	Microsteps/s^2	a[Hz/s] = a[TMC5262] × 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[TMC5262])^2/a[TMC5262]/2^8 microsteps during linear acceleration ramp (assuming acceleration from 0 to v)
TSTEP, *Txxx_THRS*	-	*TSTEP* = $f_{CLK}/f_{256STEP}$ = $f_{CLK}/(f_{STEP}$ × 256/USC) = 2^{24}/(*VACTUAL* × 256/USC) The time reference for velocity thresholds is referred to the actual 1/256 microstep frequency ($f_{256STEP}$) of the step input respectively velocity v[Hz].
Ramp generator update rate	[Hz]	f_{UPDATE} = f_{CLK}/512 *VACTUAL* updates with this frequency.

In rare cases, the upper acceleration limit might impose a limitation to the application, 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.

Downloaded from Arrow.com.

Motion Profiles

Ramp Mode

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

Three different sets of acceleration and deceleration can be combined freely. The transition velocities V1 and V2 allow for velocity dependent switching between 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 that 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.

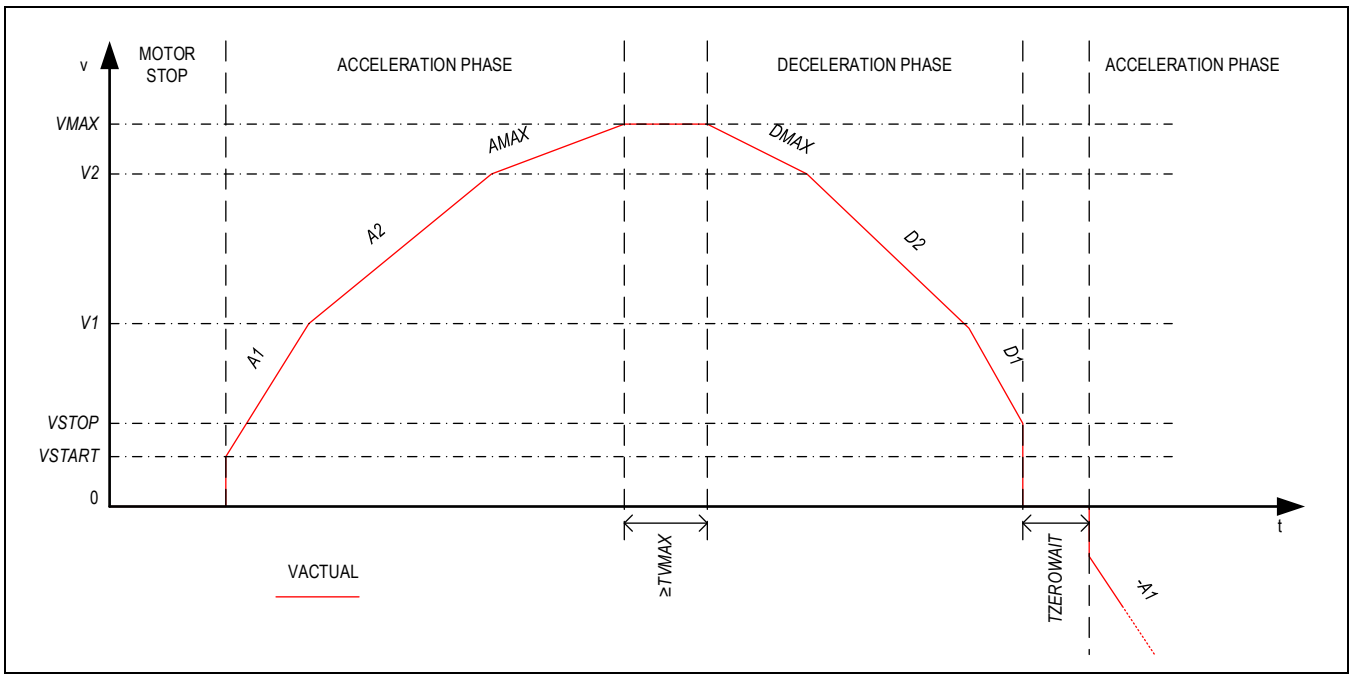


Figure 46. Ramp Generator Velocity Trace Showing Second Move into Negative Direction

Downloaded from Arrow.com.

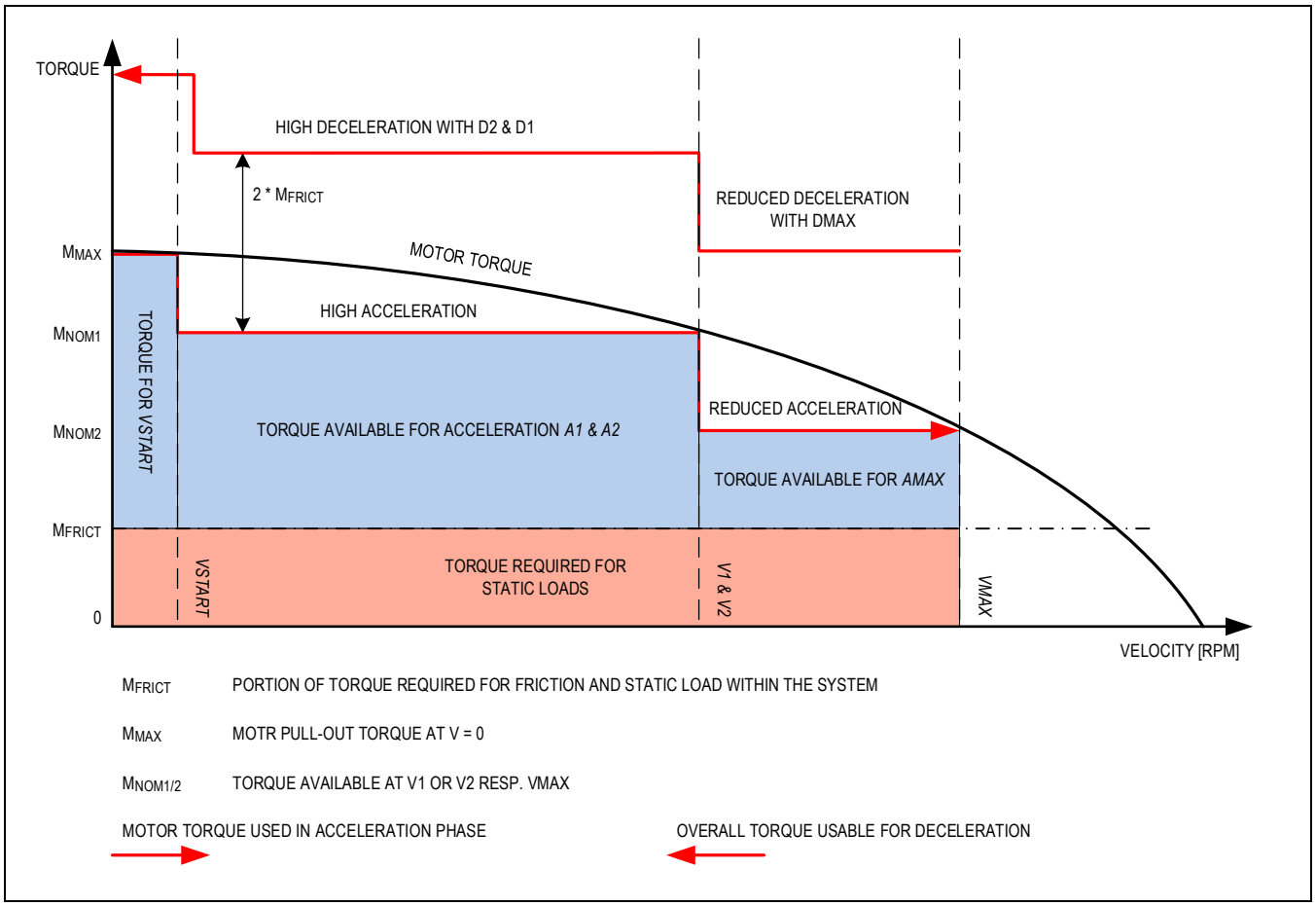


Figure 47. Illustration of Optimized Motor Torque Usage with the Ramp Generator

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.

Downloaded from Arrow.com.

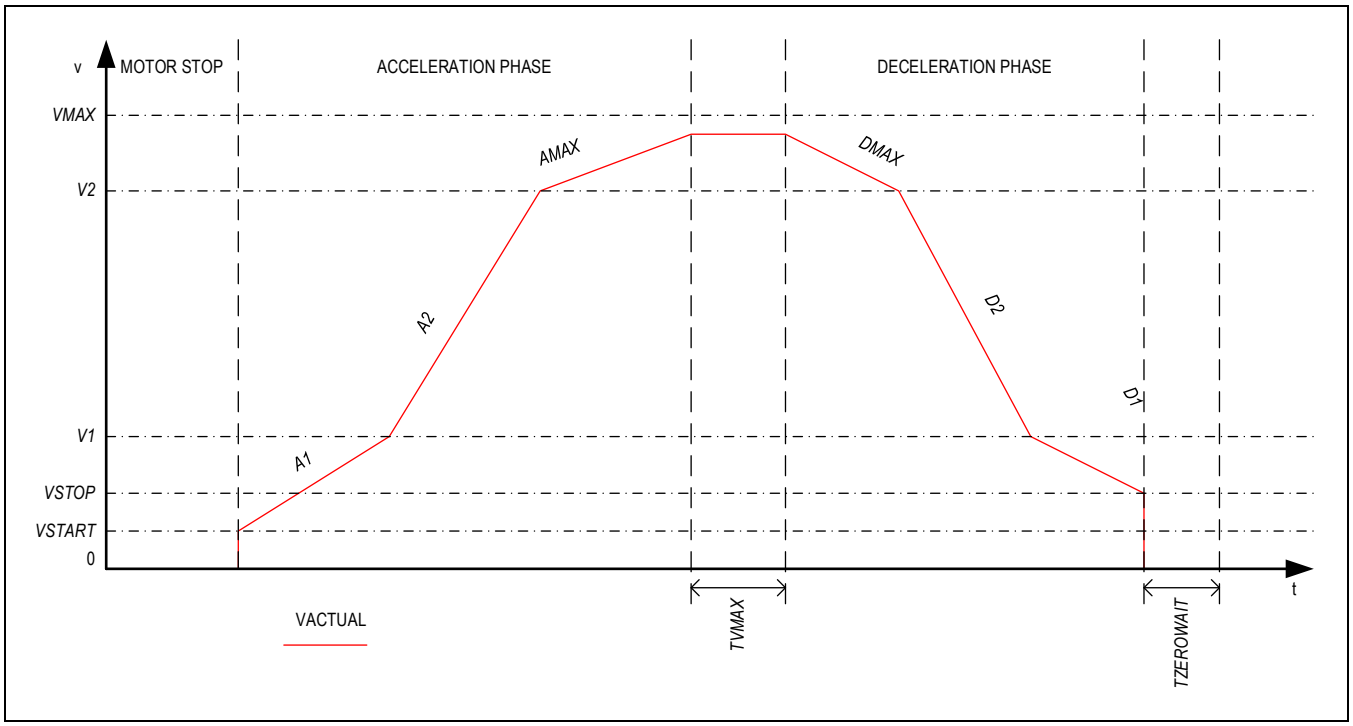


Figure 48. Eight-Point Ramp with VMAX Not Reached Due to Too Low Distance

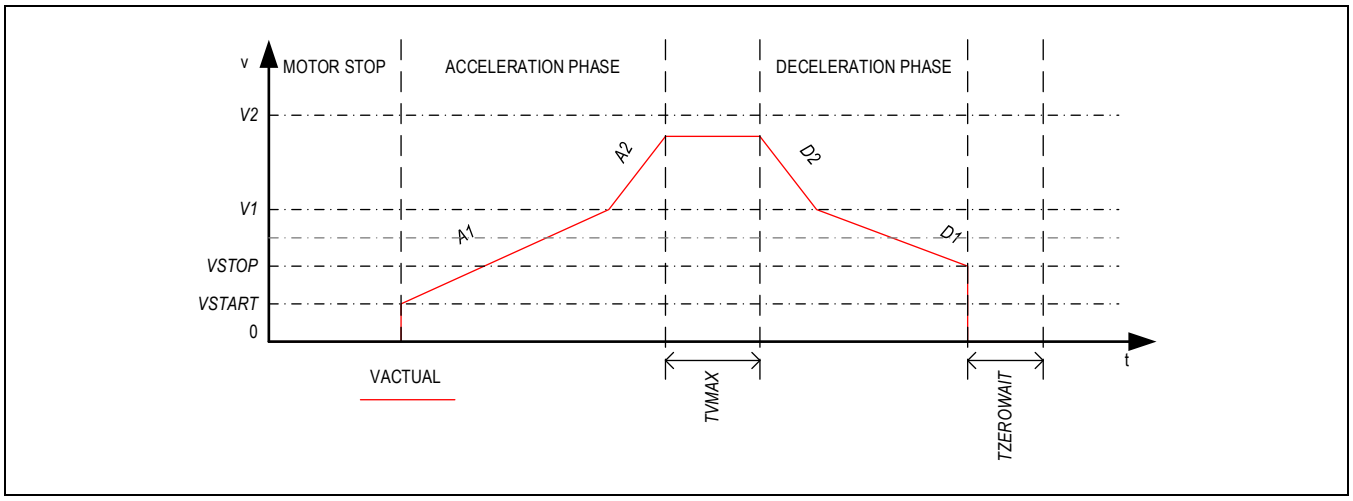


Figure 49. V2 Not Reached and No AMAX and DMAX Phase Due to Low Distance

Downloaded from Arrow.com.

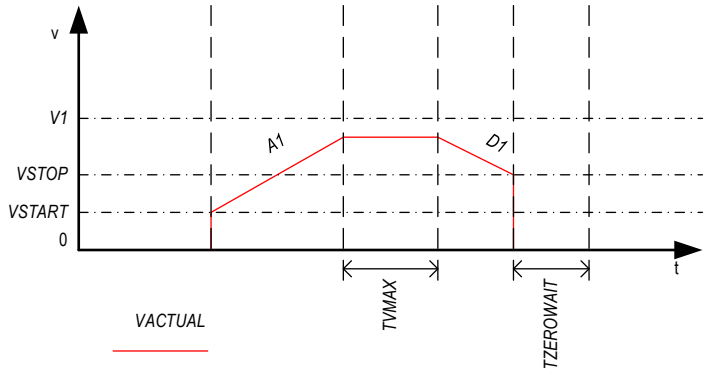


Figure 50. V1 Not Reached Due to Low Distance

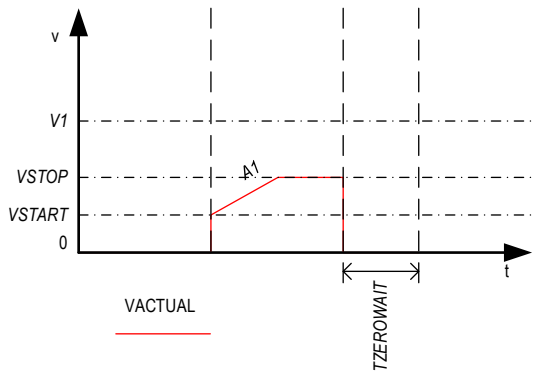


Figure 51. TVMAX Not Kept Due to Low Distance

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.

Downloaded from Arrow.com.

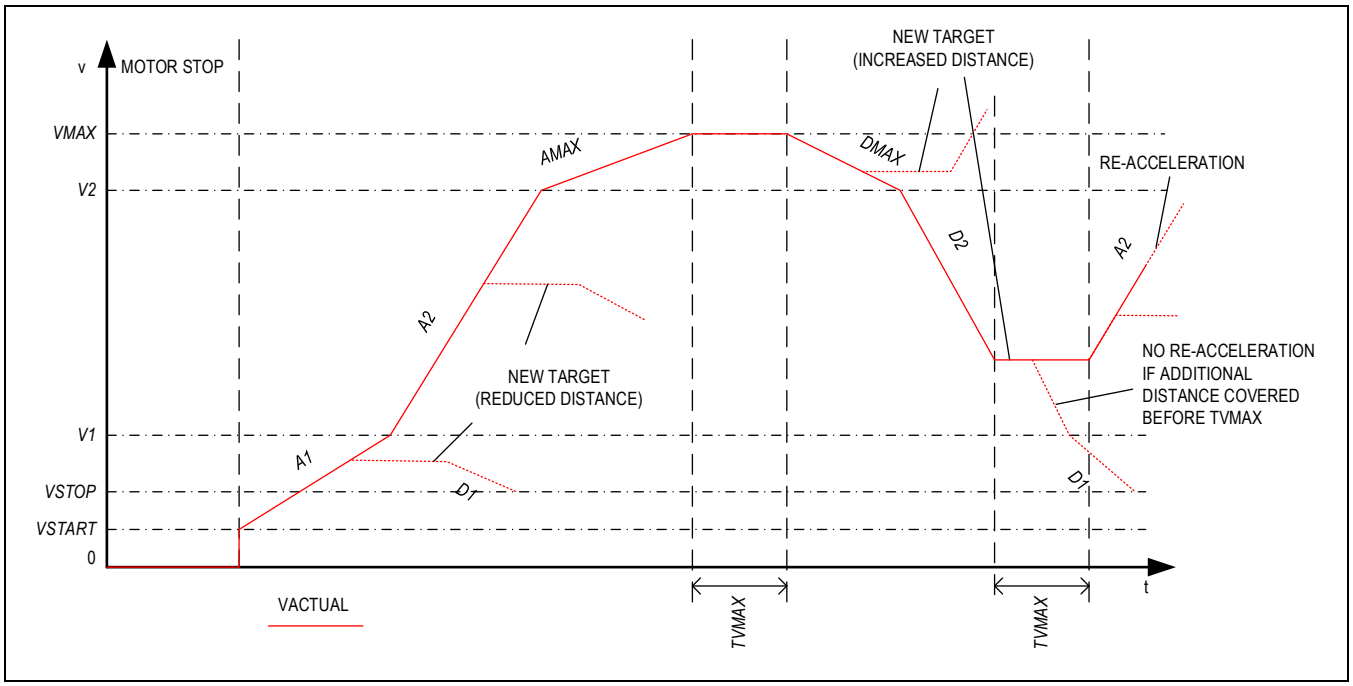


Figure 52. Eight-Point Ramp Examples with On-the-Fly Target Position Change

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,* and *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 motion using acceleration settings:

a) Switch to 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 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 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.

Downloaded from Arrow.com.

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 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 V_{STOP} only are 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, StallGuard+, 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 case 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.

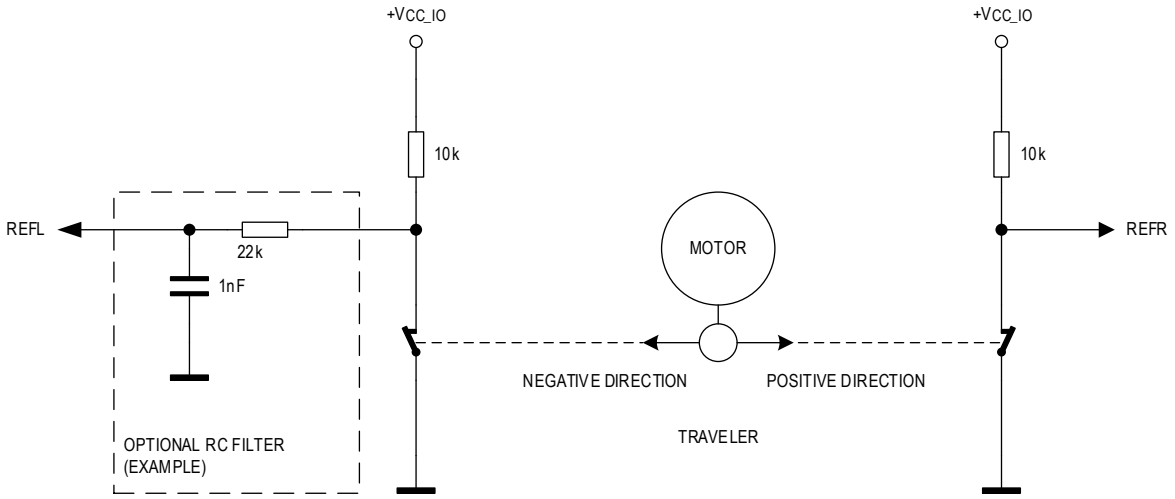


Figure 53. Using Reference Switches (Example)

Downloaded from Arrow.com.

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 TMC5262 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 25. 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 by 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 was 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 TMC5262 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 26. 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.

Downloaded from Arrow.com.

4	Write Absolute Position	After detection of the switch event, stop the motor, and subtract XLATCH from the actual position. (A detailed description of the required steps is in the homing procedure above).

Virtual Reference Switches

The TMC5262 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.

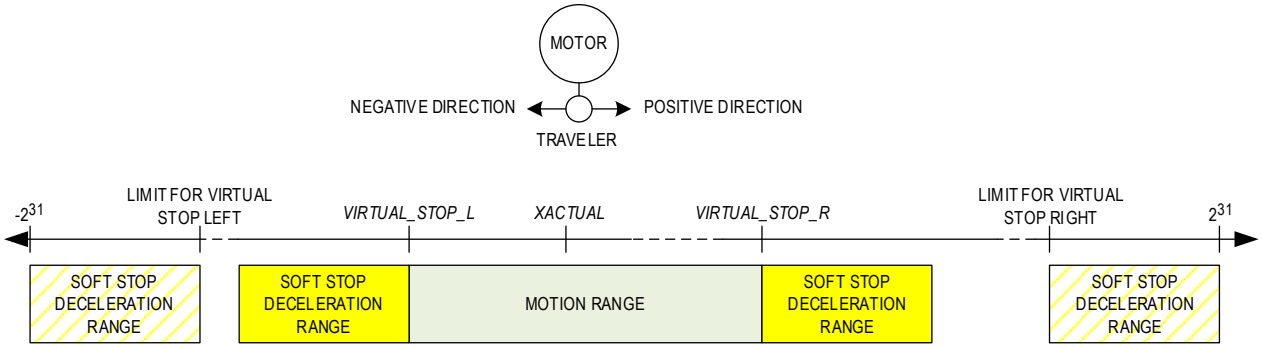


Figure 54. Virtual Stop Switches and Limit Visualization

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 TMC5262 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:

Downloaded from Arrow.com.

- 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 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 27. 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.

Downloaded from Arrow.com.

TriCoder – Back-EMF Sensorless Standstill Stepless Detection

The TriCoder function is a sensorless standstill stepless detection feature making use of the motor back-EMF (BEMF).

The BEMF decoder allows the detection of motor motion while the motor is disabled (that is, no active current is driven into the motor coils). This feature is especially useful for devices where a holding current cannot be applied due to powersaving requirements, but a certain amount of cogging torque or friction normally keeps the motor in position. The TriCoder tracks motor motion when the motor is turned by an external force. The result can be used to trigger a new homing sequence if the motor is moved, or to keep track of the number of steps and correcting for it later.

An alternative use is to employ the motor as an input device, example, for manual teach-in or for user interaction.

Considerations on sensitivity and precision:

- The principle requires a certain minimum BEMF voltage generated by the motor to detect motion. Depending on the motor, this required BEMF level is easily reached in the range of a few RPM, and thus, the occurrence of any motion can be easily detected.
- During the start or end of the motion, one or a few steps can be missed or counted incorrectly. The relevance of this must be checked with the actual motor and application mechanics.
- The system counts in multiples of one full step. By setting an encoder scaling factor fitting to the microstep resolution, the modified motor position can be directly tracked.

TriCoder BEMF Decoder Principle of Operation

The TriCoder BEMF decoder allows tracking a motor motion while the motor is disabled, or when the motor is stopped using passive braking using low-side drivers. To detect motor motion, the driver IC checks the voltage on the motor coils and compares it to a set of programmable hysteresis thresholds. *Figure 55* shows the principal of operation.

The detector circuit uses a combination of programmable hysteresis thresholds as well as bandwidth limiting to avoid false triggering, for example, due to voltage spikes coupled into the motor lines.

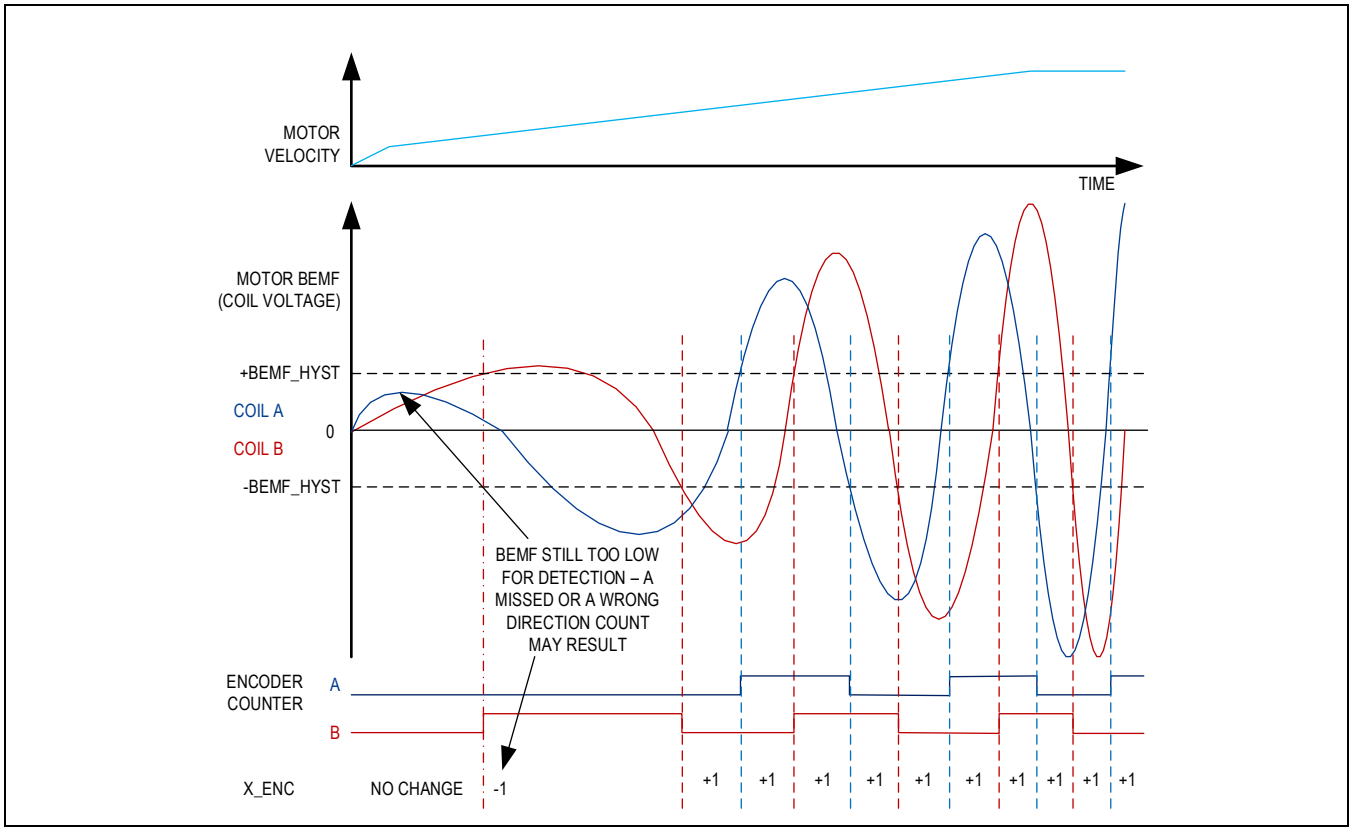


Figure 55. Detection of Motor Movement

Downloaded from Arrow.com.

Motor and Velocity Requirements for TriCoder Operation

Check the motor back-EMF voltage to determine the lowest velocity that can be detected using a certain hysteresis setting.

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

$$U_{BEMFopen} = \frac{HoldingTorque[Nm]}{2 \times I_{COILNOM}} \times \frac{2\pi \times Velocity[RPM]}{60}$$

In passive braking mode, the motor is electrically dampened by shorting out its coils. Due to this, encoder sensitivity to a certain motor velocity is reduced, as the generated back-EMF voltage first must be converted into a motor current, resulting from the motor coil resistance. This current again is transferred back into a voltage in the $R_{DS(ON)}$ of the low-side MOSFET switches within the driver. For best results in this mode, choose the lowest power stage current setting *CURRENT_RANGE* to increase low-side $R_{DS(ON)}$ vs. coil resistance.

$$U_{BEMFbrake} = \frac{HoldingTorque[Nm]}{2 \times I_{COILNOM}} \times \frac{2\pi \times Velocity[RPM]}{60}) \times \frac{RDSon}{RDSon + R_{COIL}}$$

where:

$R_{DS(ON)}$ is the resistance of the low-side MOSFETs, as taken from the electrical characteristics table $R_{DS(ON)}$ (LS).

Time to TriCoder Enable

To operate the TriCoder function, the motor must be in the freewheeling or passive braking mode with no remaining coil current. Depending on the preceding operation, a certain delay time may be required to ensure the remaining motor coil current has decreased. Following the detection of a motor stop (standstill, set standstill detection time using *fast_standstill* to 16ms or 65ms, as required by lowest operation velocity), the motor current ramps down to zero current, as controlled by the *IHOLDDELAY* settings. In case of a fast *IHOLDDELAY* setting (0 = instantaneous power down) and a highly inductive motor, an additional time of a few hundred microseconds to a few milliseconds may be required to decay the remaining motor current (*BEMF_BLANK_TIME*) by feeding it back to the power supply. Any remaining current might look like a motor BEMF and can otherwise lead to a false step detection.

Configuring the TriCoder

Enable TriCoder

These settings are required to automatically enable the TriCoder function, whenever the motor is in freewheeling mode.

- *nBEMF_ABN_sel*: set to 0 to disable the external encoder inputs. Motor must be at standstill (*stst* = 1).
- *PWM_CONF.FREEWHEEL*: set to 1 (freewheel) or 2 (passive brake using LS drivers).

In StealthChop+:

- *IHOLD*: set to 0 to enable freewheeling options during standstill. Freewheeling only is enabled when the motor is in standstill and *CS_ACTUAL* reaches 0.

In SpreadCycle:

- Set *TOFF* = 0 to disable the motor and allow freewheeling.

Downloaded from Arrow.com.

Operational Settings

Enable the TriCoder with the set of parameters described. Further, use *ENC_CONST* to match the *X_ENC* counter to the motor microstep resolution, example, 256 for 256 microstep setting. With this setting, each motor fullstep increases/decreases *X_ENC* by 256. *X_ENC* may be set to 0, respectively, to *XACTUAL* before the encoder is activated to either count the deviation from the original position, or to track the actual position during TriCoder action. Read out step difference/updated motor position while the encoder is in operation. An N channel is not available with the TriCoder function.

Table 28. TriCoder Control Parameters

PARAMETER	DESCRIPTION	SETTING	COMMENT
nBEMF_ABN_sel	Selection of encoder input Default/reset: 0	0	TriCoder: Encoder uses motor back-EMF.
		1	External incremental encoder using ABN inputs.
IHOLD	Motor standstill current. Set to 0 to enable TriCoder while motor is in standstill (works only in combination with StealthChop+). See *Table 22*. Also check *IHOLDDELAY* and *fast_standstill* to shorten the time to standstill detection and current reduction to 0.	0	TriCoder enabled during motor standstill.
		1…255	TriCoder not available.
FREEWHEELING	Selection of freewheeling or passive braking mode in combination with StealthChop+, while *CS_ACTUAL* = 0. (After reduction to I_{HOLD} = 0)	0	Chopper remains on. TriCoder not available.
		1	Motor goes to freewheeling. High sensitivity of TriCoder.
		2	Motor passive braking low-side. TriCoder available with reduced sensitivity due to shorting out motor BEMF with LS driver FETs.
		3	Motor passive braking high-side. TriCoder not available.
BEMF_HYST	Hysteresis setting for BEMF encoder. A higher hysteresis reduces noise suppression, but the lower velocity limit for motion detection goes up. Increase, if ripple on motor output leads to false detection of motion. Default: 0 (highest sensitivity)	0…7	10mV/25mV/50mV/75mV/100mV/150mV/200mV/250mV
BEMF_BLANK_TIME	Additional waiting time for the enable of TriCoder after current reduction to 0. Use to blank away spikes resulting from any remaining motor current or oscillation. Increase, if a false motion detection occurs when switching over to TriCoder.	0…255	Blank time in multiple of 2^14 × $1/f_{CLK}$
BEMF_FILTER_SEL	Limits the bandwidth of the BEMF filter for both the motor coil voltages to filter out disturbance. The theoretical count rate can be up to four times the bandwidth limit due to quadrature encoding. Set well above the expected maximum count rate.	0	Bandwidth 500Hz
		1	Bandwidth 1kHz
		2	Bandwidth 2kHz
		3	Bandwidth 4.3kHz

Downloaded from Arrow.com.

ABN Incremental Encoder Interface

The TMC5262 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.

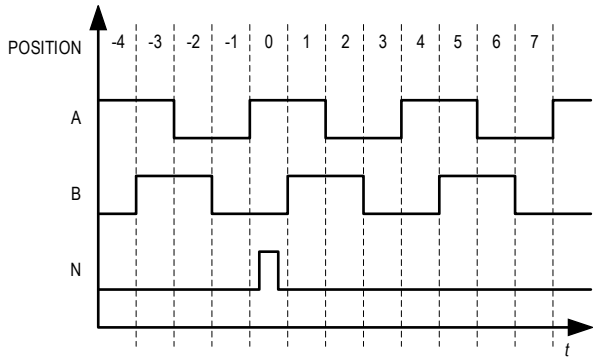


Figure 56. .Outline of ABN Signals of an Incremental Encoder

The Encoder Counter X_ENC

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

The Register ENC_STATUS

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

The Encoder Constant ENC_CONST

The encoder constant (or encoder factor) *ENC_CONST* is added to or subtracted from the encoder counter on each polarity change of the quadrature signals AB of the incremental encoder. The encoder constant *ENC_CONST* represents a signed fixed point number (16.16) to facilitate the generic adaption between motors and encoders. In 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 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)

Downloaded from Arrow.com.

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 motor parameters:

- USC = 256 microsteps
- FSC = 200 fullstep motor
- Factor = FSC × USC/encoder resolution

Table 29. Encoder Example Settings for a 200 Fullstep Motor with 256 Microsteps

ENCODER RESOLUTION	REQUIRED ENCODER FACTOR	COMMENT
200	256	
360	142.2222 = 9320675.5555/2^{16} (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

Downloaded from Arrow.com.

Sine Wave Lookup Table

The TMC5262 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

To minimize the required memory and the 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*, etc.

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.

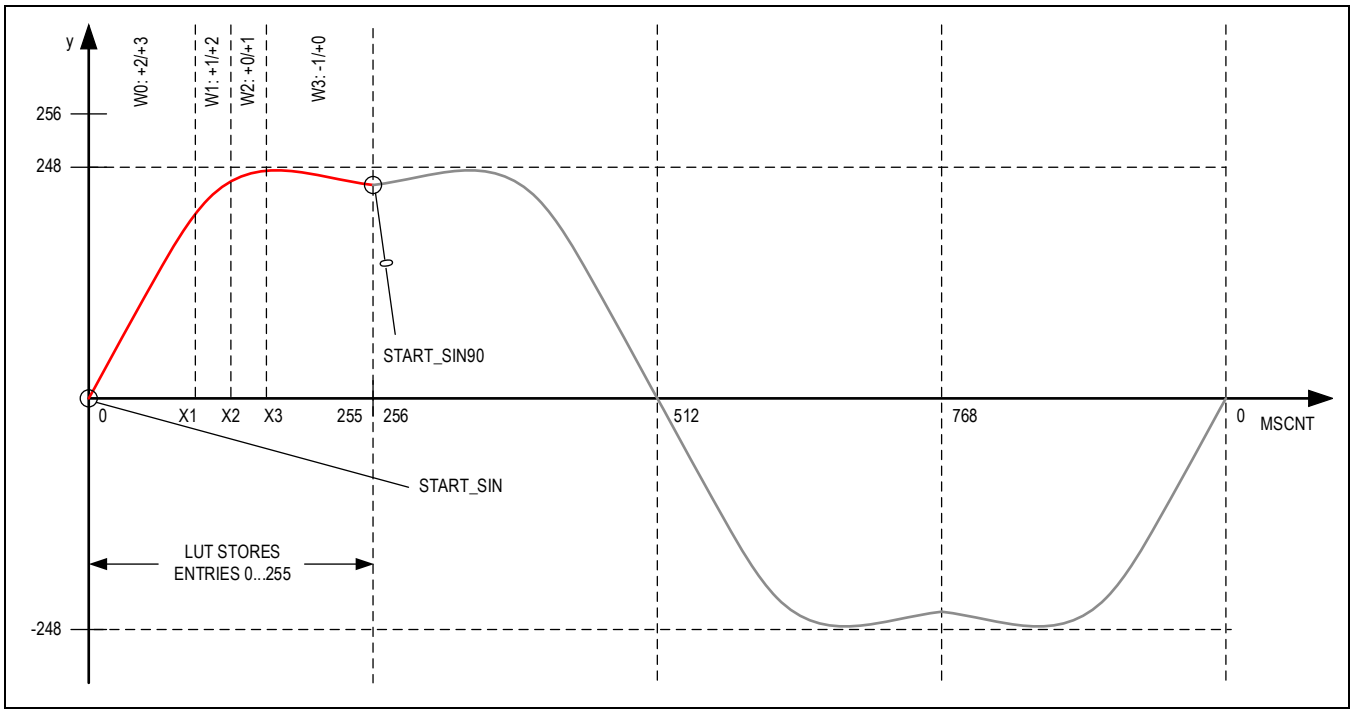


Figure 57. LUT Programming Example

When the microstep sequencer advances within the table, it calculates the actual current values for the motor coils with each microstep and stores them to the registers *CUR_A* and *CUR_B*. However, the incremental coding requires an absolute initialization, especially when the microstep table is modified. Therefore, *CUR_A* and *CUR_B* are initialized from *START_SIN90* and *START_SIN* whenever *MSCNT* passes zero.

Note: Wave shapes with StealthChop+

Downloaded from Arrow.com.

With StealthChop+, the sine wave amplitude always is regulated to an RMS value. This means the amplitude regulation tries to enforce a constant square-sum of *CUR_A* and *CUR_B*. Due to this, an adapted waveform should only be generated by modulating (symmetrically dragging and compressing) a sine wave, rather than using completely different wave shapes like a triangle wave.

Matching the Phase Shift to the Motor

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

- The starting value of the sinewave 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 cosine wave for the second motor coil must be programmed to *START_SIN90*. 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 90° for a two-phase motor.
- However, to adapt for motor tolerances, the 0-point of the cosine wave can be shifted to the left or right. In special cases, the phase shift can be modified from 90° (256 microsteps) to anywhere between 45° and 135° by adding a microstep offset in the range of -127 to +127 (register *OFFSET_SIN90*). However, compensating for motor tolerance might at most require moderate adaptations of a few steps for hybrid stepper motors and a few ten steps maximum for low quality motors. The required correction offset can be found out by carefully measuring each individual motor using a high-resolution encoder. In this case, *START_SIN90* must be adapted as fitting to the new zero-transition point.

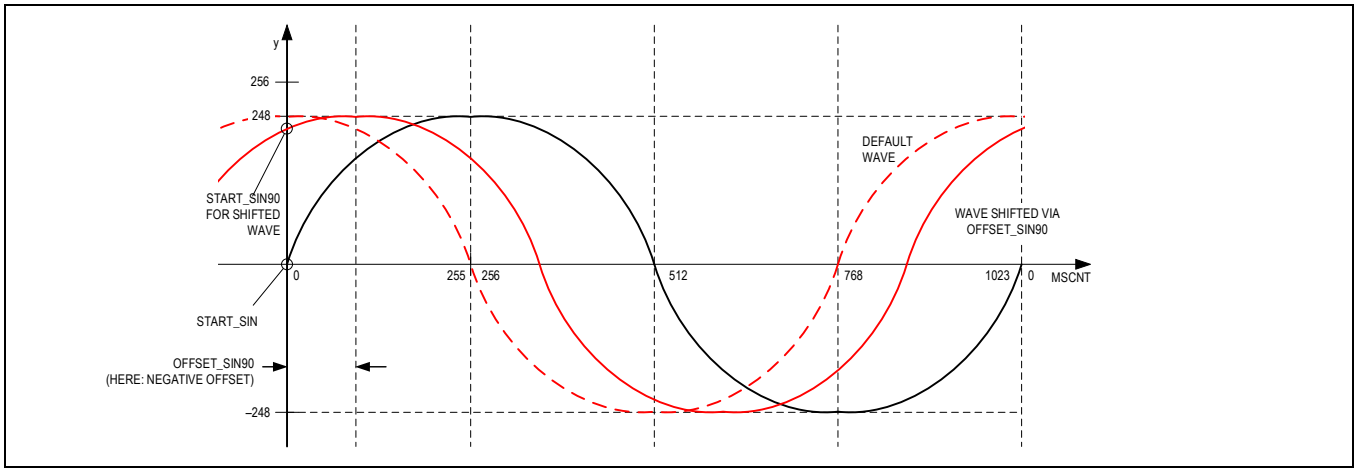


Figure 58. Shifting the Cosine Wave through OFFSET_SIN90

The default table fits best to most general hybrid stepper motors. It is a good base for realizing an own table. This is the initialization example for the reset-default microstep table.

MSLUT[0] = %10101010101010101011010101010100 = 0xAAAAB554

MSLUT[1] = %01001010100101010101010010101010 = 0x4A9554AA

MSLUT[2] = %00100100010010010010100100101001 = 0x24492929

MSLUT[3] = %00010000000100000100001000100010 = 0x10104222

MSLUT[4] = %11111011111111111111111111111111 = 0xFBFFFFFF

MSLUT[5] = %10110101101110110111011101111101 = 0xB5BB777D

MSLUT[6] = %01001001001010010101010101010110 = 0x49295556

MSLUT[7] = %00000000010000000100001000100010 = 0x00404222

MSLUTSEL = 0xFFFF8056:

Downloaded from Arrow.com.

X1 = 128, X2 = 255, X3 = 255

W3 = %01, W2 = %01, W1 = %01, W0 = %10

MSLUTSTART = 0x00F70000: START_SIN_0 = 0, START_SIN90 = 247

To optimize the motor phase shift, check the motor rotation using a high-resolution encoder and increment *OFFSET_SIN90* or decrement to reduce asymmetric behavior within a 360° periodicity. Repeat until the best match is found and counter check/take mean value when running in the reverse direction.

Be sure to correctly enter the start value for the cosine wave *START_SIN90* to avoid a discontinuity in the wave shape. When using the default wave, an offset of -10 to +9 requires *START_SIN90* = 247; up to -17 or +17 requires *START_SIN90* = 246. *START_SIN* is always 0. For different wave shapes, these values can be determined by moving to the offset position and checking *CUR_A*, while *OFFSET_SIN90* is still at 0.

Downloaded from Arrow.com.

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 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 I_{HOLD}, *IHOLD_DELAY*, and StealthChop+ standstill options.

External Reset and Sleep Mode

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

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

Downloaded from Arrow.com.

Diagnostics and Protections

The TMC5262 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 TMC5262 allows the measurement and read out of the chip temperature as well as feedback on the motor phase winding temperature by evaluating the change of coil resistance (see *R_COIL* measurement). Driver temperature measurement allows preventive action for extreme thermal conditions, for example, by slowing down the motor to reduce its load or by interrupting motion to allow the cool-down of the motor and driver.

For improved system reliability and overall circuit protection, the TMC5262 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 TMC5262 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-OCSI DAC channel 0.
- The diagnostic output pin DO1 is a functionally shared pin with RT-OCSI DAC channel 1.

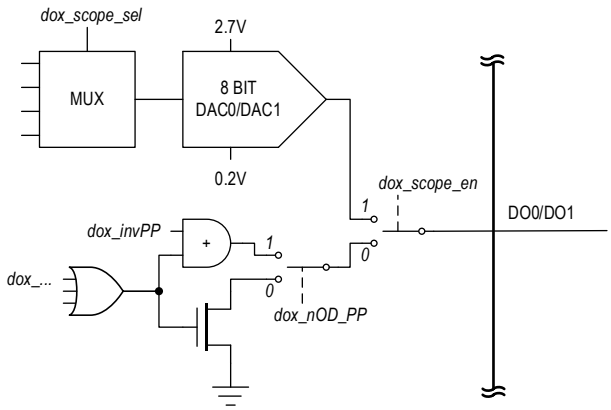


Figure 59. Schematic of DO0 and DO1 Outputs

Real-Time On-Chip Scope Interface (RT-OCSI)

The *real-time on-chip scope interface* (RT-OCSI) facilitates tuning and debugging an application within the application by providing test points for two scope probes and without the need to visualize real-time data on a PC by accessing the SPI. The TMC5262 visualizes real-time data on an oscilloscope using two internal, independent DAC outputs that can be mapped to the diagnostic outputs DO0 and DO1.

With this, RT-OCSI supports StealthChop+, StallGuard+, and CoolStep+ tuning as well as tuning for StallGuard2 and CoolStep by directly monitoring two operation parameters at a time like motor current scaling and StallGuard result. With

Downloaded from Arrow.com.

a four-channel scope, these parameters can be monitored in parallel to physically accessible voltages and currents like the motor current.

Both DAC channels allow flexible assignment of different internal parameters. To overcome limitations imposed by the scope and DAC resolution or electrical noise floor in a stepper application, a selection of bit ranges is available for registers with more than 12-bits.

Set *do0_scope_en* and *do1_scope_en* in the *DO_SCOPE_CONF* register (0x03) to enable the DAC output on one or both diagnostic outputs DO0 and DO1.

To support a fast analog settling to digital zero and to full scale, the DAC outputs provide a 2.5V swing within the voltage range of 0.2V to 2.7V by adding an internal zero-offset of 0.2V. For signed values, the 0-value, thus, is mapped to (2.7V + 0.2V)/2 = 1.45V by adding an offset of 128.

Testing of the DAC output or using the outputs to provide external analog voltages is possible by selecting index 0x1C for USER_VALUE.

Each DAC output allows the selection of one of the following internal parameters using the *DO0_SCOPE_SEL* and *DO1_SCOPE_SEL configuration parameters* in the *DO_SCOPE_CONF* register (0x03) through its index number.

Table 30. Real-Time On-Chip Scope Interface (RT-OCSI) DAC Configuration Options

INDEX	SIGNAL	DESCRIPTION	USE CASE
0x0	*ADC_I_A*[11..5] + 128	Current coil A (signed w. offset)	Monitor coil A/B currents. Check current PI regulator.
0x1	*ADC_I_B*[11..5] + 128		
0x2	*RCOIL_A*[11..4]	Coil A resistance automatically determined by driver.	Check update time, low velocity behavior, and rough value of coil resistance measurement A/B.
0x3	*RCOIL_B*[11..4]		
0x4	*UL_A*[11..5] + 128	Calculated inductance voltage (lagging current by 90°) on coil A/B (signed w. offset).	-
0x5	*UL_B*[11..5] + 128		
0x6	*COOLSTEP_LOAD_RESERVE*	Actual value of target load reserve	Monitor current-dependent load reserve. Compare to actual *SGP_RAW* value for tracking CoolStep.
0x7	*ADC_TEMPERATURE*	Actual value of temperature ADC	Monitor IC temperature.
0x8	*ANGLE_MEAS*[9..2]	Electrical angle calculated from coil currents	Monitor calculated current angle and compare against *MSCNT* to check angle regulator.
0x9	*SGP_RAW*[7..0]	Raw value of StallGuard+ Clipped to range 0..255 (SGP_FILT_EN is not applied).	Monitor motor load and load measurement stability; compare to *COOLSTEP_LOAD_RESERVE.*
0xA	*ANGLE_CORR_CALC*	Result of angle regulator	Monitor corrective action of angle regulator.
0xB	*AMPL_MEAS*[11..4]	Effective motor current calculated from ADC_I_A and ADC_I_B. 100% current corresponds to ½ full scale (2048) as target value.	Check response of current regulator and current amplitude. Visualize together with *CS_ACTUAL* using half V/div scaling.
0xC	*PWM_CALC*[11..4]	Result of current regulator	Check PWM duty cycle and check for headroom of drive voltage.
0xD	*CS_ACTUAL*	Actual current scale as resulting from *IRUN*, *IHOLD,* and CoolStep.	Check CoolStep current profile vs. load, check headroom for torque.
0xE	*UBEMF_ABS*[11..4]	Height of back-EMF voltage	Check height to assess expected quality of back-EMF-based StallGuard+ signal.
0xF	*SG_RESULT*[9..2]	Actual StallGuard2/StallGuard+ value (unsigned, range 0…1023)	Monitor for tuning StallGuard settings.

Downloaded from Arrow.com.

0x10	*SGP_RESULT*[9..2]+128	Actual StallGuard+ value (signed, range -512…+511)	Check filtered vs. unfiltered signal and also monitor generatoric range.
0x11	*CUR_A*[8..1]+128	Actual microstep wave table (coil voltage) with full amplitude. The coil currents (ADC_I_x values) lag voltages as given by CUR_x due to motor inductivity and BEMF.	Check microstep table waveform.
0x12	*CUR_B*[8..1]+128		
0x13	DAC_A	Input of internal DAC for current regulation in SpreadCycle (showing chopper operation). The DACs receive positive values (on-phase) and negative values (fast decay phase).	Monitor SpreadCycle target current (showing hysteresis chopper).
0x14	DAC_B		
0x15	*ANGLE_ERROR*	Angle difference between *MSCNT_SNPSHT* and *ANGLE_MEAS*	Monitor angle regulator regulation error. (**Note:** Signal jitters due to time lag of *ANGLE_MEAS* for ~12µs with each update of *MSCNT_SNPSHT*).
0x16	*MSCNT*[9..2]	Microstep counter (increments upon each step as defined by *MRES*).	Monitor microstep counter position.
0x17	*MSCNT_SNPSHT*	*MSCNT_OFFSET* sampled each 32 chopper cycles for angle calculation.	Monitor any parameter vs. position where wave is sampled. The position is given as a saw-tooth wave.
0x18	*MSCNT_OFFSET*	Interpolated microstep counter fitting to *MSCNT* but with microstep interpolation (without offset calculated by angle regulator).	Monitor any parameter vs. position in current wave when using reduced microstep resolution. The position is given as a saw-tooth wave. (Reference for *ANGLE_MEAS*)
0x19	*TSTEP_VELOCITY*[20...13]	Motor velocity calculated from *TSTEP* with different scaling to monitor lower/higher range (unsigned).	Monitor any parameter vs. velocity. Value overflows from full scale to 0 when exceeding the selected bit range.
0x1A	*TSTEP_VELOCITY*[18...11]		
0x1B	*TSTEP_VELOCITY*[16...9]		
0x1C	USER_VALUE	Output DAC voltage as defined by *DO_CONF*[7..0] for DO0, respectively, *DO_CONF*[20..13] for DO1 (unsigned value).	Test of DAC interface, output of user-defined value.
0x1D	BEMF voltage A/B[13…7] + 128	Internal BEMF calculation result signed vector in different scaling; not clipped to range limits BEMF for coil A on DO0, for coil B on DO1	Monitor calculated SGP BEMF voltage; compare to current waves. Select the fitting range giving sufficient signal for monitoring, starting from least sensitive range (0x1D).
0x1E	BEMF voltage A/B[11…5] + 128		
0x1F	BEMF voltage A/B[9…3] + 128		

Downloaded from Arrow.com.

RT-OCSI Examples of Use

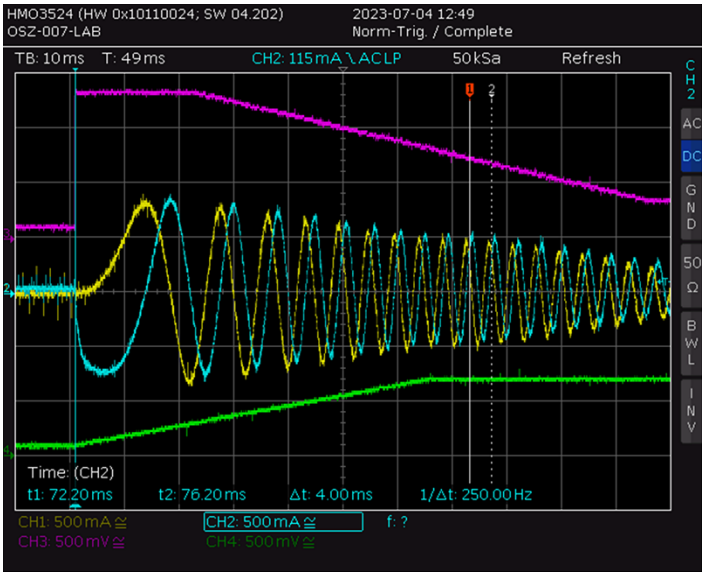


Figure 60. Monitoring CoolStep+ Using the RT-OCSI

The example in *Figure 60* shows how to use the RT-OCSI to monitor CoolStep+.

Yellow and blue traces: motor current measured with current clamp showing motor start from standstill (using full *IRUN* current) with CoolStep+ enabled ~ 22ms after start of the motion and decreasing current as controlled by the CoolStep+ settings.

Green trace: 0x1A *TSTEP_VELOCITY* motor velocity shows acceleration.

Purple trace: 0xD *CS_ACTUAL* current scale showing current reduction controlled by CoolStep+.

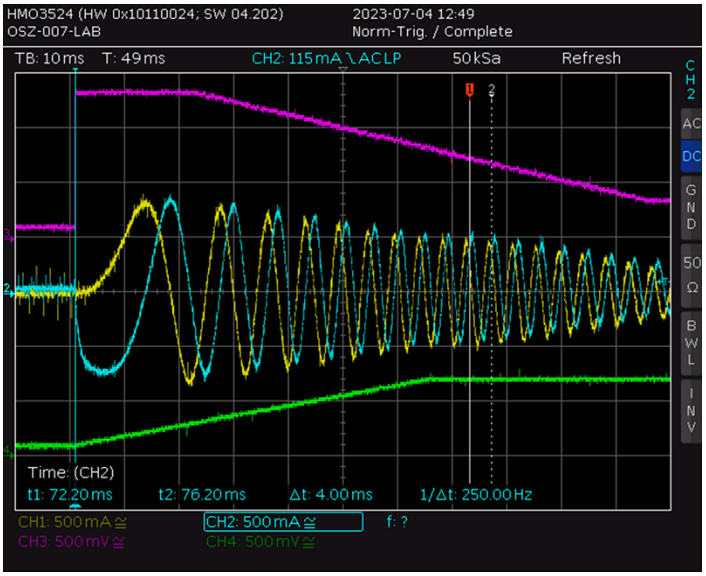


Figure 61. Monitoring the Current PI Regulator Using the RT-OCSI

The example in *Figure 61* shows how to use the RT-OCSI to monitor the current PI regulator.

Yellow and blue traces: motor current measured with current clamp showing motor start from standstill.

Downloaded from Arrow.com.

Green trace: 0xC *PWM_CALC* current regulator output showing adaption of duty cycle to ramp up current.

Purple trace: 0xB *AMPL_MEAS* motor current amplitude measured and calculated from both coils showing effective motor current ramp up to 90% in 4ms.

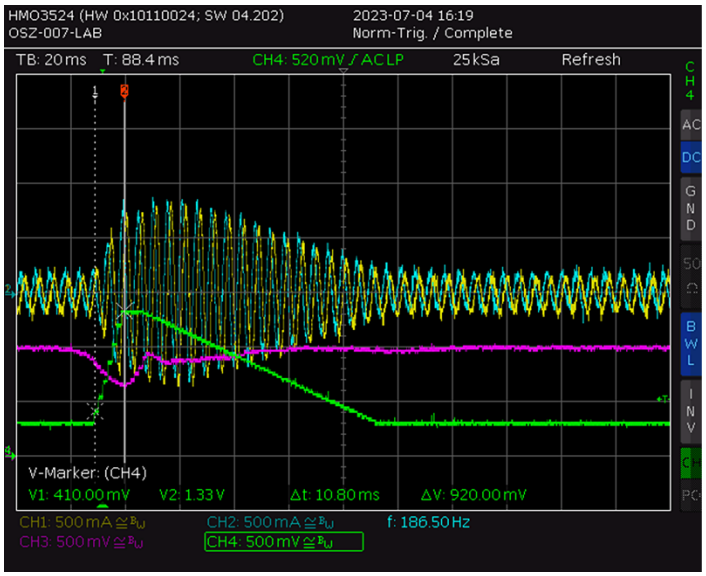


Figure 62. Monitoring the CoolStep+ PI Regulator Using the RT-OCSI

The example in *Figure 62* shows how to use the RT-OCSI to monitor the CoolStep+ PI regulator.

Yellow and blue traces: motor current measured with current clamp showing motor operation at reduced current with a load peak on the axis starting at t = 20ms.

Green trace: 0xD *CS_ACTUAL* showing current scaling going into the current regulator with reaction to a load impact and current decrement slope.

Purple trace: 0x10 *SGP_RESULT* showing StallGuard+ load measurement going from no load/near no load to 0 (half DAC range) at peak of load impact.

Overcurrent Protection

Overcurrent protection (OCP) protects the device against short circuits to the rails (supply voltage and ground) and between the outputs (OA1, OA2, OB1, and OB2).

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*. Anyway, a change in *CURRENT_RANGE* requires remeasurement of the StealthChop+ coil resistance.

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 H-bridge immediately is disabled.

To avoid false triggering upon an ESD event, the short protection retries three times in consecutive chopper cycles, before a fault flag (*s2ga*, *s2gb*, *s2vsa*, *s2vsb* in the *DRV_STATUS* register) 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 software.

Downloaded from Arrow.com.

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 TMC5262 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 TMC5262 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 TMC5262 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 eight 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 TMC5262 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:

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

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

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.

Motor Temperature Measurement

As StealthChop+ requires the motor coil resistance for its function, it allows relatively precise measurement of the motor coil resistance. By monitoring the change of coil resistance from its 25° initial value, the motor coil temperature can be determined. For a copper coil, the thermal coefficient is 0.39% per Kelvin. However, the coil resistance measurement includes the resistance of the driver's output power stage, which also rises with driver temperature. As long as the motor's coil resistance is high compared to the driver MOSFET resistance, the error from this component is low. With a very low resistive motor, the correction factor should be determined by measurement.

$$T_{Motor} = T_{REF} + \frac{R_{COIL} - R_{REF}}{R_{REF} \times \alpha}$$

α: Temperature coefficient of coil material, 0.39%/K for copper.

T_{REF}, R_{REF}: Reference temperature and resistance of motor coil at reference temperature.

Downloaded from Arrow.com.

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 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 TMC5262 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 roughly 2.45kHz, 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 has to 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:

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

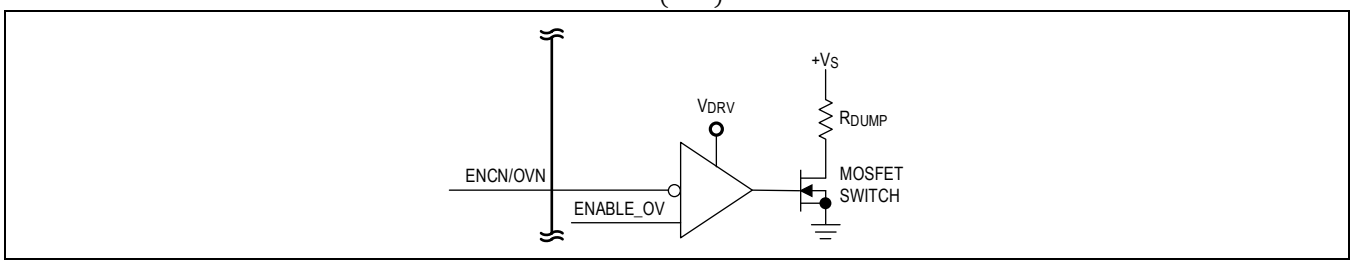


Figure 63. Brake Chopper Circuit Example

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

The TMC5262 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

Downloaded from Arrow.com.

is protected against spurious triggering, for example, by ESD discharges, by retrying three times before switching off the motor.

Once a short condition is safely detected, the corresponding driver bridge is switched off and the *s2ga* or *s2gb* 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 TMC5262 detects open-load conditions by checking if it can reach the desired motor coil current. This way also, undervoltage conditions, high motor velocity settings, or short and overtemperature conditions may trigger the open-load flag. In motor standstill, open load cannot be measured as the coils might eventually have zero current.

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

Undervoltage Lockout Protection

The TMC5262 features an UVLO protection for V_S, V_{CC_IO}, and the charge pump.

UVLO condition on V_S is triggered below 4.15V.

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

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.

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 TMC5262 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.

Downloaded from Arrow.com.

Clock Oscillator and Clock Input

Internal PLL Block

The TMC5262 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, and a higher frequency of 80MHz for operation of the StealthChop+ PWM.

The PLL must be correctly configured and run before the operation of the IC. While the PLL is disabled or in a fault mode, 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 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.

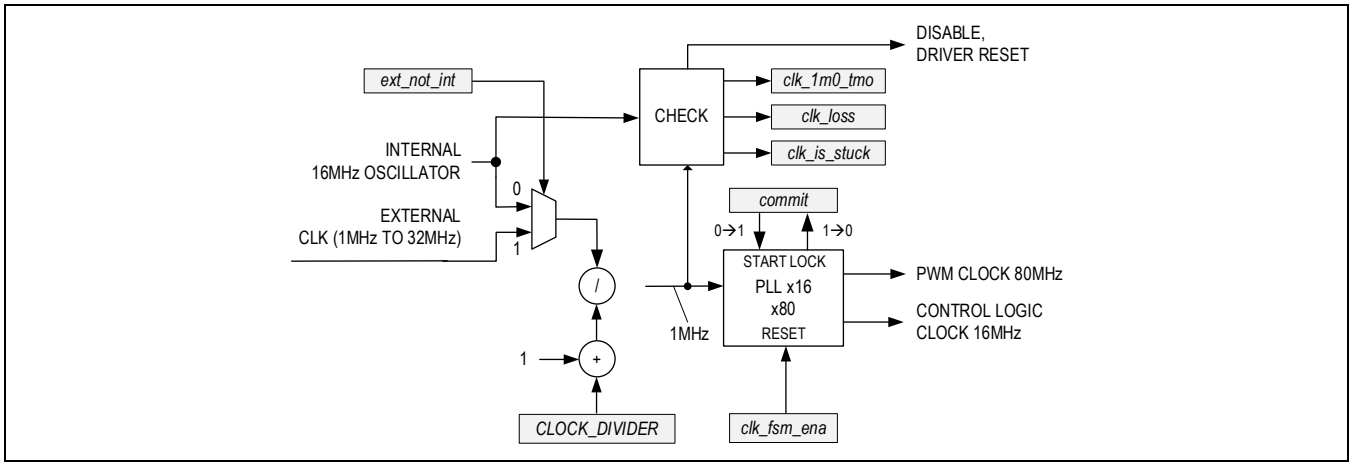


Figure 64. Block Diagram of PLL

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 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.

Downloaded from Arrow.com.

Table 31. 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 commited 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 commited to the PLL configuration.	0	Normal operation
		1	PLL is in error state. Clock signal has fixed polarity.

Downloaded from Arrow.com.

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 between multiple motors is required.

Using an External Clock

For exact timing of the motor motion using the internal motion controller, 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 and to an 80MHz clock for the StealthChop+ PWM unit.

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).

Downloaded from Arrow.com.

Quick Configuration Guides

This guide is meant as 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 functionality 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.

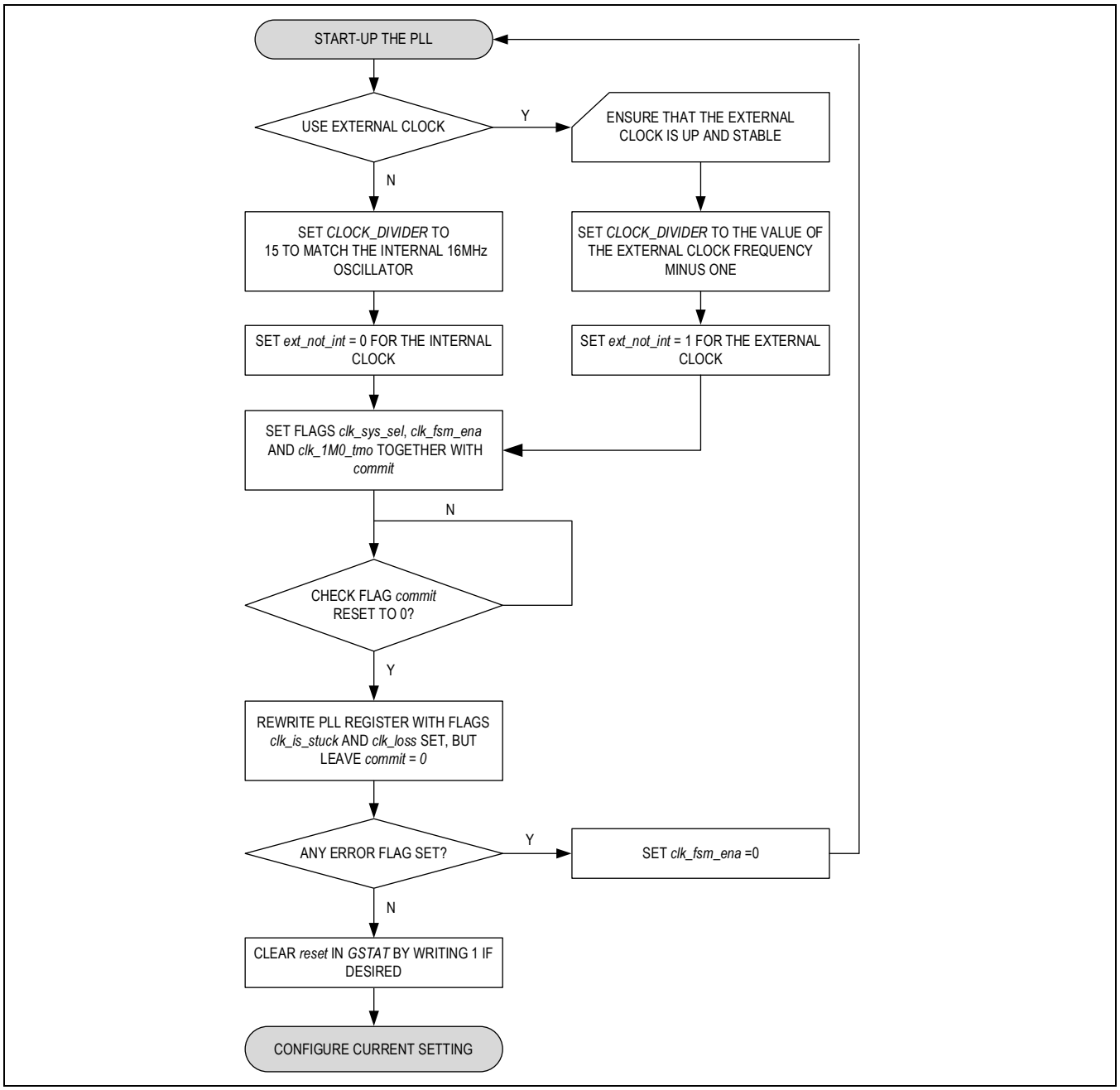


Figure 65. Quick Configuration Guide for Starting the PLL

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Current Setting Guide

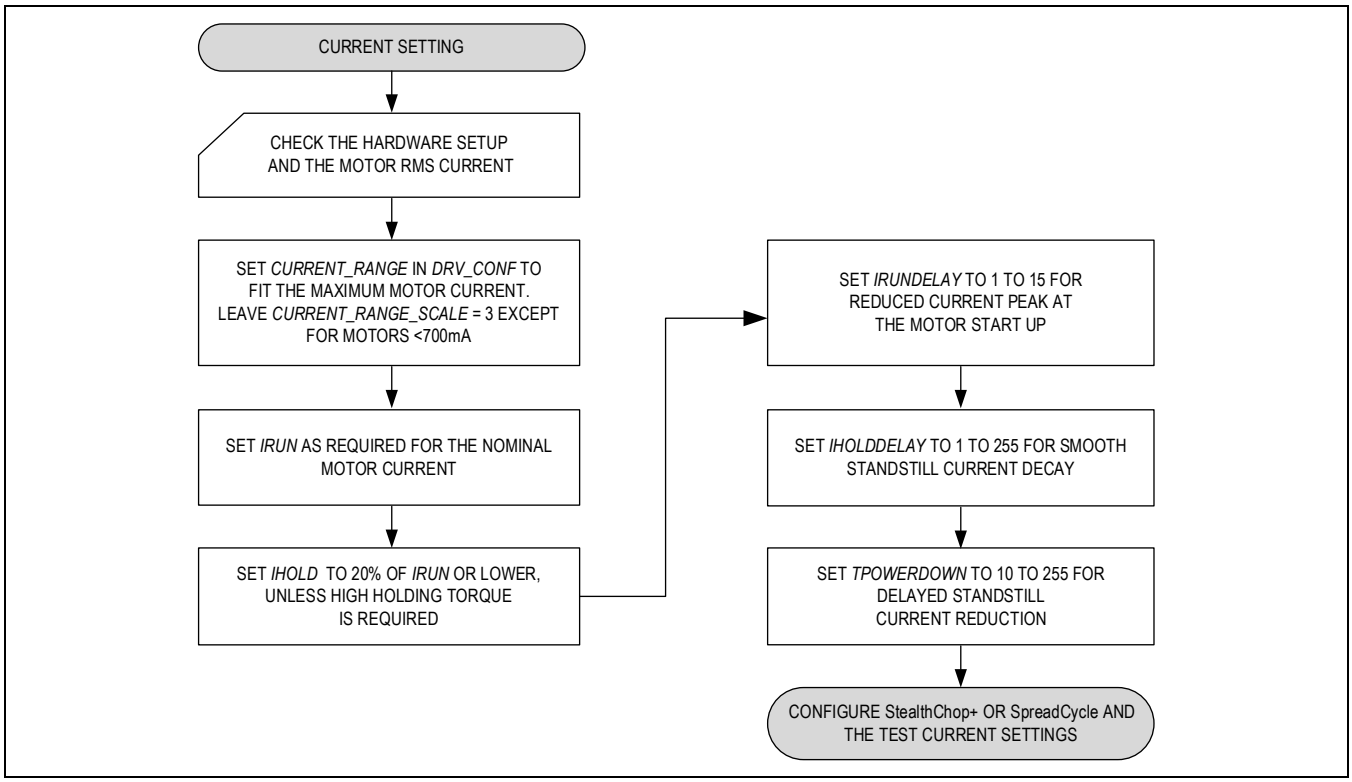


Figure 66. Quick Configuration Guide for Current Setting

Downloaded from Arrow.com.

StealthChop+ Configuration

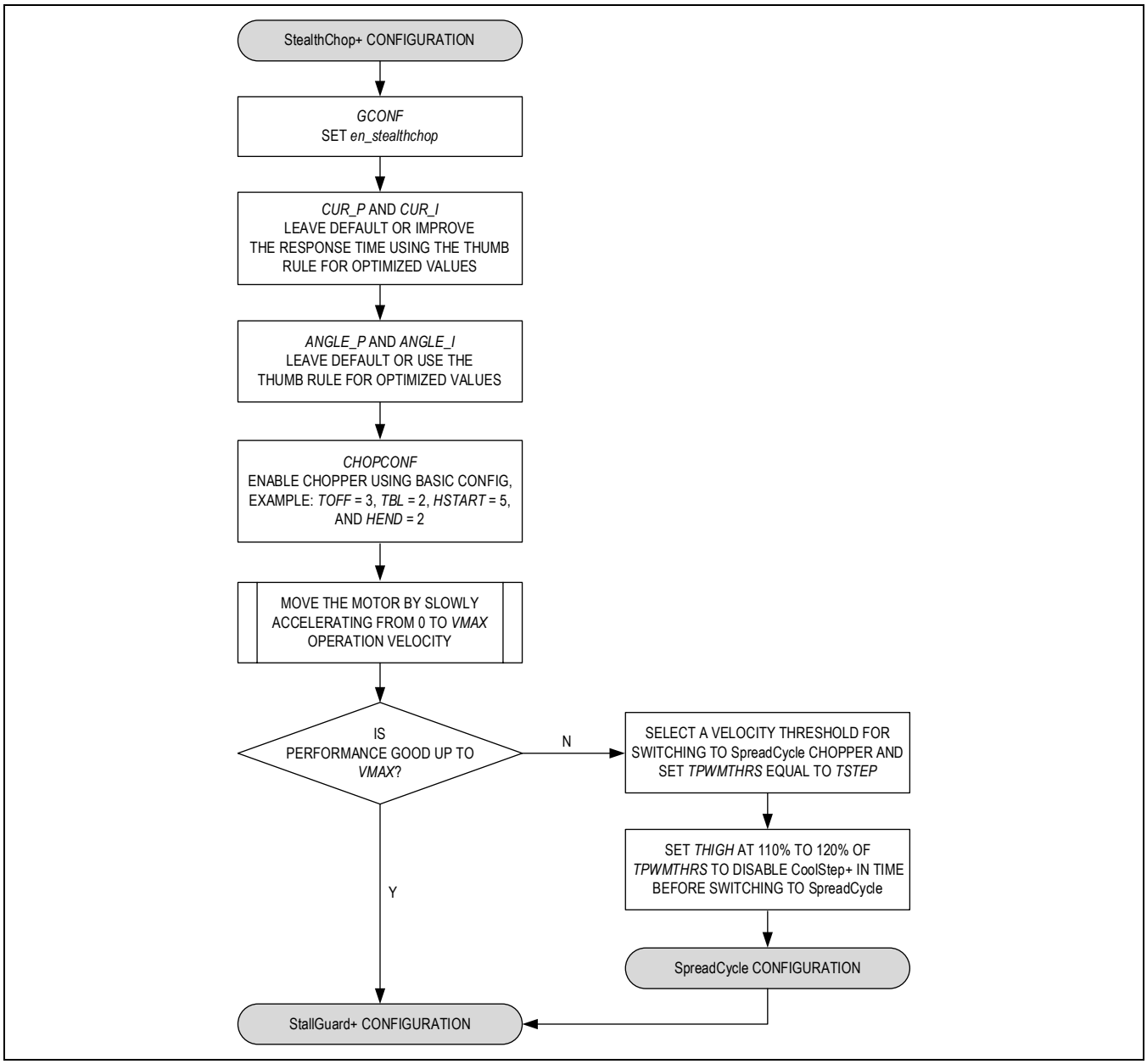


Figure 67. Quick Configuration Guide for StealthChop+ Operation

Downloaded from Arrow.com.

StallGuard+ in Combination with StealthChop+

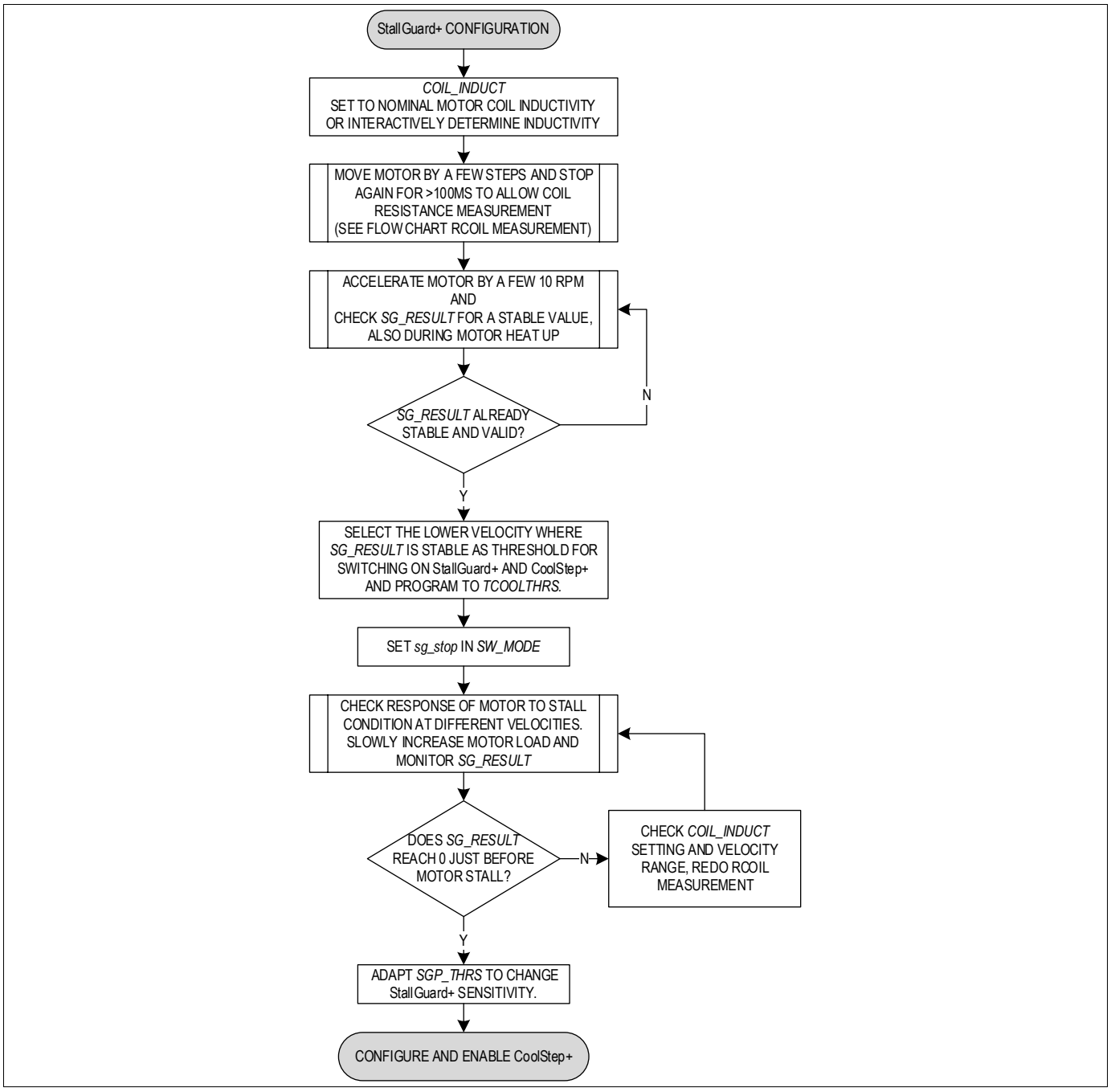


Figure 68. Quick Configuration Guide for Basic StallGuard+ Configuration (Following Configuration of StealthChop+)

Downloaded from Arrow.com.

StallGuard+ in Combination with StealthChop+

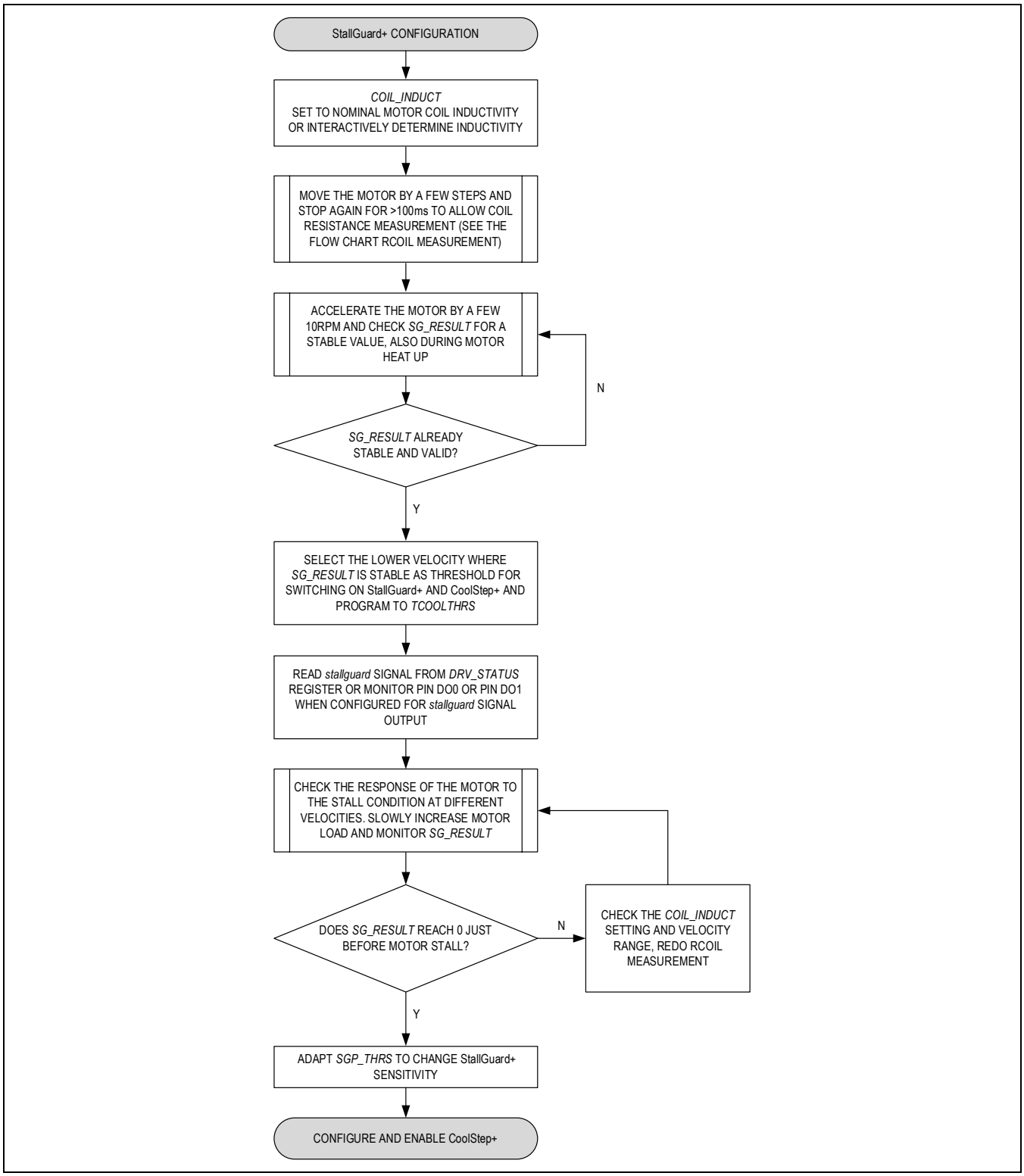


Figure 69. Quick Configuration Guide for Basic StallGuard+ Configuration (Following Configuration of StealthChop+)

Downloaded from Arrow.com.

CoolStep+ in Combination with StealthChop+

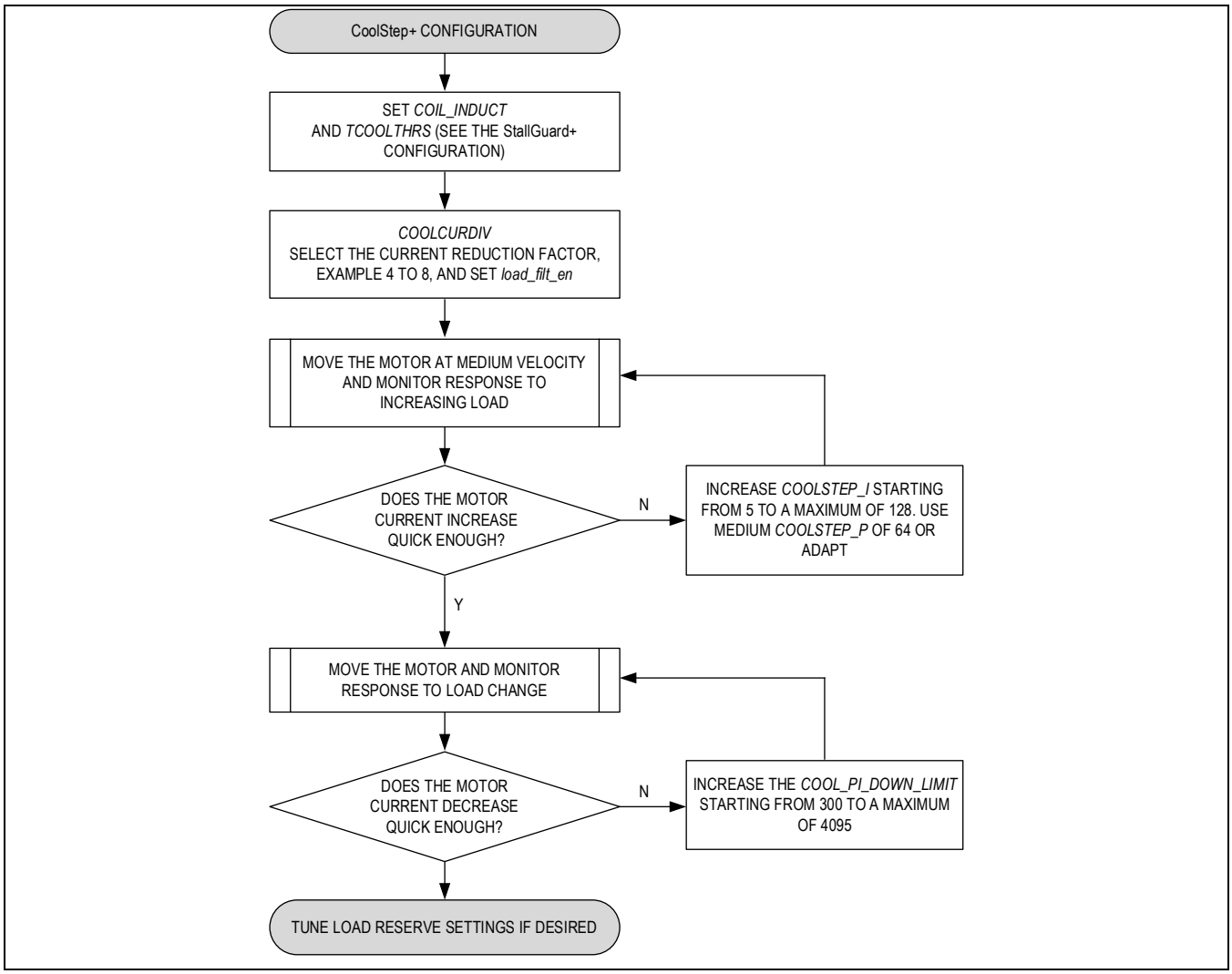


Figure 70. Quick Configuration Guide for CoolStep+ (Following Configuration of StealthChop+)

Downloaded from Arrow.com.

SpreadCycle Configuration

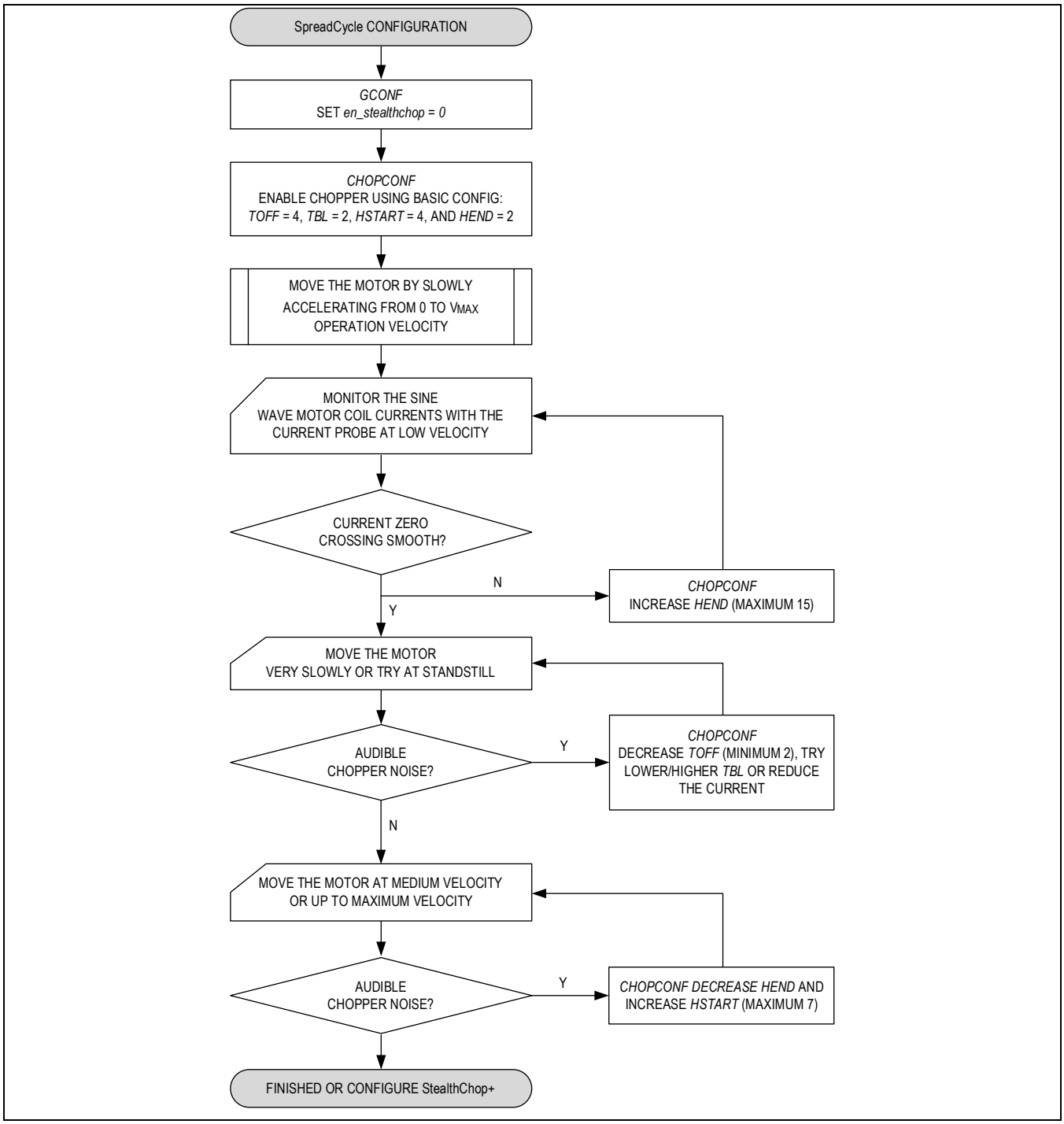


Figure 71. Quick Configuration Guide for SpreadCycle

Downloaded from Arrow.com.

CoolStep in Combination with SpreadCycle

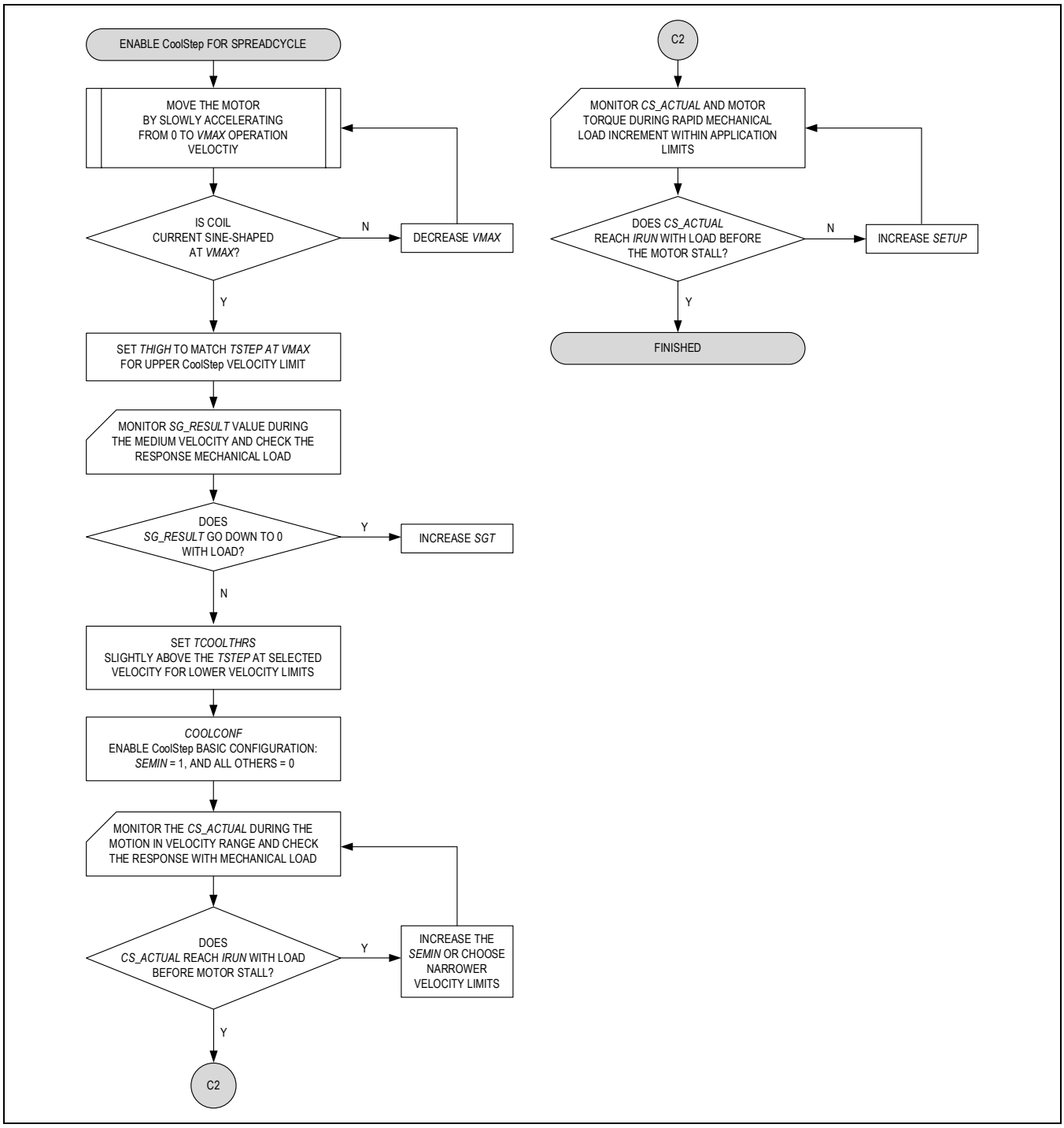


Figure 72. Quick Configuration Guide for CoolStep with SpreadCycle

Downloaded from Arrow.com.

Moving the Motor Using the Motion Controller

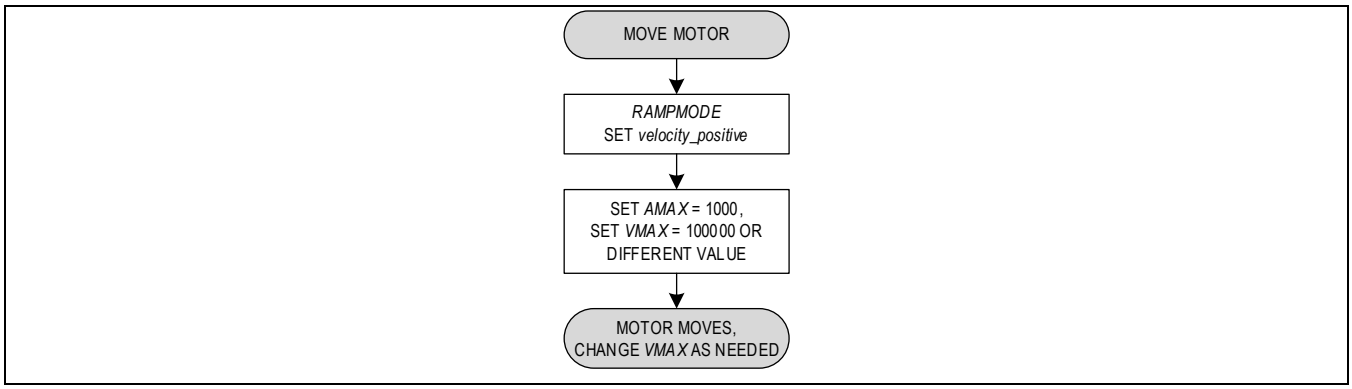


Figure 73. Quick Configuration Guide for Moving a Motor in Velocity Mode

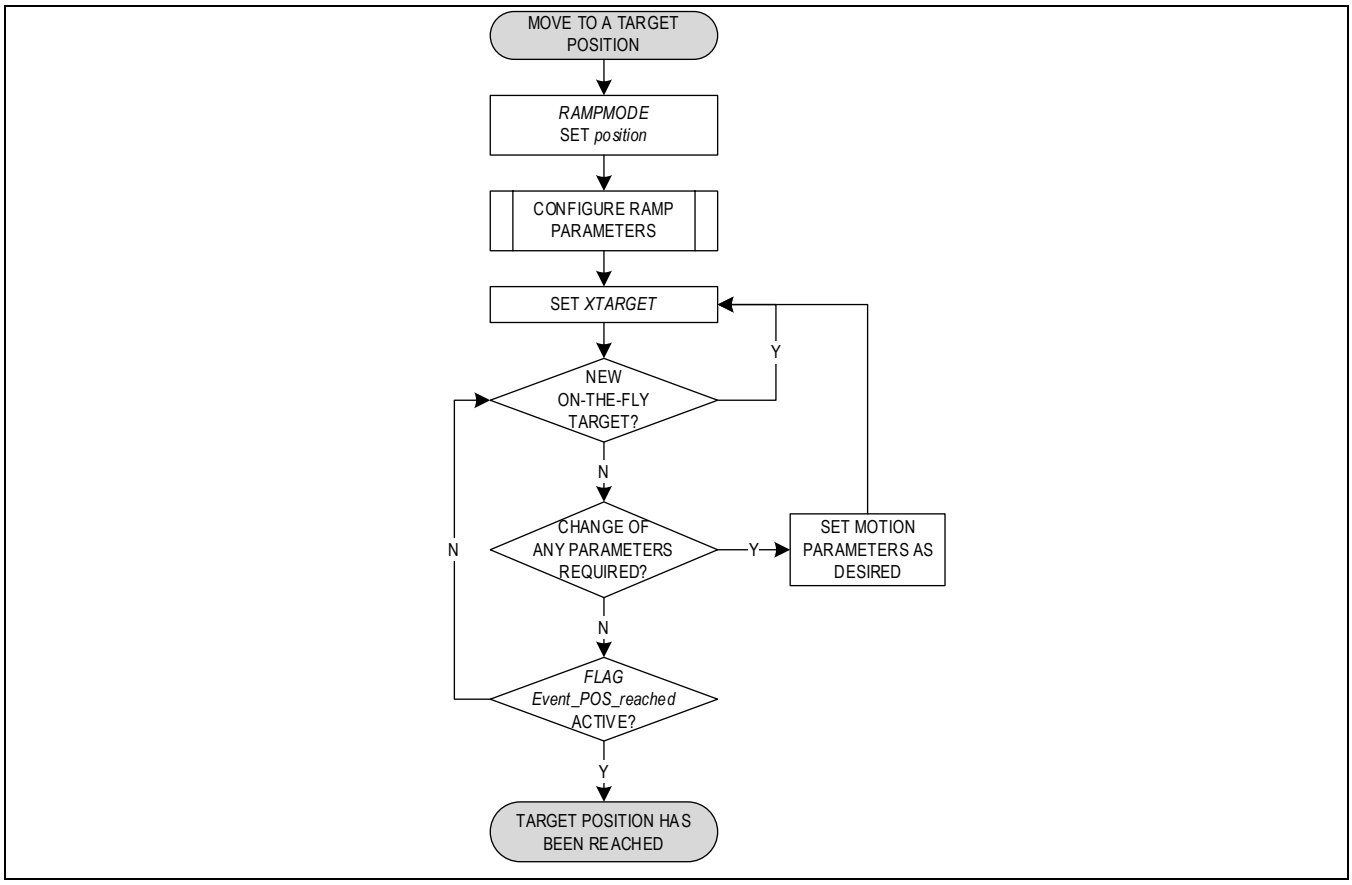


Figure 74. Quick Configuration Guide for Moving a Motor to a Target Position

Downloaded from Arrow.com.

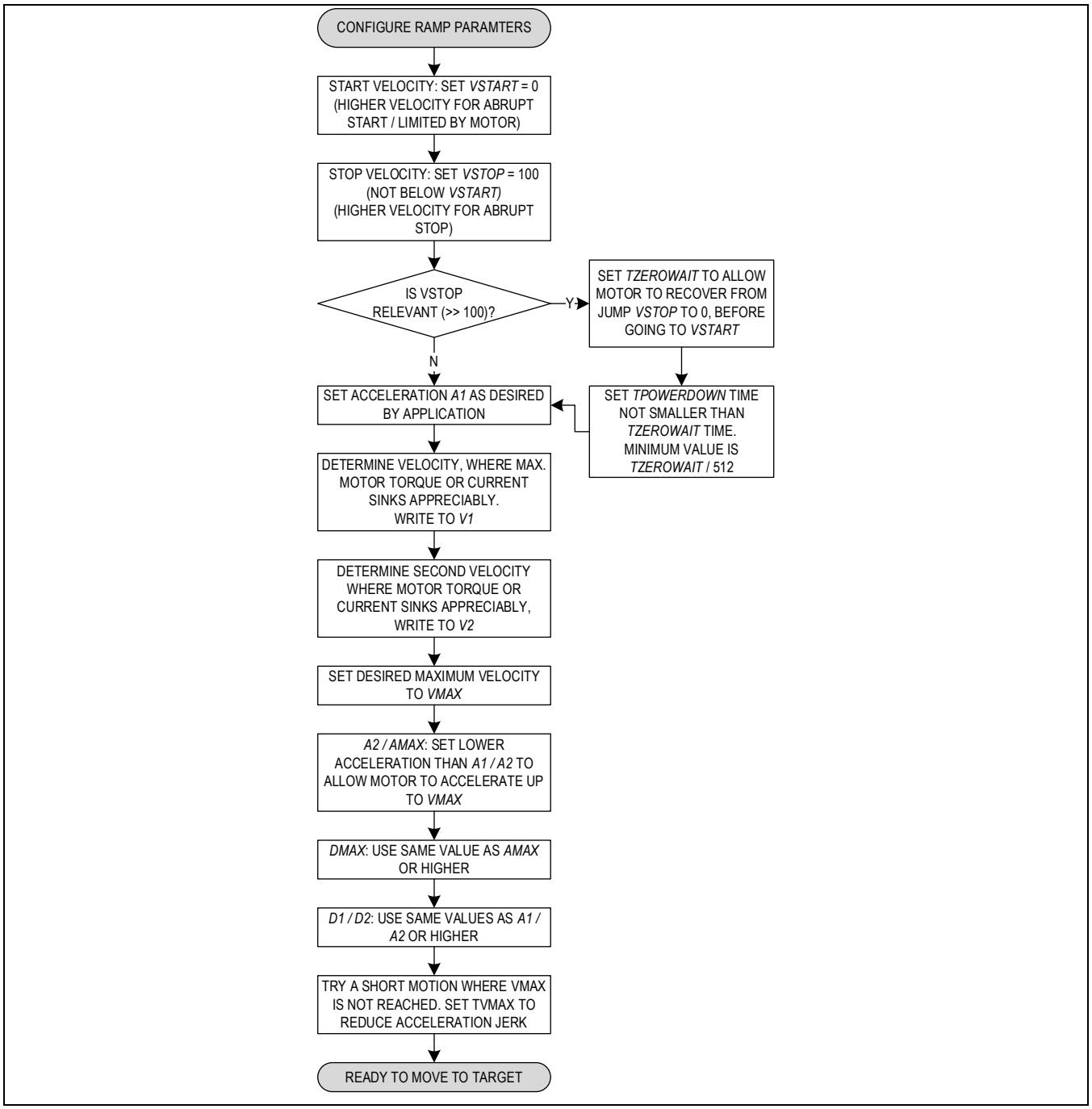


Figure 75. Quick Configuration Guide for Motion Ramp Parameter Settings

Downloaded from Arrow.com.

Initialization and Configuration Example

The following example shows a basic initialization example of the TMC5262 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 TMC5262.

```
//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, 0x00410255);    //MRES=0,TPDFD=4,TBL=3,CHM=0, DISFDCC=0,FD3=0
                                //,HEND_OFFSET=4,HSTRT_TFD210=5, TOFF=5
Write(DRV_CONF, 0x0000003f);    //full current range (4A),CURRENT_RANGE=3,
                                //CURRENT_RANGE_SCALE=3,SLOPE_CONTROL=3
Write(IHOLD_IRUN, 0x00077f01); //IRUN=127(=50%),IHOLD=1,IHOLDDELAY=7


//Ramp generator setup
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 lower threshold velocity for switching on
                                //CoolStep(+) and StallGuard(+)
```

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.

Downloaded from Arrow.com.

Register Map

TMC5262

ADDRESS	NAME	MSB							LSB
General Configuration Registers									
0x00	GCONF[31:24]	step_dir	–	–	–	–	–	–	–
	GCONF[23:16]	–	–	–	–	–	–	–	–
	GCONF[15:8]	–	–	–	OV_nN	length_steppulse[3:0]			
	GCONF[7:0]	–	direct_mode	stop_enable	small_hysteresis	shaft	multistep_filt	en_stealthchop	fast_standstill
0x01	GSTAT[31:24]	–	–	–	–	–	–	–	–
	GSTAT[23:16]	–	–	–	–	–	–	–	–
	GSTAT[15:8]	–	–	–	–	–	–	–	–
	GSTAT[7:0]	–	–	vccio_uv	vm_uvlo	register_reset	uv_cp	drv_err	reset
0x02	DO_CONF[31:24]	do1_invPP	do1_nOD_PP	do0_invPP	do0_nOD_PP	–	–	do1_ev_n_deviation	do1_ev_pos_reached
	DO_CONF[23:16]	do1_ev_stop_sg	do1_ev_stop_ref	do1_udcstep	do1_ov	do1_xcomp	do1_dir	do1_step	do1_index
	DO_CONF[15:8]	do1_stall	do1_otpw	do1_error	do0_ev_n_deviation	do0_ev_pos_reached	do0_ev_stop_sg	do0_ev_stop_ref	do0_udcstep
	DO_CONF[7:0]	do0_ov	do0_xcomp	do0_dir	do0_step	do0_index	do0_stall	do0_otpw	do0_error
0x03	DO_SCOPE_CONF[31:24]	–	–	–	–	–	–	–	–
	DO_SCOPE_CONF[23:16]	–	–	–	do1_scope_sel[4:0]				
	DO_SCOPE_CONF[15:8]	–	–	–	do1_scope_en	–	–	–	do0_scope_sel[4]
	DO_SCOPE_CONF[7:0]	do0_scope_sel[3:0]				–	–	–	do0_scope_en
0x04	IOIN[31:24]	–	–	–	–	–	–	–	–
	IOIN[23:16]	–	–	–	–	–	SILICON_RV[2:0]		

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	IOIN[15:8]	–	ext_clk	ext_res_det	–	–	–	–	–
	IOIN[7:0]	–	–	ENCN	DRV_ENN	ENCA	ENCB	REFR	REFL
0x05	X_COMPARE[31:24]	X_COMPARE[31:24]							
	X_COMPARE[23:16]	X_COMPARE[23:16]							
	X_COMPARE[15:8]	X_COMPARE[15:8]							
	X_COMPARE[7:0]	X_COMPARE[7:0]							
0x06	X_COMPARE_REPEAT[31:24]	–	–	–	–	–	–	–	–
	X_COMPARE_REPEAT[23:16]	X_COMPARE_REPEAT[23:16]							
	X_COMPARE_REPEAT[15:8]	X_COMPARE_REPEAT[15:8]							
	X_COMPARE_REPEAT[7:0]	X_COMPARE_REPEAT[7:0]							
0x0A	DRV_CONF[31:24]	–	–	–	–	–	–	–	–
	DRV_CONF[23:16]	–	–	–	–	–	–	–	–
	DRV_CONF[15:8]	–	–	–	–	–	–	–	–
	DRV_CONF[7:0]	–	–	SLOPE_CONTROL[1:0]		CURRENT_RANGE_SCALE[1:0]		CURRENT_RANGE[1:0]	
0x0B	PLL[31:24]	–	–	–	–	–	–	–	–
	PLL[23:16]	–	–	–	–	–	–	–	–
	PLL[15:8]	–	clk_is_stuck	clk_loss	clk_1m0_tmo	–	–	CLOCK_DIVIDER[4:3]	
	PLL[7:0]	CLOCK_DIVIDER[2:0]			–	clk_fsm_ena	clk_sys_sel	ext_not_int	commit
Velocity-Dependent Configuration Registers									
0x10	IHOLD_IRUN[31:24]	–	–	–	–	IRUNDELAY[3:0]			
	IHOLD_IRUN[23:16]	IHOLDDELAY[7:0]							

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	IHOLD_IRUN[15:8]	IRUN[7:0]							
	IHOLD_IRUN[7:0]	IHOLD[7:0]							
0x11	TPOWERDOWN[31:24]	–	–	–	–	–	–	–	–
	TPOWERDOWN[23:16]	–	–	–	–	–	–	–	–
	TPOWERDOWN[15:8]	–	–	–	–	–	–	–	–
	TPOWERDOWN[7:0]	TPOWERDOWN[7:0]							
0x12	TSTEP[31:24]	–	–	–	–	–	–	–	–
	TSTEP[23:16]	–	–	–	–	TSTEP[19:16]			
	TSTEP[15:8]	TSTEP[15:8]							
	TSTEP[7:0]	TSTEP[7:0]							
0x13	TPWMTHRS[31:24]	–	–	–	–	–	–	–	–
	TPWMTHRS[23:16]	–	–	–	–	TPWMTHRS[19:16]			
	TPWMTHRS[15:8]	TPWMTHRS[15:8]							
	TPWMTHRS[7:0]	TPWMTHRS[7:0]							
0x14	TCOOLTHRS[31:24]	–	–	–	–	–	–	–	–
	TCOOLTHRS[23:16]	–	–	–	–	TCOOLTHRS[19:16]			
	TCOOLTHRS[15:8]	TCOOLTHRS[15:8]							
	TCOOLTHRS[7:0]	TCOOLTHRS[7:0]							
0x15	THIGH[31:24]	–	–	–	–	–	–	–	–
	THIGH[23:16]	–	–	–	–	THIGH[19:16]			
	THIGH[15:8]	THIGH[15:8]							
	THIGH[7:0]	THIGH[7:0]							
0x16	TSGP_LOW_VEL_THRS[31:24]	–	–	–	–	–	–	–	–

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	TSGP_LOW_VEL_THRS[23:16]	–	–	–	–	TSGP_LOW_VEL_THRS[19:16]			
	TSGP_LOW_VEL_THRS[15:8]	TSGP_LOW_VEL_THRS[15:8]							
	TSGP_LOW_VEL_THRS[7:0]	TSGP_LOW_VEL_THRS[7:0]							
0x17	T_RCOIL_MEAS[31:24]	–	–	–	–	–	–	–	–
	T_RCOIL_MEAS[23:16]	–	–	–	–	T_RCOIL_MEAS[19:16]			
	T_RCOIL_MEAS[15:8]	T_RCOIL_MEAS[15:8]							
	T_RCOIL_MEAS[7:0]	T_RCOIL_MEAS[7:0]							
0x18	TUDCSTEP[31:24]	–	–	–	–	–	–	–	–
	TUDCSTEP[23:16]	–	–	–	–	TUDCSTEP[19:16]			
	TUDCSTEP[15:8]	TUDCSTEP[15:8]							
	TUDCSTEP[7:0]	TUDCSTEP[7:0]							
0x19	UDC_CONF[31:24]	–	–	–	–	–	–	–	–
	UDC_CONF[23:16]	–	–	–	–	–	–	–	–
	UDC_CONF[15:8]	–	–	–	–	–	–	–	udc_enable
	UDC_CONF[7:0]	ACCEL_THRS[3:0]				DECEL_THRS[3:0]			
0x1A	STEPS_LOST[31:24]	–	–	–	–	–	–	–	–
	STEPS_LOST[23:16]	–	–	–	–	STEPS_LOST[19:16]			
	STEPS_LOST[15:8]	STEPS_LOST[15:8]							
	STEPS_LOST[7:0]	STEPS_LOST[7:0]							
Ramp Generator Registers									
0x20	RAMPMODE[31:24]	–	–	–	–	–	–	–	–
	RAMPMODE[23:16]	–	–	–	–	–	–	–	–
	RAMPMODE[15:8]	–	–	–	–	–	–	–	–

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	RAMPMODE[7:0]	–	–	–	–	–	–	RAMPMODE[1:0]	
0x21	XACTUAL[31:24]	XACTUAL[31:24]							
	XACTUAL[23:16]	XACTUAL[23:16]							
	XACTUAL[15:8]	XACTUAL[15:8]							
	XACTUAL[7:0]	XACTUAL[7:0]							
0x22	VACTUAL[31:24]	–	–	–	–	–	–	–	–
	VACTUAL[23:16]	VACTUAL[23:16]							
	VACTUAL[15:8]	VACTUAL[15:8]							
	VACTUAL[7:0]	VACTUAL[7:0]							
0x23	VSTART[31:24]	–	–	–	–	–	–	–	–
	VSTART[23:16]	–	–	–	–	–	–	VSTART[17:16]	
	VSTART[15:8]	VSTART[15:8]							
	VSTART[7:0]	VSTART[7:0]							
0x24	A1[31:24]	–	–	–	–	–	–	–	–
	A1[23:16]	–	–	–	–	–	–	A1[17:16]	
	A1[15:8]	A1[15:8]							
	A1[7:0]	A1[7:0]							
0x25	V1[31:24]	–	–	–	–	–	–	–	–
	V1[23:16]	–	–	–	–	V1[19:16]			
	V1[15:8]	V1[15:8]							
	V1[7:0]	V1[7:0]							
0x26	AMAX[31:24]	–	–	–	–	–	–	–	–
	AMAX[23:16]	–	–	–	–	–	–	AMAX[17:16]	
	AMAX[15:8]	AMAX[15:8]							
	AMAX[7:0]	AMAX[7:0]							

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
0x27	VMAX[31:24]	–	–	–	–	–	–	–	–
	VMAX[23:16]	–	VMAX[22:16]						
	VMAX[15:8]	VMAX[15:8]							
	VMAX[7:0]	VMAX[7:0]							
0x28	DMAX[31:24]	–	–	–	–	–	–	–	–
	DMAX[23:16]	–	–	–	–	–	–	DMAX[17:16]	
	DMAX[15:8]	DMAX[15:8]							
	DMAX[7:0]	DMAX[7:0]							
0x29	TVMAX[31:24]	–	–	–	–	–	–	–	–
	TVMAX[23:16]	–	–	–	–	–	–	–	–
	TVMAX[15:8]	TVMAX[15:8]							
	TVMAX[7:0]	TVMAX[7:0]							
0x2A	D1[31:24]	–	–	–	–	–	–	–	–
	D1[23:16]	–	–	–	–	–	–	D1[17:16]	
	D1[15:8]	D1[15:8]							
	D1[7:0]	D1[7:0]							
0x2B	VSTOP[31:24]	–	–	–	–	–	–	–	–
	VSTOP[23:16]	–	–	–	–	–	–	VSTOP[17:16]	
	VSTOP[15:8]	VSTOP[15:8]							
	VSTOP[7:0]	VSTOP[7:0]							
0x2C	TZEROWAIT[31:24]	–	–	–	–	–	–	–	–
	TZEROWAIT[23:16]	–	–	–	–	–	–	–	–
	TZEROWAIT[15:8]	TZEROWAIT[15:8]							
	TZEROWAIT[7:0]	TZEROWAIT[7:0]							
0x2D	XTARGET[31:24]	XTARGET[31:24]							

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	XTARGET[23:16]	XTARGET[23:16]							
	XTARGET[15:8]	XTARGET[15:8]							
	XTARGET[7:0]	XTARGET[7:0]							
0x2E	V2[31:24]	–	–	–	–	–	–	–	–
	V2[23:16]	–	–	–	–	V2[19:16]			
	V2[15:8]	V2[15:8]							
	V2[7:0]	V2[7:0]							
0x2F	A2[31:24]	–	–	–	–	–	–	–	–
	A2[23:16]	–	–	–	–	–	–	A2[17:16]	
	A2[15:8]	A2[15:8]							
	A2[7:0]	A2[7:0]							
0x30	D2[31:24]	–	–	–	–	–	–	–	–
	D2[23:16]	–	–	–	–	–	–	D2[17:16]	
	D2[15:8]	D2[15:8]							
	D2[7:0]	D2[7:0]							
0x31	AACTUAL[31:24]	–	–	–	–	–	–	–	–
	AACTUAL[23:16]	AACTUAL[23:16]							
	AACTUAL[15:8]	AACTUAL[15:8]							
	AACTUAL[7:0]	AACTUAL[7:0]							
Ramp Generator Driver Feature Control Registers									
0x34	SW_MODE[31:24]	–	–	–	–	–	–	–	–
	SW_MODE[23:16]	–	–	–	–	–	–	–	–
	SW_MODE[15:8]	hard_stop_clr_cur_int	virtual_stop_enc	en_virtual_stop_r	en_virtual_stop_l	en_softstop	sg_stop	en_latch_encoder	latch_r_inactive
	SW_MODE[7:0]	latch_r_active	latch_l_inactive	latch_l_active	swap_lr	pol_stop_r	pol_stop_l	stop_r_enable	stop_l_enable

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
0x35	RAMP_STAT[31:24]	–	–	–	–	–	–	–	–
	RAMP_STAT[23:16]	–	–	–	–	–	–	–	–
	RAMP_STAT[15:8]	status_virtual_stop_r	status_virtual_stop_l	status_sg	second_move	t_zerowait_active	vzero	position_reached	velocity_reached
	RAMP_STAT[7:0]	event_pos_reached	event_stop_sg	event_stop_r	event_stop_l	status_latch_r	status_latch_l	status_stop_r	status_stop_l
0x36	XLATCH[31:24]	XLATCH[31:24]							
	XLATCH[23:16]	XLATCH[23:16]							
	XLATCH[15:8]	XLATCH[15:8]							
	XLATCH[7:0]	XLATCH[7:0]							
Encoder Registers									
0x38	ENCMODE[31:24]	–	–	BEMF_FILTER_SEL[1:0]		–	–	–	–
	ENCMODE[23:16]	BEMF_BLANK_TIME[7:0]							
	ENCMODE[15:8]	–	bemf_hyst[2:0]			nBEMF_ABN_SEL	enc_sel_decimal	latch_x_act	clr_enc_x
	ENCMODE[7:0]	pos_neg_edge[1:0]		clr_once	clr_cont	ignore_AB	pol_N	pol_B	pol_A
0x39	X_ENC[31:24]	X_ENC[31:24]							
	X_ENC[23:16]	X_ENC[23:16]							
	X_ENC[15:8]	X_ENC[15:8]							
	X_ENC[7:0]	X_ENC[7:0]							
0x3A	ENC_CONST[31:24]	ENC_CONST[31:24]							
	ENC_CONST[23:16]	ENC_CONST[23:16]							
	ENC_CONST[15:8]	ENC_CONST[15:8]							
	ENC_CONST[7:0]	ENC_CONST[7:0]							
0x3B	ENC_STATUS[31:24]	–	–	–	–	–	–	–	–
	ENC_STATUS[23:16]	–	–	–	–	–	–	–	–

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	ENC_STATUS[15:8]	–	–	–	–	–	–	–	–
	ENC_STATUS[7:0]	–	–	–	–	–	–	deviation_warn	n_event
0x3C	ENC_LATCH[31:24]	ENC_LATCH[31:24]							
	ENC_LATCH[23:16]	ENC_LATCH[23:16]							
	ENC_LATCH[15:8]	ENC_LATCH[15:8]							
	ENC_LATCH[7:0]	ENC_LATCH[7:0]							
0x3D	ENC_DEVIATION[31:24]	–	–	–	–	–	–	–	–
	ENC_DEVIATION[23:16]	–	–	–	–	ENC_DEVIATION[19:16]			
	ENC_DEVIATION[15:8]	ENC_DEVIATION[15:8]							
	ENC_DEVIATION[7:0]	ENC_DEVIATION[7:0]							
0x3E	VIRTUAL_STOP_L[31:24]	VIRTUAL_STOP_L[31:24]							
	VIRTUAL_STOP_L[23:16]	VIRTUAL_STOP_L[23:16]							
	VIRTUAL_STOP_L[15:8]	VIRTUAL_STOP_L[15:8]							
	VIRTUAL_STOP_L[7:0]	VIRTUAL_STOP_L[7:0]							
0x3F	VIRTUAL_STOP_R[31:24]	VIRTUAL_STOP_R[31:24]							
	VIRTUAL_STOP_R[23:16]	VIRTUAL_STOP_R[23:16]							
	VIRTUAL_STOP_R[15:8]	VIRTUAL_STOP_R[15:8]							
	VIRTUAL_STOP_R[7:0]	VIRTUAL_STOP_R[7:0]							
StealthChopPlus									
0x40	CURRENT_PI_REG[31:24]	–	–	–	–	–	–	CUR_I[9:8]	

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	CURRENT_PI_REG[23:16]	CUR_I[7:0]							
	CURRENT_PI_REG[15:8]	–	–	–	–	CUR_P[11:8]			
	CURRENT_PI_REG[7:0]	CUR_P[7:0]							
0x41	ANGLE_PI_REG[31:24]	–	–	–	–	–	–	ANGLE_I[9:8]	
	ANGLE_PI_REG[23:16]	ANGLE_I[7:0]							
	ANGLE_PI_REG[15:8]	–	–	–	–	ANGLE_P[11:8]			
	ANGLE_PI_REG[7:0]	ANGLE_P[7:0]							
0x42	CUR_ANGLE_LIMIT[31:24]	cur_pi_neg_clip	cur_pi_pos_clip	cur_pi_int_neg_clip	cur_pi_int_pos_clip	CUR_PI_LIMIT[11:8]			
	CUR_ANGLE_LIMIT[23:16]	CUR_PI_LIMIT[7:0]							
	CUR_ANGLE_LIMIT[15:8]	angle_pi_neg_clip	angle_pi_pos_clip	angle_pi_int_neg_clip	angle_pi_int_pos_clip	–	–	ANGLE_PI_LIMIT[9:8]	
	CUR_ANGLE_LIMIT[7:0]	ANGLE_PI_LIMIT[7:0]							
0x43	ANGLE_LOWER_LIMIT[31:24]	–	–	–	–	–	–	ANGLE_ERROR[9:8]	
	ANGLE_LOWER_LIMIT[23:16]	ANGLE_ERROR[7:0]							
	ANGLE_LOWER_LIMIT[15:8]	–	–	–	–	–	–	ANGLE_LOWER_I_LIMIT[9:8]	
	ANGLE_LOWER_LIMIT[7:0]	ANGLE_LOWER_I_LIMIT[7:0]							
0x44	CUR_ANGLE_MEAS[31:24]	–	–	–	–	–	–	ANGLE_MEAS[9:8]	
	CUR_ANGLE_MEAS[23:16]	ANGLE_MEAS[7:0]							

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	CUR_ANGLE_MEAS[15:8]	–	–	–	–	AMPL_MEAS[11:8]			
	CUR_ANGLE_MEAS[7:0]	AMPL_MEAS[7:0]							
0x45	PI_RESULTS[31:24]	–	–	–	–	–	–	ANGLE_CORR_CALC[9:8]	
	PI_RESULTS[23:16]	ANGLE_CORR_CALC[7:0]							
	PI_RESULTS[15:8]	–	–	–	PWM_CALC[12:8]				
	PI_RESULTS[7:0]	PWM_CALC[7:0]							
0x46	COIL_INDUCT[31:24]	–	–	–	–	–	–	–	–
	COIL_INDUCT[23:16]	–	–	–	–	–	–	rcoil_thermal_coupling	rcoil_manual
	COIL_INDUCT[15:8]	–	COIL_INDUCT[14:8]						
	COIL_INDUCT[7:0]	COIL_INDUCT[7:0]							
0x47	R_COIL[31:24]	–	–	–	–	R_COIL_AUTO_A[11:8]			
	R_COIL[23:16]	R_COIL_AUTO_A[7:0]							
	R_COIL[15:8]	–	–	–	–	R_COIL_AUTO_B[11:8]			
	R_COIL[7:0]	R_COIL_AUTO_B[7:0]							
0x48	R_COIL_USER[31:24]	–	–	–	–	R_COIL_USER_A[11:8]			
	R_COIL_USER[23:16]	R_COIL_USER_A[7:0]							
	R_COIL_USER[15:8]	–	–	–	–	R_COIL_USER_B[11:8]			
	R_COIL_USER[7:0]	R_COIL_USER_B[7:0]							
0x49	SGP_CONF[31:24]	–	–	SGP_LOW_VEL_CNTS[1:0]		–	–	–	–
	SGP_CONF[23:16]	SGP_LOW_VEL_SLOPE[7:0]							
	SGP_CONF[15:8]	–	sgp_clear_cur_pi	sgp_low_vel_freeze	sgp_filt_en	–	–	–	SGP_THRS[8]
	SGP_CONF[7:0]	SGP_THRS[7:0]							
0x4A	SGP_IND_2_3[31:24]	–	–	–	–	–	–	SGP_IND_3[9:8]	

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	SGP_IND_2_3[23:16]	SGP_IND_3[7:0]							
	SGP_IND_2_3[15:8]	–	–	–	–	–	–	SGP_IND_2[9:8]	
	SGP_IND_2_3[7:0]	SGP_IND_2[7:0]							
0x4B	SGP_IND_0_1[31:24]	–	–	–	–	–	–	SGP_IND_1[9:8]	
	SGP_IND_0_1[23:16]	SGP_IND_1[7:0]							
	SGP_IND_0_1[15:8]	–	–	–	–	–	–	SGP_IND_0[9:8]	
	SGP_IND_0_1[7:0]	SGP_IND_0[7:0]							
0x4C	INDUCTANCE_VOLTAGE[31:24]	–	–	–	–	UL_A[11:8]			
	INDUCTANCE_VOLTAGE[23:16]	UL_A[7:0]							
	INDUCTANCE_VOLTAGE[15:8]	–	–	–	–	UL_B[11:8]			
	INDUCTANCE_VOLTAGE[7:0]	UL_B[7:0]							
0x4D	SGP_BEMF[31:24]	–	–	–	–	UBEMF_ABS[11:8]			
	SGP_BEMF[23:16]	UBEMF_ABS[7:0]							
	SGP_BEMF[15:8]	–	–	–	–	–	–	SGP_RAW[9:8]	
	SGP_BEMF[7:0]	SGP_RAW[7:0]							
0x4E	COOLSTEPPLUS_CONF[31:24]	–	–	–	–	–	–	–	–
	COOLSTEPPLUS_CONF[23:16]	–	–	–	–	–	–	–	–
	COOLSTEPPLUS_CONF[15:8]	–	–	–	–	–	–	–	–
	COOLSTEPPLUS_CONF[7:0]	–	–	–	load_filt_en	COOL_CUR_DIV[3:0]			
0x4F	COOLSTEPPLUS_PI_REG[31:24]	–	–	–	–	–	–	COOLSTEP_I[9:8]	

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	COOLSTEPPLUS_PI_REG[23:16]	COOLSTEP_I[7:0]							
	COOLSTEPPLUS_PI_REG[15:8]	–	–	–	–	COOLSTEP_P[11:8]			
	COOLSTEPPLUS_PI_REG[7:0]	COOLSTEP_P[7:0]							
0x50	COOLSTEPPLUS_PI_DOWN[31:24]	–	–	–	–	COOL_PI_OFF_SPEED[11:8]			
	COOLSTEPPLUS_PI_DOWN[23:16]	COOL_PI_OFF_SPEED[7:0]							
	COOLSTEPPLUS_PI_DOWN[15:8]	–	–	–	–	COOL_PI_DOWN_LIMIT[11:8]			
	COOLSTEPPLUS_PI_DOWN[7:0]	COOL_PI_DOWN_LIMIT[7:0]							
0x51	COOLSTEPPLUS_RESERVE_CONF[31:24]	COOL_HI_GENERATORIC_RESERVE[7:0]							
	COOLSTEPPLUS_RESERVE_CONF[23:16]	COOL_LOW_GENERATORIC_RESERVE[7:0]							
	COOLSTEPPLUS_RESERVE_CONF[15:8]	COOL_HI_LOAD_RESERVE[7:0]							
	COOLSTEPPLUS_RESERVE_CONF[7:0]	COOL_LOW_LOAD_RESERVE[7:0]							
0x52	COOLSTEPPLUS_LOAD_RESERVE[31:24]	–	–	–	–	–	–	–	COOLSTEP_LOAD_RESERVE[8]
	COOLSTEPPLUS_LOAD_RESERVE[23:16]	COOLSTEP_LOAD_RESERVE[7:0]							
	COOLSTEPPLUS_LOAD_RESERVE[15:8]	–	–	–	–	–	–	SGP_RESULT[9:8]	
	COOLSTEPPLUS_LOAD_RESERVE[7:0]	SGP_RESULT[7:0]							
0x53	TSTEP Velocity[31:24]	–	–	–	–	–	–	–	–
	TSTEP Velocity[23:16]	–	TSTEP_VELOCITY[22:16]						
	TSTEP Velocity[15:8]	TSTEP_VELOCITY[15:8]							

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	TSTEP Velocity[7:0]	TSTEP_VELOCITY[7:0]							
ADC Registers									
0x58	ADC_VSUPPLY_TEMP[31:24]	–	–	–	–	–	–	–	ADC_TEMP[8]
	ADC_VSUPPLY_TEMP[23:16]	ADC_TEMP[7:0]							
	ADC_VSUPPLY_TEMP[15:8]	–	–	–	–	–	–	–	ADC_VSUPPLY[8]
	ADC_VSUPPLY_TEMP[7:0]	ADC_VSUPPLY[7:0]							
0x59	ADC_I[31:24]	–	–	–	–	ADC_I_B[11:8]			
	ADC_I[23:16]	ADC_I_B[7:0]							
	ADC_I[15:8]	–	–	–	–	ADC_I_A[11:8]			
	ADC_I[7:0]	ADC_I_A[7:0]							
0x5A	OTW_OV_VTH[31:24]	–	–	–	–	–	–	–	OVERTEMPPREWARNING_VTH[8]
	OTW_OV_VTH[23:16]	OVERTEMPPREWARNING_VTH[7:0]							
	OTW_OV_VTH[15:8]	–	–	–	–	–	–	–	OVERVOLTAGE_VTH[8]
	OTW_OV_VTH[7:0]	OVERVOLTAGE_VTH[7:0]							
Motor Driver Registers									
0x60	MSLUT_0[31:24]	MSLUT_0[31:24]							
	MSLUT_0[23:16]	MSLUT_0[23:16]							
	MSLUT_0[15:8]	MSLUT_0[15:8]							
	MSLUT_0[7:0]	MSLUT_0[7:0]							
0x61	MSLUT_1[31:24]	MSLUT_1[31:24]							
	MSLUT_1[23:16]	MSLUT_1[23:16]							
	MSLUT_1[15:8]	MSLUT_1[15:8]							

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
	MSLUT_1[7:0]	MSLUT_1[7:0]							
0x62	MSLUT_2[31:24]	MSLUT_2[31:24]							
	MSLUT_2[23:16]	MSLUT_2[23:16]							
	MSLUT_2[15:8]	MSLUT_2[15:8]							
	MSLUT_2[7:0]	MSLUT_2[7:0]							
0x63	MSLUT_3[31:24]	MSLUT_3[31:24]							
	MSLUT_3[23:16]	MSLUT_3[23:16]							
	MSLUT_3[15:8]	MSLUT_3[15:8]							
	MSLUT_3[7:0]	MSLUT_3[7:0]							
0x64	MSLUT_4[31:24]	MSLUT_4[31:24]							
	MSLUT_4[23:16]	MSLUT_4[23:16]							
	MSLUT_4[15:8]	MSLUT_4[15:8]							
	MSLUT_4[7:0]	MSLUT_4[7:0]							
0x65	MSLUT_5[31:24]	MSLUT_5[31:24]							
	MSLUT_5[23:16]	MSLUT_5[23:16]							
	MSLUT_5[15:8]	MSLUT_5[15:8]							
	MSLUT_5[7:0]	MSLUT_5[7:0]							
0x66	MSLUT_6[31:24]	MSLUT_6[31:24]							
	MSLUT_6[23:16]	MSLUT_6[23:16]							
	MSLUT_6[15:8]	MSLUT_6[15:8]							
	MSLUT_6[7:0]	MSLUT_6[7:0]							
0x67	MSLUT_7[31:24]	MSLUT_7[31:24]							
	MSLUT_7[23:16]	MSLUT_7[23:16]							
	MSLUT_7[15:8]	MSLUT_7[15:8]							
	MSLUT_7[7:0]	MSLUT_7[7:0]							

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
0x68	MSLUTSEL[31:24]	X3[7:0]							
	MSLUTSEL[23:16]	X2[7:0]							
	MSLUTSEL[15:8]	X1[7:0]							
	MSLUTSEL[7:0]	W3[1:0]		W2[1:0]		W1[1:0]		W0[1:0]	
0x69	MSLUTSTART[31:24]	OFFSET_SIN90[7:0]							
	MSLUTSTART[23:16]	START_SIN90[7:0]							
	MSLUTSTART[15:8]	–	–	–	–	–	–	–	–
	MSLUTSTART[7:0]	START_SIN[7:0]							
0x6A	MSCNT[31:24]	–	–	–	–	–	–	–	–
	MSCNT[23:16]	–	–	–	–	–	–	–	–
	MSCNT[15:8]	–	–	–	–	–	–	MSCNT[9:8]	
	MSCNT[7:0]	MSCNT[7:0]							
0x6B	MSCURACT[31:24]	–	–	–	–	–	–	–	CUR_A[8]
	MSCURACT[23:16]	CUR_A[7:0]							
	MSCURACT[15:8]	–	–	–	–	–	–	–	CUR_B[8]
	MSCURACT[7:0]	CUR_B[7:0]							
0x6C	CHOPCONF[31:24]	–	–	dedge	intpol	MRES[3:0]			
	CHOPCONF[23:16]	TPFD[3:0]				–	–	–	TBL[1]
	CHOPCONF[15:8]	TBL[0]	chm	–	disfdcc	fd3	HEND_OFFSET[3:1]		
	CHOPCONF[7:0]	HEND_OFFSET[0]	HSTRT_TFD210[2:0]			TOFF[3:0]			
0x6D	COOLCONF[31:24]	–	–	–	–	–	–	–	sfilt
	COOLCONF[23:16]	thigh_sg_off	sgt[6:0]						
	COOLCONF[15:8]	seimin	sedn[2:0]			semax[3:0]			
	COOLCONF[7:0]	–	seup[1:0]		–	semin[3:0]			

Downloaded from Arrow.com.

ADDRESS	NAME	MSB							LSB
0x6F	DRV_STATUS[31:24]	stst	olb	ola	s2gb	s2ga	otpw	ot	stallguard
	DRV_STATUS[23:16]	CS_ACTUAL[7:0]							
	DRV_STATUS[15:8]	–	stealth	s2vsb	s2vsa	ov	seq_stopped	SG_RESULT[9:8]	
	DRV_STATUS[7:0]	SG_RESULT[7:0]							
0x70	PWMCONF[31:24]	–	–	–	–	–	–	–	–
	PWMCONF[23:16]	–	–	–	–	SD_ON_MEAS_HI[3:0]			
	PWMCONF[15:8]	SD_ON_MEAS_LO[3:0]				–	–	–	–
	PWMCONF[7:0]	OL_THRSH[1:0]		FREEWHEEL[1:0]		PWM_FREQ[3:0]			

Register Blocks

General Configuration Registers

Register Details

GCONF (0x0)

Global Configuration Flags

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

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	OV_nN	length_steppulse[3:0]			
Reset	–	–	–	0x0				
Access Type	–	–	–	Write, Read	Write, Read			
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	direct_mode	stop_enable	small_hysteresis	shaft	multistep_filt	en_stealthchop	fast_standstill
Reset	–	0b0	0b0	0b1	0b0	0b1	0b0	0b0
Access Type	–	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
step_dir	31	Enable use of STEP and DIR interface as step source (steps do not affect XACTUAL).	0x0: Disable 0x1: Enable
OV_nN	12	Enable use of ENCN as overvoltage output (high active open-drain).	0x0: Use ENCN as N-channel of encoder input (default). 0x1: Use ENCN as overvoltage output (high active open-drain). Hint: in this case, the N signal seen by the IC is always 0.
length_steppulse	11:8	Length of step pulse and dir. to step setup time when using step pulse output signal.	0x0: Double edge output signal: each step toggles the STEP output.
direct_mode	6	Enable direct motor phase current control through the serial interface.	0x0: Normal operation 0x1: Motor coil currents and polarity directly programmed through the serial interface: register XTARGET (0x2D) specifies signed coil A current (bits 8..0) and coil B current (bits 24..16). In this mode, the current is scaled by IHOLD setting. Velocity-based current regulation of StealthChop+ is not available in this mode. The automatic StealthChop+ current regulation works only for low stepper motor velocities.
stop_enable	5	Motor hard stop function enable.	0x0: Normal operation 0x1: Emergency stop: ENCA stops the sequencer when tied high (no steps are executed by the sequencer, motor goes to standstill state).
small_hysteresis	4	Hysteresis control for TSTEP comparison of velocity thresholds.	0x0: Hysteresis for step frequency comparison is 1/16 (higher tolerance to frequency jitter with external STEP and DIR source). 0x1: Hysteresis for step frequency comparison is 1/32 (recommended with hardware-based ramp generator).
shaft	3	Change motor direction invert DIR signal (STEP and DIR driver only).	0x0: Default motor direction 0x1: Inverse motor direction (allowed for STEP and DIR driver only).
multistep_filt	2	Enable step input filtering for StealthChop+.	0x0: Step input filtering disabled. 0x1: Enable step input filtering for StealthChop+ optimization with external step source (default = 1).
en_stealthchop	1	Enable the StealthChop+ mode	0x0: No StealthChop+ 0x1: StealthChop+ voltage PWM mode enabled (depending on velocity thresholds). Switch from off to on state while in standstill.
fast_standstill	0	Step pulse timeout for standstill detection.	0x0: Normal time: 2^{20} clocks (required if very low step frequencies are used, example, with low microstep

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
			resolution). 0x1: Short time: 2^{18} clocks (recommended for fast current reduction).

GSTAT (0x1)

Global Status Flags
(Rewrite with '1' bit to clear respective flags.)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	vccio_uv	vm_uvlo	register_reset	uv_cp	drv_err	reset
Reset	–	–	0b0	0b0	0b1	0b0	0b0	0b1
Access Type	–	–	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
vccio_uv	5	1: VCC_IO undervoltage occured since last reset. Clear once after initial power-up.	0x0: Normal operation 0x1: VCC_IO dropped below the lower threshold after last access to this register. Interface commands may have been lost.
vm_uvlo	4	1: VM undervoltage occured since last reset. Clear once after initial power-up.	0x0: Normal operation 0x1: VM had undervoltage, potential loss of motor position.
register_reset	3	Indicates a reset of the configuration registers	0x0: Normal operation 0x1: Indicates the register map is reset. All registers are cleared to reset values.
uv_cp	2	Charge pump undervoltage condition flag. Clear once after power-up.	0x0: Normal operation 0x1: Indicates an undervoltage on the charge pump. The driver is disabled during undervoltage. This flag is latched for information.
drv_err	1	Driver error flag	0x0: Normal operation 0x1: Indicates the driver is shut down due to overtemperature or short circuit detection. Read DRV_STATUS for details. The flag can only be cleared when the temperature is below the limit again.
reset	0	Reset flag	0x0: normal operation 0x1: Indicates the IC is reset since the last clear of this bit.

DO_CONF (0x2)

DO0 and DO1 output pin configuration.

BIT	31	30	29	28	27	26	25	24
Field	do1_invPP	do1_nOD_PP	do0_invPP	do0_nOD_PP	–	–	do1_ev_n_deviation	do1_ev_pos_reached
Reset	0x1	0b0	0x1	0b0	–	–		
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	–	–	Write, Read	Write, Read
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	do1_ev_stop_sg	do1_ev_stop_ref	do1_udcstep	do1_ov	do1_xcomp	do1_dir	do1_step	do1_index
Reset						0b0	0b0	
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**

Downloaded from Arrow.com.

Field	do1_stall	do1_otpw	do1_error	do0_ev_n_deviation	do0_ev_pos_reached	do0_ev_stop_sg	do0_ev_stop_ref	do0_udcstep
Reset	0b0	0b0	0b0					
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	do0_ov	do0_xcomp	do0_dir	do0_step	do0_index	do0_stall	do0_otpw	do0_error
Reset			0b0	0b0		0b0	0b0	0b0
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
do1_invPP	31	Select polarity for DO1 in push-pull mode.	0x0: High active output 0x1: Low active output
do1_nOD_PP	30	DO1 output type configuration.	0x0: DO1 is open collector output (active low). 0x1: Enable DO1 push-pull output.
do0_invPP	29	Select polarity for DO0 in push-pull mode.	0x0: High active output 0x1: Low active output
do0_nOD_PP	28	DO0 output type configuration.	0x0: DO0 is open collector output (active low). 0x1: Enable DO0 push-pull output.
do1_ev_n_deviation	25	Map N event and deviation warning to DO1.	0x0: Disable 0x1: Enable
do1_ev_pos_reached	24	Map RAMPSTAT_EVENT_POS_REACHED to DO1.	0x0: Disable 0x1: Enable
do1_ev_stop_sg	23	Map RAMPSTAT_EVENT_SG_STOP to DO1.	0x0: Disable 0x1: Enable
do1_ev_stop_ref	22	Map RAMPSTAT_EVENT_STOPL || RAMPSTAT_EVENT_STOPR to DO1.	0x0: Disable 0x1: Enable
do1_udcstep	21	Map deceleration activity of µDcStep to DO1.	0x0: Disable 0x1: Enable
do1_ov	20	Map detection of overvoltage to DO1.	0x0: Disable 0x1: Enable
do1_xcomp	19	Map position comparator to DO1.	0x0: Disable 0x1: Enable
do1_dir	18	Map DIR output signal to DO1.	0x0: Disable 0x1: Enable
do1_step	17	Map STEP output signal to DO1.	0x0: Disable 0x1: Enable
do1_index	16	Map index signal to DO1.	0x0: Disable 0x1: Enable
do1_stall	15	Map StallGuard signal stallguard to DO1.	0x0: Disable 0x1: Enable

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
do1_otpw	14	Map OTPW (overtemperature prewarning) to DO1.	0x0: Disable 0x1: Enable
do1_error	13	Map driver error condition to DO1.	0x0: Disable 0x1: Enable
do0_ev_n_deviation	12	Map N event and deviation warning to DO0.	0x0: Disable 0x1: Enable
do0_ev_pos_reached	11	Map RAMPSTAT_EVENT_POS_REACHED to DO0.	0x0: Disable 0x1: Enable
do0_ev_stop_sg	10	Map RAMPSTAT_EVENT_SG_STOP to DO0.	0x0: Disable 0x1: Enable
do0_ev_stop_ref	9	Map RAMPSTAT_EVENT_STOPL || RAMPSTAT_EVENT_STOPR to DO0.	0x0: Disable 0x1: Enable
do0_udcstep	8	Map deceleration activity of µDcStep to DO0.	0x0: Disable 0x1: Enable
do0_ov	7	Map detection of overvoltage to DO0.	0x0: Disable 0x1: Enable
do0_xcomp	6	Map position comparator to DO0.	0x0: Disable 0x1: Enable
do0_dir	5	Map DIR output signal to DO0.	0x0: Disable 0x1: Enable
do0_step	4	Map STEP output signal to DO0.	0x0: Disable 0x1: Enable
do0_index	3	Map index signal to DO0.	0x0: Disable 0x1: Enable
do0_stall	2	Map StallGuard signal stallguard to DO0.	0x0: Disable 0x1: Enable
do0_otpw	1	Map OTPW (overtemperature prewarning) to DO0.	0x0: Disable 0x1: Enable
do0_error	0	Map driver error condition to DO0. DO0 always shows the reset status. It is active low during reset condition.	0x0: Disable 0x1: Enable

DO_SCOPE_CONF (0x3)

DO0 and DO1 output pin RT-OCSI channel configuration

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

Downloaded from Arrow.com.

BIT	23	22	21	20	19	18	17	16
Field	–	–	–	do1_scope_sel[4:0]				
Reset	–	–	–					
Access Type	–	–	–	Write, Read				
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	do1_scope_en	–	–	–	do0_scope_sel[4]
Reset	–	–	–		–	–	–	
Access Type	–	–	–	Write, Read	–	–	–	Write, Read
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	do0_scope_sel[3:0]				–	–	–	do0_scope_en
Reset					–	–	–	
Access Type	Write, Read				–	–	–	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
do1_scope_sel	20:16	Selection of DO1 RT-OSCI source	0x0: ADC_IA 0x1: ADC_IB 0x2: RCOIL_A 0x3: RCOIL_B 0x4: UL_A 0x5: UL_B 0x6: COOLSTEP_LOAD_RESERVE 0x7: ADC_TEMPERATURE 0x8: ANGLE_MEAS 0x9: SGP_RAW 0xA: ANGLE_CORR_CALC 0xB: AMPL_MEAS 0xC: PWM_CALC 0xD: CS_ACTUAL 0xE: UBEMF_ABS 0xF: SG_RESULT (sg2/sgp) 0x10: SGP_RESULT 0x11: CUR_A 0x12: CUR_B 0x13: DAC_X 0x14: DAC_Y 0x15: ANGLE_ERROR 0x16: MSCNT 0x17: MSCNT_SNPSHT 0x18: MSCNT_OFFSET 0x19: TSTEP_VELOCITY[20:13] 0x1A: TSTEP_VELOCITY[18:11] 0x1B: TSTEP_VELOCITY[16:9] 0x1C: USER_VALUE (=DO_CONF[20:13])
do1_scope_en	12	Enable RT-OSCI on DO1 pin	0x0: RT-OSCI disabled 0x1: RT-OSCI enabled

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
do0_scope_sel	8:4	Selection of D0 RT-OSCI source	0x0: ADC_IA 0x1: ADC_IB 0x2: RCOIL_A 0x3: RCOIL_B 0x4: UL_A 0x5: UL_B 0x6: COOLSTEP_LOAD_RESERVE 0x7: ADC_TEMPERATURE 0x8: ANGLE_MEAS 0x9: SGP_RAW 0xA: ANGLE_CORR_CALC 0xB: AMPL_MEAS 0xC: PWM_CALC 0xD: CS_ACTUAL 0xE: UBEMF_ABS 0xF: SG_RESULT (sg2/sgp) 0x10: SGP_RESULT 0x11: CUR_A 0x12: CUR_B 0x13: DAC_X 0x14: DAC_Y 0x15: ANGLE_ERROR 0x16: MSCNT 0x17: MSCNT_SNPSHT 0x18: MSCNT_OFFSET 0x19: TSTEP_VELOCITY[20:13] 0x1A: TSTEP_VELOCITY[18:11] 0x1B: TSTEP_VELOCITY[16:9] 0x1C: USER_VALUE (=DO_CONF[7:0])
do0_scope_en	0	Enable RT-OSCI on DO0 pin	0x0: RT-OSCI disabled 0x1: RT-OSCI enabled

IOIN (0x4)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	SILICON_RV[2:0]		
Reset	–	–	–	–	–	0b000		
Access Type	–	–	–	–	–	Read Only		
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	ext_clk	ext_res_det	–	–	–	–	–

Downloaded from Arrow.com.

Reset	–	0b0	0b0	–	–	–	–	–
Access Type	–	Read Only	Read Only	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	ENCN	DRV_ENN	ENCA	ENCB	REFR	REFL
Reset	–	–	0b0	0b0	0b0	0b0	0b0	0b0
Access Type	–	–	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION	DECODE
SILICON_RV	18:16	Silicon revision number	
ext_clk	14	Detection of valid external clock CLK.	0x0: The internal oscillator is used for generating the clock-signal (16 MHz). 0x1: An external oscillator is used for generating the clock-signal.
ext_res_det	13	Check of external reference resistor.	0x0: Resistor is missing, driver cannot operate. 0x1: Normal operation: External resistor between REF and GND.
ENCN	5	State of pin ENCN.	0x0: ENCN is logic level low. 0x1: ENCN is logic level high.
DRV_ENN	4	Driver disabled/enabled state.	0x0: Enable 0x1: Disable
ENCA	3	State of pin ENCA.	0x0: ENCA is logic level low. 0x1: ENCA is logic level high.
ENCB	2	State of pin ENCB.	0x0: ENCB is logic level low. 0x1: ENCB is logic level high.
REFR	1	REFR pin state (DIR pin in STEP&DIR).	0x0: REFR pin is low logic level. 0x1: REFR pin is high logic level.
REFL	0	REFL pin state (STEP pin in STEP&DIR).	0x0: REFL pin is low logic level. 0x1: REFL pin is high logic level.

X_COMPARE (0x5)

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

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION
X_COMPARE	31:0	**Position comparison register for motion controller position strobe. *X_COMPARE* is an absolute position. The position pulse is available on output 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.**

X_COMPARE_REPEAT (0x6)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	X_COMPARE_REPEAT[23:16]							
Reset	0x000000							

Downloaded from Arrow.com.

Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	X_COMPARE_REPEAT[15:8]							
Reset	0x000000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	X_COMPARE_REPEAT[7:0]							
Reset	0x000000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
X_COMPARE_REPEAT	23:0	This register defines a relative distance in microsteps (based on *MRES* confguration). If set to >1, the position compare pulse is raised every time a multiple of *X_COMPARE_REPEAT* µsteps is made. *X_COMPARE* defines the position of the first pulse.	0x0: Repetition mechanism off. Number of microsteps between two pulses.

DRV_CONF (0xA)

Driver current range and slope settings.

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

Downloaded from Arrow.com.

Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	SLOPE_CONTROL[1:0]		CURRENT_RANGE_SCALE[1:0]		CURRENT_RANGE[1:0]	
Reset	–	–	0b11		0x3		0b00	
Access Type	–	–	Write, Read		Write, Read		Write, Read	

BITFIELD	BITS	DESCRIPTION	DECODE
SLOPE_CONTROL	5:4	Slope control setting	0x0: 100V/µs 0x1: 200V/µs 0x2: 400V/µs 0x3: 800V/µs
CURRENT_RANGE_SCALE	3:2	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.	0x0: 25% 0x1: 50% 0x2: 75% 0x3: 100%/normal setting
CURRENT_RANGE	1:0	This setting allows a basic adaptation of the drivers $R_{DS(ON)}$ current sensing to the motor current range. Select the lowest fitting range for best current precision. The value is the peak current setting.	0x0: 1A$_{RMS}$ 0x1: 2A$_{RMS}$ 0x2: 3A$_{RMS}$ 0x3: 4A$_{RMS}$

PLL (0xB)

PLL configuration and operation.

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

Downloaded from Arrow.com.

Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	clk_is_stuck	clk_loss	clk_1m0_tmo	–	–	CLOCK_DIVIDER[4:3]	
Reset	–				–	–	0d0	
Access Type	–	Write 1 to Clear, Read	Write 1 to Clear, Read	Write 1 to Clear, Read	–	–	Write, Read	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CLOCK_DIVIDER[2:0]			–	clk_fsm_ena	clk_sys_sel	ext_not_int	commit
Reset	0d0			–	0x0	0x0	0x0	0x0
Access Type	Write, Read			–	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
clk_is_stuck	14	Missing external clock signal. Also sets together with CLK loss and whenever new values are committed to the PLL.	0x0: Clock OK 0x1: Clock missing
clk_loss	13	Clock fault detection for external clock. Sets when the clock frequency or divider is out of range. Also sets whenever new values are committed to the PLL.	0x0: Clock OK 0x1: Clock is faulty or lost.
clk_1m0_tmo	12	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 correct parameters.	0x0: Clock OK 0x1: Clock timeout
CLOCK_DIVIDER	9:5	Clock divider value (set to have Input_clock/(CLOCK_DIVIDER + 1) = 1MHz).	0: 1MHz (pin CLK) 1: 2MHz (pin CLK) 2: 3MHz (pin CLK) 3: 4MHz (pin CLK) 4: 5MHz (pin CLK)

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
			5: 6MHz (pin CLK) 6: 7MHz (pin CLK) 7: 8MHz (pin CLK) 8: 9MHz (pin CLK) 9: 10MHz (pin CLK) 10: 11MHz (pin CLK) 11: 12MHz (pin CLK) 12: 13MHz (pin CLK) 13: 14MHz (pin CLK) 14: 15MHz (pin CLK) 15: 16MHz (pin CLK) or internal oscillator 16: 17MHz (pin CLK) 17: 18MHz (pin CLK) 18: 19MHz (pin CLK) 19: 20MHz (pin CLK) 20: 21MHz (pin CLK) 21: 22MHz (pin CLK) 22: 23MHz (pin CLK) 23: 24MHz (pin CLK) 24: 25MHz (pin CLK) 25: 26MHz (pin CLK) 26: 27MHz (pin CLK) 27: 28MHz (pin CLK) 28: 29MHz (pin CLK) 29: 30MHz (pin CLK) 30: 31MHz (pin CLK) 31: 32MHz (pin CLK)
clk_fsm_ena	3	Enable PLL FSM.	0x0: PLL disabled, chip goes to reset but registers bank is kept. 0x1: Chip in operation.
clk_sys_sel	2	Primary clock selector.	0x0: Internal 16MHz clock with PLL disabled. The chip remains in reset during this time. 0x1: PLL clock
ext_not_int	1		0x0: Select internal clock oscillator. 0x1: Select external clock signal applied to pin CLK.
commit	0	Commit changes to PLL logic (self-clear bit). This bit clears when the PLL is stabilized. Wait until the bit is cleared before resetting the error flags clk_loss and clk_is_stuck.	0x0: Write 0 to reset clk_loss and clk_is_stuck, and release the driver from the reset state. 0x1: Set to 1 after new valid values are progammed to ext_not_int, clk_sys_sel, and CLOCK_DIVIDER. The chip goes to a reset state (except for register space) during this time.

Velocity-Dependent Configuration Registers

Current setting and velocity thresholds for the choice of operation mode (based on TSTEP).

Register Details

IHOLD_IRUN (0x10)

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

Downloaded from Arrow.com.

BIT	23	22	21	20	19	18	17	16
Field	IHOLDDELAY[7:0]							
Reset	0x7							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	IRUN[7:0]							
Reset	0d240							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	IHOLD[7:0]							
Reset	0d64							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
IRUNDELAY	27:24	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.	0x0: Instant power-up 0x1.. 0xF: Delay per current increment step in multiples of *IRUNDELAY* × 64 clock.
IHOLDDELAY	23:16	Controls the number of clock cycles for motor power down after a motion as soon as standstill is detected (stst = 1) and *TPOWERDOWN* has expired. The smooth transition avoids a motor jerk upon power down.	0x0: Instant power down Delay per current reduction step in multiples of 2^{11} clocks.
IRUN	15:8	Motor run current scaler. Info: Choose CURRENT_RANGE in a way that normal IRUN is 128 to 255 for best microstep performance. StealthChop+ current settings 251 to 255 are clipped to 250 internally.	
IHOLD	7:0	Standstill current scaler. Info: In combination with StealthChop+, setting *IHOLD* = 0 allows to choose freewheeling or coil short circuit for motor standstill.	

Downloaded from Arrow.com.

TPOWERDOWN (0x11)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	7	6	5	4	3	2	1	0
Field	TPOWERDOWN[7:0]							
Reset	0d10							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TPOWERDOWN	7:0	*TPOWERDOWN* sets the delay time after standstill (stst) detection 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. **Hint:** Set *GCONF.fast_standstill* to achieve short power down times.

Downloaded from Arrow.com.

TSTEP (0x12)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	TSTEP[19:16]			
Reset	–	–	–	–	0x00000			
Access Type	–	–	–	–	Read Only			
BIT	15	14	13	12	11	10	9	8
Field	TSTEP[15:8]							
Reset	0x00000							
Access Type	Read Only							
BIT	7	6	5	4	3	2	1	0
Field	TSTEP[7:0]							
Reset	0x00000							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
TSTEP	19:0	Actual measured time between two 1/256 microsteps derived from the step input frequency in units of $1/f_{CLK}$. 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

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION
		in the range $(2^{24}/V) \leq TSTEP \leq 2^{24}/(V - 1)$.

TPWMTHRS (0x13)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	TPWMTHRS[19:16]			
Reset	–	–	–	–	0x00000			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	TPWMTHRS[15:8]							
Reset	0x00000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TPWMTHRS[7:0]							
Reset	0x00000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TPWMTHRS	19:0	Upper velocity for StealthChop+ operation. TSTEP ≥ TPWMTHRS StealthChop+ is enabled, if configured.

Downloaded from Arrow.com.

TCOOLTHRS (0x14)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	TCOOLTHRS[19:16]			
Reset	–	–	–	–	0x00000			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	TCOOLTHRS[15:8]							
Reset	0x00000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TCOOLTHRS[7:0]							
Reset	0x00000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TCOOLTHRS	19:0	Lower threshold velocity for switching on CoolStep/CoolStep+ and StallGuard/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 when exceeding this velocity. It is disabled again once the velocity falls below this threshold. *TCOOLTHRS* ≥ *TSTEP* ≥ *THIGH*: CoolStep(+) is enabled, if configured. *TCOOLTHRS* ≥ *TSTEP:*

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION
		Stop-on-stall is enabled, if configured. Stall output signal (D0/1) is enabled, if configured.

THIGH (0x15)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	THIGH[19:16]			
Reset	–	–	–	–	0x00000			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	THIGH[15:8]							
Reset	0x00000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	THIGH[7:0]							
Reset	0x00000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
THIGH	19:0	Upper velocity limit for CoolStep/CoolStep+. *TSTEP* < *THIGH* disables CoolStep and CoolStep+. Use together with *TPWMTRHS* for a smooth switch from StealthChop+ with CoolStep+ to Spreadcycle at full current.

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION
		THIGH > *TSTEP* > *TPWMTHRS:* CoolStep+ is disabled and current ramps to *IRUN*, as defined by *COOL_PI_OFF_SPEED.*

TSGP_LOW_VEL_THRS (0x16)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	TSGP_LOW_VEL_THRS[19:16]			
Reset	–	–	–	–	0x1000			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	TSGP_LOW_VEL_THRS[15:8]							
Reset	0x1000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TSGP_LOW_VEL_THRS[7:0]							
Reset	0x1000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
TSGP_LOW_VEL_THRS	19:0	*TSTEP*-based velocity threshold for low velocity StallGuard+-based stall detection. Set to a lower velocity/higher value than *TCOOLTHRS.* *TSGP_LOW_VEL_THRS* > *TSTEP* > *TCOOLTHRS:* low velocity StallGuard+ is enabled.

Downloaded from Arrow.com.

T_RCOIL_MEAS (0x17)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	T_RCOIL_MEAS[19:16]			
Reset	–	–	–	–	0x1000			
Access Type	–	–	–	–	Write, Read			
BIT	15	14	13	12	11	10	9	8
Field	T_RCOIL_MEAS[15:8]							
Reset	0x1000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	T_RCOIL_MEAS[7:0]							
Reset	0x1000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
T_RCOIL_MEAS	19:0	Upper velocity for automatic measurement of coil resistance. *TSTEP > T_RCOIL_MEAS*: Automatic resistance measurement enabled during low velocity motion or standstill. *TSTEP <= T_RCOIL_MEAS*: Automatic resistance measurement disabled.

TUDCSTEP (0x18)

BIT	31	30	29	28	27	26	25	24

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION
TUDCSTEP	19:0	Enable threshold for µDcStep. *TUDCSTEP* < *TSTEP*: µDcStep enabled. (Only while CoolStep+ is active).

UDC_CONF (0x19)

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

Downloaded from Arrow.com.

Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	udc_enable
Reset	–	–	–	–	–	–	–	
Access Type	–	–	–	–	–	–	–	Write, Read
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ACCEL_THRS[3:0]				DECEL_THRS[3:0]			
Reset	0xC				0xE			
Access Type	Write, Read				Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
udc_enable	8	Enable feature. Must be in StealthChop+ mode, Softstop must be activated, *TSTEP* < *TCOOL_THRS*, *TSTEP* > *THIGH*, *D1*, *D2*, *DMAX* of rampgen must be set.	0x0: disable udcstep 0x1: enable udcstep
ACCEL_THRS	7:4	Set < *DECEL_THRS*. If (*CS_ACTUAL* <= ((*ACCEL_THRS* + 1) × *IRUN*)/16), increase ramp velocity. In position mode, this might not happen as reaching the target is prioritized.	
DECEL_THRS	3:0	Set > *ACCEL_THRS*. If (*CS_ACTUAL* >= ((*DECEL_THRS* + 1) × *IRUN*)/16), reduce ramp velocity.	

Downloaded from Arrow.com.

STEPS_LOST (0x1A)

Lost steps counter for Step&Dir operation.

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

BITFIELD	BITS	DESCRIPTION
STEPS_LOST	19:0	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.

Ramp Generator Registers

Register Details

Downloaded from Arrow.com.

RAMPMODE (0x20)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	7	6	5	4	3	2	1	0
Field	–	–	–	–	–	–	RAMPMODE[1:0]	
Reset	–	–	–	–	–	–	0b00	
Access Type	–	–	–	–	–	–	Write, Read	

BITFIELD	BITS	DESCRIPTION	DECODE
RAMPMODE	1:0	Motion Controller ramping mode	0x0: Positioning mode (using all A, D, and V parameters). 0x1: Velocity mode to positive *VMAX* (using *A1, A2, AMAX, V1, V2, VMAX*). 0x2: Velocity mode to negative *VMAX* (using A1,A2, AMAX, V1, V2, VMAX). 0x3: Hold mode (velocity remains unchanged, unless stop event occurs).

XACTUAL (0x21)

BIT	31	30	29	28	27	26	25	24

Downloaded from Arrow.com.

Field	XACTUAL[31:24]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	XACTUAL[23:16]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	XACTUAL[15:8]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	XACTUAL[7:0]							
Reset	0x00000000							
Access Type	Write, Read							

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

VACTUAL (0x22)

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

Downloaded from Arrow.com.

Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	VACTUAL[23:16]							
Reset	0x000000							
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	VACTUAL[15:8]							
Reset	0x000000							
Access Type	Read Only							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	VACTUAL[7:0]							
Reset	0x000000							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
VACTUAL	23:0	Actual motor velocity from ramp generator (signed) The sign matches the motion direction. A negative sign means motion to lower *XACTUAL*. $\pm(2^{23}) - 1$ [µsteps/t]

VSTART (0x23)

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

Downloaded from Arrow.com.

BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	VSTART[17:16]	
Reset	–	–	–	–	–	–	0b000000000000000000	
Access Type	–	–	–	–	–	–	Write, Read	

BIT	15	14	13	12	11	10	9	8
Field	VSTART[15:8]							
Reset	0b000000000000000000							
Access Type	Write, Read							

BIT	7	6	5	4	3	2	1	0
Field	VSTART[7:0]							
Reset	0b000000000000000000							
Access Type	Write, Read							

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

A1 (0x24)

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

Downloaded from Arrow.com.

BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	A1[17:16]	
Reset	–	–	–	–	–	–	0b000000000000000000	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	A1[15:8]							
Reset	0b000000000000000000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	A1[7:0]							
Reset	0b000000000000000000							
Access Type	Write, Read							

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

V1 (0x25)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	V1[19:16]			
Reset	–	–	–	–	0x00000			

Downloaded from Arrow.com.

Access Type	–	–	–	–	Write, Read			
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	V1[15:8]							
Reset	0x00000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	V1[7:0]							
Reset	0x00000							
Access Type	Write, Read							

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

AMAX (0x26)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	AMAX[17:16]	
Reset	–	–	–	–	–	–	0b000000000000000000	
Access Type	–	–	–	–	–	–	Write, Read	

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	AMAX[15:8]							
Reset	0b000000000000000000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	AMAX[7:0]							
Reset	0b000000000000000000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
AMAX	17:0	Acceleration between *V2* and *VMAX* 0…(2^{18}) - 1 [µsteps/ta^2]

VMAX (0x27)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	VMAX[22:16]						
Reset	–	0b000000000000000000000						
Access Type	–	Write, Read						
BIT	15	14	13	12	11	10	9	8
Field	VMAX[15:8]							
Reset	0b000000000000000000000							

Downloaded from Arrow.com.

Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	VMAX[7:0]							
Reset	0b00000000000000000000000							
Access Type	Write, Read							

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

DMAX (0x28)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	DMAX[17:16]	
Reset	–	–	–	–	–	–	0b000000000000000000	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	DMAX[15:8]							
Reset	0b000000000000000000							
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	7	6	5	4	3	2	1	0
Field	DMAX[7:0]							
Reset	0b000000000000000000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
DMAX	17:0	Deceleration between *VMAX* and *V2* 0…(2^{18}) - 1 [µsteps/ta^2]

TVMAX (0x29)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	TVMAX[15:8]							
Reset	0x0000							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	TVMAX[7:0]							
Reset	0x0000							

Downloaded from Arrow.com.

Access Type	Write, Read

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

D1 (0x2A)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	D1[17:16]	
Reset	–	–	–	–	–	–	0b000000000000001010	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	D1[15:8]							
Reset	0b000000000000001010							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Downloaded from Arrow.com.

Field	D1[7:0]
Reset	0b000000000000001010
Access Type	Write, Read

BITFIELD	BITS	DESCRIPTION
D1	17:0	Deceleration between *V1* and *VSTOP* **Attention:** Do not set 0 in positioning mode, even if *V1* = 0! 1…(2^{16}) - 1 [µsteps/ta^2] Reset default = 10

VSTOP (0x2B)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	VSTOP[17:16]	
Reset	–	–	–	–	–	–	0b000000000000001010	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	VSTOP[15:8]							
Reset	0b000000000000001010							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	VSTOP[7:0]							

Downloaded from Arrow.com.

Reset	0b000000000000001010
Access Type	Write, Read

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

TZEROWAIT (0x2C)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	TZEROWAIT[15:8]							
Reset	0x0000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TZEROWAIT[7:0]							

Downloaded from Arrow.com.

Reset	0x0000
Access Type	Write, Read

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

XTARGET (0x2D)

BIT	31	30	29	28	27	26	25	24
Field	XTARGET[31:24]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	XTARGET[23:16]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	XTARGET[15:8]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	XTARGET[7:0]							
Reset	0x00000000							

Downloaded from Arrow.com.

Access Type	Write, Read

BITFIELD	BITS	DESCRIPTION
XTARGET	31:0	Target position for ramp mode. 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 $\pm((2^{31}) - 1)$. **Hint:** When increasing *V1*, *D1,* or *DMAX* during a motion, rewrite *XTARGET* afterwards to trigger a second acceleration phase, if desired. $-2^{31}\ldots+(2^{31}) - 1$

V2 (0x2E)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	V2[19:16]			
Reset	–	–	–	–	0x00000			
Access Type	–	–	–	–	Write, Read			
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	V2[15:8]							
Reset	0x00000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

Downloaded from Arrow.com.

Field	V2[7:0]
Reset	0x00000
Access Type	Write, Read

BITFIELD	BITS	DESCRIPTION
V2	19:0	Second acceleration/deceleration phase threshold velocity (unsigned). Uses A2/D2 while velocity is >V1 and < V2. 0: Disables *A2* and *D2* phase, use *AMAX*, *DMAX* only. 0…(2^{20}) - 1 [µsteps/t]

A2 (0x2F)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	A2[17:16]	
Reset	–	–	–	–	–	–	0b000000000000000000	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	A2[15:8]							
Reset	0b000000000000000000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	A2[7:0]							

Downloaded from Arrow.com.

Reset	0b000000000000000000
Access Type	Write, Read

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

D2 (0x30)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	D2[17:16]	
Reset	–	–	–	–	–	–	0b000000000000001010	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	D2[15:8]							
Reset	0b000000000000001010							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	D2[7:0]							
Reset	0b000000000000001010							
Access Type	Write, Read							

Downloaded from Arrow.com.

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

AACTUAL (0x31)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	AACTUAL[23:16]							
Reset	0x000000							
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	AACTUAL[15:8]							
Reset	0x000000							
Access Type	Read Only							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	AACTUAL[7:0]							
Reset	0x000000							
Access Type	Read Only							

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION
AACTUAL	23:0	Current acceleration used by the ramp generator.

Ramp Generator Driver Feature Control Registers

Register Details

SW_MODE (0x34)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	15	14	13	12	11	10	9	8
Field	hard_stop_clr_cur_int	virtual_stop_enc	en_virtual_stop_r	en_virtual_stop_l	en_softstop	sg_stop	en_latch_encoder	latch_r_inactive
Reset	0x1	0b0	0b0	0b0	0b0	0b0	0b0	0b0
Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read
BIT	7	6	5	4	3	2	1	0
Field	latch_r_active	latch_l_inactive	latch_l_active	swap_lr	pol_stop_r	pol_stop_l	stop_r_enable	stop_l_enable
Reset	0b0	0b0	0b0	0b0	0b0	0b0	0b0	0b0

Downloaded from Arrow.com.

Access Type	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
hard_stop_clr_cur_int	15	Allows clearing the current PI integrator to prevent overcurrent on hard stop.	0x0: Do not clear integrator. 0x1: Clear integrator upon hard stop (recommended).
virtual_stop_enc	14	Source for virtual stop (VIRTUAL_STOP_L and VIRTUAL_STOP_R).	0x0: Virtual stop relates to ramp generator position XACTUAL. 0x1: Virtual stop relates to encoder position X_ENC.
en_virtual_stop_r	13	Enables automatic motor stop during active right virtual stop condition.	0x0: Disable 0x1: Enable
en_virtual_stop_l	12	Enables automatic motor stop during active left virtual stop condition.	0x0: Disable 0x1: Enable
en_softstop	11	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.	0x0: Hard stop 0x1: Soft stop
sg_stop	10	Enable stop by StallGuard(+). Disable to release motor after 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*. **Note:** When used with STEP and DIR operation, an angle offset caused by the StealthChop+ angle regulator remains active until the motor is released by clearing the stall condition. This can lead to a position jerk of a several microsteps when clearing the stall condition.	0x0: disabled 0x1: enabled
en_latch_encoder	9	Latch encoder position to *ENC_LATCH* upon reference switch event.	0x0: Disable 0x1: Enable
latch_r_inactive	8	Activates latching of the position to *XLATCH* upon an inactive going edge on the right reference	0x0: Disable 0x1: Enable

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
		switch input REFR. The active level is defined by *pol_stop_r*.	
latch_r_active	7	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*.	0x0: Disable 0x1: Enable
latch_l_inactive	6	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*.	0x0: Disable 0x1: Enable
latch_l_active	5	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*.	0x0: Disable 0x1: Enable
swap_lr	4	Swap the left and right reference switch inputs REFL and REFR.	0x0: Normal order 0x1: Left and right switch swapped
pol_stop_r	3	Inverts the active polarity of the right reference switch input.	0x0: Noninverted, high active: a high level on REFR stops the motor. 0x1: Inverted, low active: a low level on REFR stops the motor.
pol_stop_l	2	Inverts the active polarity of the left reference switch input.	0x0: Noninverted, high active: a high level on REFL stops the motor. 0x1: Inverted, low active: a low level on REFL stops the motor.
stop_r_enable	1	Enables automatic motor stop during active right reference switch input. **Hint:** The motor restarts in case the stop switch is released.	0x0: Disable 0x1: Enable
stop_l_enable	0	Enables automatic motor stop during active left reference switch input. **Hint:** The motor restarts in case the stop switch is released.	0x0: Disable 0x1: Enable

RAMP_STAT (0x35)

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

Downloaded from Arrow.com.

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

BIT	15	14	13	12	11	10	9	8
Field	status_virtual_stop_r	status_virtual_stop_l	status_sg	second_move	t_zerowait_active	vzero	position_reached	velocity_reached
Reset	0b1	0b1	0b0	0b0	0b0	0b0	0b0	0b0
Access Type	Read Only	Read Only	Read Only	Write 1 to Clear, Read	Read Only	Read Only	Read Only	Read Only

BIT	7	6	5	4	3	2	1	0
Field	event_pos_reached	event_stop_sg	event_stop_r	event_stop_l	status_latch_r	status_latch_l	status_stop_r	status_stop_l
Reset	0b0	0b0	0b0	0b0	0b0	0b0	0b0	0b0
Access Type	Write 1 to Clear, Read	Write 1 to Clear, Read	Read Only	Read Only	Write 1 to Clear, Read	Write 1 to Clear, Read	Read Only	Read Only

BITFIELD	BITS	DESCRIPTION	DECODE
status_virtual_stop_r	15	Virtual reference switch right status	0x0: Inactive 0x1: Active
status_virtual_stop_l	14	Virtual reference switch left status	0x0: Inactive 0x1: Active
status_sg	13	Signals an active StallGuard2/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.	0x0: Momentary inactive 0x1: Momentary active
second_move	12	Signals that the automatic ramp required moving back in the opposite direction, example, due to on-the-fly parameter change.	0x0: Normal motion 0x1: Direction is changed to reach target.
t_zerowait_active	11	Signals that *TZEROWAIT* is active after a motor stop. During this time, the motor is in standstill.	0x0: Not active 0x1: Motion controller in *TZEROWAIT* time.

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
vzero	10	Signals that the actual velocity is 0.	0x0: Actual velocity is not 0. 0x1: Actual velocity is 0.
position_reached	9	Signals that the target position is reached. This flag is set while *XACTUAL* and *XTARGET* match.	0x0: *XTARGET* not reached 0x1: *XTARGET* reached
velocity_reached	8	Signals that the target velocity is reached. This flag is set while *VACTUAL* and *VMAX* match.	0x0: *VMAX* not reached 0x1: *VMAX* reached
event_pos_reached	7	Signals that the target position is reached (*position_reached* is active). (interrupt condition).	0x0: Inactive 0x1: Active
event_stop_sg	6	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. (interrupt condition)	0x0: Inactive 0x1: Active
event_stop_r	5	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.	0x0: Inactive 0x1: Active
event_stop_l	4	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.	0x0: Inactive 0x1: Active
status_latch_r	3	1: Latch right ready (enable position latching using *SW_MODE* settings *latch_r_active* or *latch_r_inactive*).	0x0: Not latched 0x1: Latched
status_latch_l	2	1: Latch left ready (enable position latching using *SW_MODE* settings *latch_l_active* or *latch_l_inactive*).	0x0: Not latched 0x1: Latched
status_stop_r	1	Reference switch right status	0x0: Inactive 0x1: Active

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
status_stop_l	0	Reference switch left status	0x0: Inactive 0x1: Active

XLATCH (0x36)

BIT	31	30	29	28	27	26	25	24
Field	XLATCH[31:24]							
Reset	0x00000000							
Access Type	Read Only							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	XLATCH[23:16]							
Reset	0x00000000							
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	XLATCH[15:8]							
Reset	0x00000000							
Access Type	Read Only							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	XLATCH[7:0]							
Reset	0x00000000							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
XLATCH	31:0	Ramp generator latch position, latches *XACTUAL* upon a programmable switch event (see *SW_MODE*). **Hint:** The encoder position can be latched to *ENC_LATCH* together with *XLATCH* to allow consistency checks.

Downloaded from Arrow.com.

Encoder Registers

Register Details

ENCMODE (0x38)

BIT	31	30	29	28	27	26	25	24
Field	–	–	BEMF_FILTER_SEL[1:0]		–	–	–	–
Reset	–	–			–	–	–	–
Access Type	–	–	Write, Read		–	–	–	–
BIT	23	22	21	20	19	18	17	16
Field	BEMF_BLANK_TIME[7:0]							
Reset	0d16							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	–	bemf_hyst[2:0]			nBEMF_ABN_SEL	enc_sel_decimal	latch_x_act	clr_enc_x
Reset	–					0b0	0b0	0b0
Access Type	–	Write, Read			Write, Read	Write, Read	Write, Read	Write, Read
BIT	7	6	5	4	3	2	1	0
Field	pos_neg_edge[1:0]		clr_once	clr_cont	ignore_AB	pol_N	pol_B	pol_A
Reset	0b00		0b0	0b0	0b0	0b0	0b0	0b0
Access Type	Write, Read		Write, Read	Write, Read	Write, Read	Write, Read	Write, Read	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
BEMF_FILTER_SEL	29:28	TriCoder filter corner frequency selection.	0x0: f_{CLK}/32000 (0.5kHz) 0x1: f_{CLK}/16000 (1kHz) 0x2: f_{CLK}/7800 (2kHz) 0x3: f_{CLK}/3700 (4.3kHz)
BEMF_BLANK_TIME	23:16	Blank time is *BEMF_BLANK_TIME* × 62,5ns ×((2^{14}) - 1).	
bemf_hyst	14:12		0x0: 10mV 0x1: 25mV

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
			0x2: 50mV 0x3: 75mV 0x4: 100mV 0x5: 150mV 0x6: 200mV 0x7: 250mV
nBEMF_ABN_SEL	11	X_ENC source selector	0x0: Back-EMF full step Tricoder 0x1: Incremental encoder interface
enc_sel_decimal	10	Encoder prescaler mode selection	0x0: Encoder prescaler divisor binary mode: counts *ENC_CONST(fractional part)*/65536. 0x1: Encoder prescaler divisor decimal mode: counts in *ENC_CONST(fractional part)*/10000.
latch_x_act	9	Position latch configuration	0x0: Disabled 0x1: Also latch *XACTUAL* position together with *X_ENC*. Allows latching the ramp generator position upon an N channel event as selected by *pos_edge* and *neg_edge*.
clr_enc_x	8	Encoder latch configuration	0x0: Upon N event, *X_ENC* is latched to *ENC_LATCH* only. 0x1: Latch and additionally clear encoder counter *X_ENC* at N-event.
pos_neg_edge	7:6	N channel event sensitivity	0x0: N channel event is active during an active N event level. 0x1: N channel is valid upon active going N event. 0x2: N channel is valid upon inactive going N event. 0x3: N channel is valid upon active going and inactive going N event.
clr_once	5	Position latch configuration	0x0: Disabled 0x1: Latch or latch and clear *X_ENC* on the next N event following the write access.
clr_cont	4	Position latch configuration	0x0: Disabled 0x1: Always latch or latch and clear *X_ENC* upon an N event (once per revolution, it is recommended to combine this setting with edge sensitive N event).
ignore_AB	3	N event configuration	0x0: An N event occurs only when polarities given by *pol_N*, *pol_A* and *pol_B* match. 0x1: Ignore A and B polarity for N channel event.
pol_N	2	Defines active polarity of N.	0x0: Low active 0x1: High active
pol_B	1	Required B polarity for an N channel event.	0x0: Negative polarity 0x1: Positive polarity
pol_A	0	Required A polarity for an N channel event.	0x0: Negative polarity 0x1: Positive polarity

X_ENC (0x39)

BIT	31	30	29	28	27	26	25	24
Field	X_ENC[31:24]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	X_ENC[23:16]							

Downloaded from Arrow.com.

Reset	0x00000000							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	X_ENC[15:8]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	X_ENC[7:0]							
Reset	0x00000000							
Access Type	Write, Read							

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

ENC_CONST (0x3A)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	ENC_CONST[31:24]							
Reset	0x00010000							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	ENC_CONST[23:16]							
Reset	0x00010000							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**

Downloaded from Arrow.com.

Field	ENC_CONST[15:8]							
Reset	0x00010000							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ENC_CONST[7:0]							
Reset	0x00010000							
Access Type	Write, Read							

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

ENC_STATUS (0x3B)

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

Downloaded from Arrow.com.

Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	–	–	–	–	deviation_warn	n_event
Reset	–	–	–	–	–	–	0b0	0b0
Access Type	–	–	–	–	–	–	Write 1 to Clear, Read	Write 1 to Clear, Read

BITFIELD	BITS	DESCRIPTION	DECODE
deviation_warn	1	Signals that X_ENC passed the ENC_DEVIATION setting. (Works only with Tricoder).	0x0: No warning 0x1: Deviation_warn cannot be cleared while a warning still persists. Set *ENC_DEVIATION* to zero to disable.
n_event	0	Signals detection of a previous or active N event. Write one to clear.	0x0: No event 0x1: Event detected

ENC_LATCH (0x3C)

BIT	31	30	29	28	27	26	25	24
Field	ENC_LATCH[31:24]							
Reset	0x00000000							
Access Type	Read Only							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	ENC_LATCH[23:16]							
Reset	0x00000000							
Access Type	Read Only							

Downloaded from Arrow.com.

BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	ENC_LATCH[15:8]							
Reset	0x00000000							
Access Type	Read Only							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ENC_LATCH[7:0]							
Reset	0x00000000							
Access Type	Read Only							

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

ENC_DEVIATION (0x3D)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	ENC_DEVIATION[19:16]			
Reset	–	–	–	–	0x00000			
Access Type	–	–	–	–	Write, Read			
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	ENC_DEVIATION[15:8]							
Reset	0x00000							

Downloaded from Arrow.com.

Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ENC_DEVIATION[7:0]							
Reset	0x00000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
ENC_DEVIATION	19:0	Maximum number of steps deviation between encoder counter and *XACTUAL* for deviation warning. Result in flag *ENC_STATUS.deviation_warn.* 0 = function is off.	0x0: Deviation check = off

VIRTUAL_STOP_L (0x3E)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	VIRTUAL_STOP_L[31:24]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	VIRTUAL_STOP_L[23:16]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	VIRTUAL_STOP_L[15:8]							
Reset	0x00000000							
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	VIRTUAL_STOP_L[7:0]							
Reset	0x00000000							
Access Type	Write, Read							

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

VIRTUAL_STOP_R (0x3F)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	VIRTUAL_STOP_R[31:24]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	VIRTUAL_STOP_R[23:16]							
Reset	0x00000000							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	VIRTUAL_STOP_R[15:8]							
Reset	0x00000000							
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	7	6	5	4	3	2	1	0
Field	VIRTUAL_STOP_R[7:0]							
Reset	0x00000000							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
VIRTUAL_STOP_R	31:0	Virtual stop switch based on encoder. A stop is raised, based on the signed comparison. *virtual_stop_enc* = 1: *X_ENC >= VIRTUAL_STOP_R* *virtual_stop_enc* = 0: *XACTUAL >= VIRTUAL_STOP_R* $-2^{31}\ldots+(2^{31}) - 1$

StealthChopPlus

Register Details

CURRENT_PI_REG (0x40)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	CUR_I[9:8]	
Reset	–	–	–	–	–	–	0d10	
Access Type	–	–	–	–	–	–	Write, Read	

BIT	23	22	21	20	19	18	17	16
Field	CUR_I[7:0]							
Reset	0d10							
Access Type	Write, Read							

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	CUR_P[11:8]			
Reset	–	–	–	–	0d64			

Downloaded from Arrow.com.

Access Type	–	–	–	–	Write, Read			
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CUR_P[7:0]							
Reset	0d64							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
CUR_I	25:16	I-parameter of StealthChop+ current regulator. **Hint:** when setting this value to 0, the integrator content is also cleared.	0x0: Clear current integrator.
CUR_P	11:0	P-parameter of StealthChop+ current regulator.	

ANGLE_PI_REG (0x41)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	ANGLE_I[9:8]	
Reset	–	–	–	–	–	–	0d20	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	ANGLE_I[7:0]							
Reset	0d20							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	ANGLE_P[11:8]			
Reset	–	–	–	–	0d50			
Access Type	–	–	–	–	Write, Read			

Downloaded from Arrow.com.

BIT	7	6	5	4	3	2	1	0
Field	ANGLE_P[7:0]							
Reset	0d50							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION	DECODE
ANGLE_I	25:16	I-parameter of StealthChop+ angle regulator. I = 0 clears the integrator.	0x0: Clear angle PI integrator.
ANGLE_P	11:0	P-parameter of StealthChop+ angle regulator.	

CUR_ANGLE_LIMIT (0x42)

BIT	31	30	29	28	27	26	25	24
Field	cur_pi_neg_clip	cur_pi_pos_clip	cur_pi_int_neg_clip	cur_pi_int_pos_clip	CUR_PI_LIMIT[11:8]			
Reset					0xFFF			
Access Type	Read Only	Read Only	Read Only	Read Only	Write, Read			
BIT	23	22	21	20	19	18	17	16
Field	CUR_PI_LIMIT[7:0]							
Reset	0xFFF							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	angle_pi_neg_clip	angle_pi_pos_clip	angle_pi_int_neg_clip	angle_pi_int_pos_clip	–	–	ANGLE_PI_LIMIT[9:8]	
Reset					–	–	0d256	
Access Type	Read Only	Read Only	Read Only	Read Only	–	–	Write, Read	
BIT	7	6	5	4	3	2	1	0

Downloaded from Arrow.com.

Field	ANGLE_PI_LIMIT[7:0]
Reset	0d256
Access Type	Write, Read

BITFIELD	BITS	DESCRIPTION	DECODE
cur_pi_neg_clip	31	Negative clipping of StealthChop+ current PI regulator to 0 (current too high).	0x0: No clipping 0x1: Clipping
cur_pi_pos_clip	30	Positive clipping of StealthChop+ current PI regulator to *CUR_PI_LIMIT* (current not reached).	0x0: No clipping 0x1: Clipping
cur_pi_int_neg_clip	29	Negative clipping of StealthChop+ current PI regulator I-part to 0 (current too high).	0x0: No clipping 0x1: Clipping
cur_pi_int_pos_clip	28	Positive clipping of StealthChop+ current PI regulator I-part to *CUR_PI_LIMIT* (current not reached).	0x0: No clipping 0x1: Clipping
CUR_PI_LIMIT	27:16	Unsigned limit for the output of the current regulator (limits PWM duty cycle by limiting *PWM_CALC*). Set to 4095 for no limitation (typical setting).	
angle_pi_neg_clip	15	Negative clipping of angle PI regulator to -*ANGLE_PI_LIMIT*.	0x0: No clipping 0x1: Clipping
angle_pi_pos_clip	14	Positive clipping of angle PI regulator to *ANGLE_PI_LIMIT*.	0x0: No clipping 0x1: Clipping
angle_pi_int_neg_clip	13	Negative clipping of angle PI regulator I-part to -*ANGLE_PI_LIMIT*.	0x0: No clipping 0x1: Clipping
angle_pi_int_pos_clip	12	Positive clipping of angle PI regulator I-part to *ANGLE_PI_LIMIT*.	0x0: No clipping 0x1: Clipping
ANGLE_PI_LIMIT	9:0	Unsigned limit for the output of the angle controller to ±ANGLE_PI_LIMIT (the default setting of 256 clips to +-90°, which is sufficient for normal operation and cannot cause overflow/underflow into the next current period).	

ANGLE_LOWER_LIMIT (0x43)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	ANGLE_ERROR[9:8]	
Reset	–	–	–	–	–	–		
Access Type	–	–	–	–	–	–	Read Only	

Downloaded from Arrow.com.

BIT	23	22	21	20	19	18	17	16
Field	ANGLE_ERROR[7:0]							
Reset								
Access Type	Read Only							

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	ANGLE_LOWER_I_LIMIT[9:8]	
Reset	–	–	–	–	–	–	0x100	
Access Type	–	–	–	–	–	–	Write, Read	

BIT	7	6	5	4	3	2	1	0
Field	ANGLE_LOWER_I_LIMIT[7:0]							
Reset	0x100							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
ANGLE_ERROR	25:16	Actual error of current angle as calculated by *MSCNT - ANGLE_MEAS.*
ANGLE_LOWER_I_LIMIT	9:0	Lower current limit for operation of angle regulator. If *AMPL_MEAS* <= *ANGLE_LOWER_I_LIMIT,* the angle regulator is disabled to avoid current measurement noise leading to regulation noise. **Attention:** Consider *IRUN* reduction, as selected by *COOL_CUR_DIV*, when setting a limit. CoolStep+ current reduction can lead to disabling the angle regulator when falling below this limit. This can compromise motor stability.

CUR_ANGLE_MEAS (0x44)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	ANGLE_MEAS[9:8]	
Reset	–	–	–	–	–	–		
Access Type	–	–	–	–	–	–	Read Only	

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION
ANGLE_MEAS	25:16	(Signed) current angle calculated from ADC values (±256 = ±90°).
AMPL_MEAS	11:0	Actual current amplitude calculated from *ADC_I_A* and *ADC_I_B*. This value is fed into the amplitude regulator. With a current scale of 250, the amplitude is regulated to a value of 2000. *AMPL_MEAS* covers a higher range to allow faster response of the PI regulator.

PI_RESULTS (0x45)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	ANGLE_CORR_CALC[9:8]	
Reset	–	–	–	–	–	–		
Access Type	–	–	–	–	–	–	Read Only	

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION
ANGLE_CORR_CALC	25:16	Output of angle regulator.
PWM_CALC	12:0	Output of current PI regulator for modulation of PWM amplitude 0...4095.

COIL_INDUCT (0x46)

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

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION	DECODE
rcoil_thermal_coupling	17	Enables thermal coupling for coil resistance measurement when in standstill. If one coil resistance is measurable (target current >50% of current setting and actually measured current >25% of current setting), the other coil resistance value is adapted according to the change of the measurable coil resistance if its actual current is below 50% of *IRUN*.	0x0: No thermal coupling 0x1: Thermal coupling
rcoil_manual	16	Selection of source for coil resistance for StallGuard+ and CoolStep+.	0x0: Automatic resistance measurement is used. 0x1: Resistance from user value is used.
COIL_INDUCT	14:0	Enter motor inductance in [uH] for use with StallGuard+ and CoolStep+.	

R_COIL (0x47)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	R_COIL_AUTO_A[11:8]			
Reset	–	–	–	–				
Access Type	–	–	–	–	Read Only			
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	R_COIL_AUTO_A[7:0]							

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION
R_COIL_AUTO_A	27:16	Measured coil resistance for coil A. It is measured when *TSTEP* > *T_RCOIL_MEAS*. The measurement is independent of the flag *rcoil_manual*. Coil resistance is updated only when the actual coil target current is above 50% of *IRUN* and measured current is above 25% of *IRUN*.
R_COIL_AUTO_B	11:0	Measured coil resistance for coil B. It is measured when *TSTEP* > *T_RCOIL_MEAS*. The measurement is independent of the flag *rcoil_manual*. Coil resistance is updated only when the actual coil target current is above 50% of *IRUN* and measured current is above 25% of *IRUN*.

R_COIL_USER (0x48)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	R_COIL_USER_A[11:8]			
Reset	–	–	–	–				
Access Type	–	–	–	–	Write, Read			
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	R_COIL_USER_A[7:0]							

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION
R_COIL_USER_A	27:16	Optional user value for coil resistance. Allows use of a software model for temperature dependent coil resistance. Set *rcoil_manual* to use.
R_COIL_USER_B	11:0	Optional user value for coil resistance. Allows use of a software model for temperature dependent coil resistance. Set *rcoil_manual* to use.

SGP_CONF (0x49)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	SGP_LOW_VEL_CNTS[1:0]		–	–	–	–
Reset	–	–			–	–	–	–
Access Type	–	–	Write, Read		–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	SGP_LOW_VEL_SLOPE[7:0]							
Reset								
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	–	sgp_clear_cur_pi	sgp_low_vel_freeze	sgp_filt_en	–	–	–	SGP_THRS[8]
Reset	–	0x1	0x0	0b0	–	–	–	
Access Type	–	Write, Read	Write, Read	Write, Read	–	–	–	Write, Read

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

BITFIELD	BITS	DESCRIPTION	DECODE
SGP_LOW_VEL_CNTS	29:28	Number of consecutive *SGP_RESULT* measurements with negative slope >*SGP_LOW_VEL_SLOPE* required for low velocity stall detection.	0x0: 1 event 0x1: 2 events 0x2: 3 events 0x3: 4 events
SGP_LOW_VEL_SLOPE	23:16	Low velocity gradient of SGP_RESULT for stall detection.	
sgp_clear_cur_pi	14	Allows clearing of the integral part of the current controller on stall detection in combination with hard stop. In SD_MODE, this flag also relates to hard stop initiated by external pins.	0x0: Disable 0x1: Enable
sgp_low_vel_freeze	13	Determines handling of *SGP_RESULT* in very low velocity (*TSTEP*>TSGP_LOW_VEL_THRS). Freezing vs. Zeroing	0x0: *SGP_RESULT* = 0 in very low velocity. 0x1: *SGP_RESULT* freeze
sgp_filt_en	12	Enable *SGP_RESULT* filtering over one electrical period.	0x0: Disable 0x1: Enable
SGP_THRS	8:0	Stall threshold for StallGuard with *SGP_RESULT*-based stall detection.	

SGP_IND_2_3 (0x4A)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	SGP_IND_3[9:8]	
Reset	–	–	–	–	–	–		

Downloaded from Arrow.com.

Access Type	–	–	–	–	–	–	Read Only	
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	SGP_IND_3[7:0]							
Reset								
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	SGP_IND_2[9:8]	
Reset	–	–	–	–	–	–	0b0000000000	
Access Type	–	–	–	–	–	–	Read Only	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SGP_IND_2[7:0]							
Reset	0b0000000000							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
SGP_IND_3	25:16	Individual StallGuard+ measurement for MSCNT index range 768 to 1023.
SGP_IND_2	9:0	Individual StallGuard+ measurement for MSCNT index range 512 to 767.

SGP_IND_0_1 (0x4B)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	SGP_IND_1[9:8]	
Reset	–	–	–	–	–	–		
Access Type	–	–	–	–	–	–	Read Only	
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**

Downloaded from Arrow.com.

Field	SGP_IND_1[7:0]							
Reset								
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	SGP_IND_0[9:8]	
Reset	–	–	–	–	–	–	0x00	
Access Type	–	–	–	–	–	–	Read Only	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SGP_IND_0[7:0]							
Reset	0x00							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
SGP_IND_1	25:16	Individual StallGuard+ measurement for MSCNT index range 256 to 511.
SGP_IND_0	9:0	Individual StallGuard+ measurement for MSCNT index range 0 to 255.

INDUCTANCE_VOLTAGE (0x4C)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	UL_A[11:8]			
Reset	–	–	–	–				
Access Type	–	–	–	–	Read Only			
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	UL_A[7:0]							
Reset								
Access Type	Read Only							

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	UL_B[11:8]			
Reset	–	–	–	–				
Access Type	–	–	–	–	Read Only			

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

BITFIELD	BITS	DESCRIPTION
UL_A	27:16	Informative internal calculation result: voltage drop on coil inductivity calculated from coil inductivity and motor voltage, and scaled to V_S range (14-bit signed clipped to 12-bit signed, that is, full scale = 25% of voltage).
UL_B	11:0	Informative internal calculation result: voltage drop on coil inductivity calculated from coil inductivity and motor voltage, and scaled to V_S range (14-bit signed clipped to 12-bit signed, that is, full scale = 25% of voltage).

SGP_BEMF (0x4D)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	UBEMF_ABS[11:8]			
Reset	–	–	–	–				
Access Type	–	–	–	–	Read Only			

BIT	23	22	21	20	19	18	17	16
Field	UBEMF_ABS[7:0]							
Reset								
Access Type	Read Only							

Downloaded from Arrow.com.

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

BITFIELD	BITS	DESCRIPTION
UBEMF_ABS	27:16	Informative internal calculation result: amplitude of calculated back-emf.
SGP_RAW	9:0	Informative internal calculation result: StallGuard+ load reserve unfiltered.

COOLSTEPPLUS_CONF (0x4E)

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

Downloaded from Arrow.com.

Access Type	–	–	–	–	–	–	–	–
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	–	–	load_filt_en	COOL_CUR_DIV[3:0]			
Reset	–	–	–	0x1	0x2			
Access Type	–	–	–	Write, Read	Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
load_filt_en	4	Enable RC-filter over 16 measurements on SG5_RAW value to filter coolstep input.	0x0: Disable 0x1: Enable (recommended)
COOL_CUR_DIV	3:0	CoolStepPlus current reduction factor motor run current is ramped down to the programmed fraction of *IRUN*. Minimum current is *IRUN/COOL_CUR_DIV*.	0x0: CoolStepPlus off 0x1: CoolStepPlus off, use *PI_OFF_SPEED* for current increment to full current. 0x2: Reduce to 50% of *IRUN*. 0x3: Reduce to 1/3 of *IRUN*. 0x4: Reduce to 1/4 of *IRUN*. 0x5: Reduce to 1/5 of *IRUN*. 0x6: Reduce to 1/6 of *IRUN*. 0x7: Reduce to 1/7 of *IRUN*. 0x8: Reduce to 1/8 of *IRUN*. 0x9: Reduce to 1/9 of *IRUN*. 0xA: Reduce to 1/10 of *IRUN*. 0xB: Reserved 0xC: Reserved 0xD: Reserved 0xE: Reserved 0xF: Reserved

COOLSTEPPLUS_PI_REG (0x4F)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	COOLSTEP_I[9:8]	
Reset	–	–	–	–	–	–	0d16	
Access Type	–	–	–	–	–	–	Write, Read	
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	COOLSTEP_I[7:0]							
Reset	0d16							
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	COOLSTEP_P[11:8]			
Reset	–	–	–	–	0d128			
Access Type	–	–	–	–	Write, Read			
BIT	7	6	5	4	3	2	1	0
Field	COOLSTEP_P[7:0]							
Reset	0d128							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
COOLSTEP_I	25:16	I-parameter for CoolStepPlus target current regulator.
COOLSTEP_P	11:0	P-parameter for CoolStepPlus target current regulator.

COOLSTEPPLUS_PI_DOWN (0x50)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	COOL_PI_OFF_SPEED[11:8]			
Reset	–	–	–	–	0d64			
Access Type	–	–	–	–	Write, Read			
BIT	23	22	21	20	19	18	17	16
Field	COOL_PI_OFF_SPEED[7:0]							
Reset	0d64							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	COOL_PI_DOWN_LIMIT[11:8]			
Reset	–	–	–	–	0d128			

Downloaded from Arrow.com.

Access Type	–	–	–	–	Write, Read			
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	COOL_PI_DOWN_LIMIT[7:0]							
Reset	0d128							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
COOL_PI_OFF_SPEED	27:16	Current slope when CoolStepPlus is turned off. *COOL_PI_OFF_SPEED* simulates a regulation error, leading to current reduction.
COOL_PI_DOWN_LIMIT	11:0	Limits the falling slope of CoolStepPlus target current. Maximum reduction is 1/256 *COOL_PI_DOWN_LIMIT* per 32 chopper periods.

COOLSTEPPLUS_RESERVE_CONF (0x51)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	COOL_HI_GENERATORIC_RESERVE[7:0]							
Reset	0d100							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	COOL_LOW_GENERATORIC_RESERVE[7:0]							
Reset	0d220							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	COOL_HI_LOAD_RESERVE[7:0]							
Reset	0d50							
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	7	6	5	4	3	2	1	0
Field	COOL_LOW_LOAD_RESERVE[7:0]							
Reset	0d150							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
COOL_HI_GENERATORIC_RESERVE	31:24	Load reserve used in generatoric mode when CoolStep current = *IRUN*. Ensure *COOL_HI_GENERATORIC_RESERVE* < *COOL_LOW_GENERATORIC_RESERVE* for best stability of regulation.
COOL_LOW_GENERATORIC_RESERVE	23:16	Load reserve used in generatoric mode when CoolStep current is at minimum level. Ensure *COOL_HI_GENERATORIC_RESERVE* < *COOL_LOW_GENERATORIC_RESERVE* for best stability of regulation.
COOL_HI_LOAD_RESERVE	15:8	Load reserve used when CoolStep current = *IRUN*. Ensure *COOL_HI_LOAD_RESERVE* < *COOL_LOW_LOAD_RESERVE* for best stability of regulation.
COOL_LOW_LOAD_RESERVE	7:0	Load reserve used at minimum CoolStep current. Ensure *COOL_HI_LOAD_RESERVE* < *COOL_LOW_LOAD_RESERVE* for best stability of regulation.

COOLSTEPPLUS_LOAD_RESERVE (0x52)

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

Downloaded from Arrow.com.

Field	–	–	–	–	–	–	SGP_RESULT[9:8]	
Reset	–	–	–	–	–	–		
Access Type	–	–	–	–	–	–	Read Only	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SGP_RESULT[7:0]							
Reset								
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
COOLSTEP_LOAD_RESERVE	24:16	Load reserve currently used by CoolStepPlus.
SGP_RESULT	9:0	*SGP_RAW* value filtered over the last or last four fullstep segments. Signed load value for StealthChop+ operation. A value near 0 corresponds to maximum motor load, a value from 200 to 255 means no or low motor load. Range 256 to 511 is generatoric operation, negative values occur upon overload, and typically indicate a step loss, respectively, danger of step loss. *sgp_filt_en* = 0: SGP_RAW values averaged over the last fullstep segment. *sgp_filt_en* = 1: *SGP_RAW* values averaged over the last four fullstep segments.

TSTEP Velocity (0x53)

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

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	TSTEP_VELOCITY[15:8]							
Reset								
Access Type	Read Only							

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

BITFIELD	BITS	DESCRIPTION
TSTEP_VELOCITY	22:0	Actual velocity derived from TSTEP value (used for StallGuardPlus in STEP&DIR operation).

ADC Registers

Register Details

ADC_VSUPPLY_TEMP (0x58)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	ADC_TEMP[8]
Reset	–	–	–	–	–	–	–	0b0000000000000
Access Type	–	–	–	–	–	–	–	Read Only

BIT	23	22	21	20	19	18	17	16
Field	ADC_TEMP[7:0]							
Reset	0b0000000000000							
Access Type	Read Only							

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	ADC_VSUPPLY[8]
Reset	–	–	–	–	–	–	–	0b0000000000000
Access Type	–	–	–	–	–	–	–	Read Only
BIT	7	6	5	4	3	2	1	0
Field	ADC_VSUPPLY[7:0]							
Reset	0b0000000000000							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ADC_TEMP	24:16	Temperature [°C] = (1.042 x ADC_TEMP - 264.6)°C
ADC_VSUPPLY	8:0	V_S [V] = 0,1409V × ADC_VSUPPLY

ADC_I (0x59)

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

Downloaded from Arrow.com.

Access Type	–	–	–	–	Read Only			
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ADC_I_A[7:0]							
Reset								
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION
ADC_I_B	27:16	Measured momentary coil B current in StealthChopPlus operation.
ADC_I_A	11:0	Measured momentary coil A current in StealthChopPlus operation.

OTW_OV_VTH (0x5A)

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	–	–	–	–	–	–	–	OVERTEMPPREWARNING_VTH[8]
Reset	–	–	–	–	–	–	–	0x1FF
Access Type	–	–	–	–	–	–	–	Write, Read
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	OVERTEMPPREWARNING_VTH[7:0]							
Reset	0x1FF							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	OVERVOLTAGE_VTH[8]
Reset	–	–	–	–	–	–	–	0x1FF
Access Type	–	–	–	–	–	–	–	Write, Read

Downloaded from Arrow.com.

BIT	7	6	5	4	3	2	1	0
Field	OVERVOLTAGE_VTH[7:0]							
Reset	0x1FF							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
OVERTEMPPREWARNING_VTH	24:16	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.
OVERVOLTAGE_VTH	8:0	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.

Motor Driver Registers

Register Details

MSLUT_0 (0x60)

Microstep table entries 0...31

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

Downloaded from Arrow.com.

Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MSLUT_0[7:0]							
Reset	0xAAAAB554							
Access Type	Write, Read							

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

MSLUT_1 (0x61)

Microstep table entries 32…63

BIT	**31**	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Field	MSLUT_1[31:24]							
Reset	0x4A9554AA							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	MSLUT_1[23:16]							
Reset	0x4A9554AA							

Downloaded from Arrow.com.

Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	MSLUT_1[15:8]							
Reset	0x4A9554AA							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MSLUT_1[7:0]							
Reset	0x4A9554AA							
Access Type	Write, Read							

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

MSLUT_2 (0x62)

Microstep table entries 64...95

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_2[31:24]							
Reset	0x24492929							

Downloaded from Arrow.com.

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

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

Downloaded from Arrow.com.

MSLUT_3 (0x63)

Microstep table entries 96...127

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_3[31:24]							
Reset	0x10104222							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	MSLUT_3[23:16]							
Reset	0x10104222							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	MSLUT_3[15:8]							
Reset	0x10104222							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	MSLUT_3[7:0]							
Reset	0x10104222							
Access Type	Write, Read							

BITFIELD	BITS	DESCRIPTION
MSLUT_3	31:0	Each bit gives the difference between entry x and entry x+1 when combined with the corresponding *MSLUTSEL W* bits: 0: *W* = %00: -1 %01: +0 %10: +1 %11: +2 1: *W* = %00: +0 %01: +1 %10: +2 %11: +3

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION
		This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90*. *ofs31, ofs30, …, ofs01, ofs00* … *ofs255, ofs254, …, ofs225, ofs224.* Reset default = sine wave table

MSLUT_4 (0x64)

Microstep table entries 128…159

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_4[31:24]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	MSLUT_4[23:16]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	MSLUT_4[15:8]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	MSLUT_4[7:0]							
Reset	0xFBFFFFFF							
Access Type	Write, Read							

Downloaded from Arrow.com.

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

MSLUT_5 (0x65)

Microstep table entries 160…191

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

Downloaded from Arrow.com.

BIT	7	6	5	4	3	2	1	0
Field	MSLUT_5[7:0]							
Reset	0xB5BB777D							
Access Type	Write, Read							

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

MSLUT_6 (0x66)

Microstep table entries 192...223

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_6[31:24]							
Reset	0x49295556							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	MSLUT_6[23:16]							
Reset	0x49295556							
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	15	14	13	12	11	10	9	8
Field	MSLUT_6[15:8]							
Reset	0x49295556							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MSLUT_6[7:0]							
Reset	0x49295556							
Access Type	Write, Read							

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

MSLUT_7 (0x67)

Microstep table entries 224…255

BIT	31	30	29	28	27	26	25	24
Field	MSLUT_7[31:24]							
Reset	0x00404222							
Access Type	Write, Read							

Downloaded from Arrow.com.

BIT	23	22	21	20	19	18	17	16
Field	MSLUT_7[23:16]							
Reset	0x00404222							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	MSLUT_7[15:8]							
Reset	0x00404222							
Access Type	Write, Read							
BIT	7	6	5	4	3	2	1	0
Field	MSLUT_7[7:0]							
Reset	0x00404222							
Access Type	Write, Read							

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

MSLUTSEL (0x68)

BIT	31	30	29	28	27	26	25	24

Downloaded from Arrow.com.

Field	X3[7:0]							
Reset	0xFF							
Access Type	Write, Read							
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	X2[7:0]							
Reset	0xFF							
Access Type	Write, Read							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	X1[7:0]							
Reset	0x80							
Access Type	Write, Read							
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	W3[1:0]		W2[1:0]		W1[1:0]		W0[1:0]	
Reset	0b01		0b01		0b01		0b10	
Access Type	Write, Read		Write, Read		Write, Read		Write, Read	

BITFIELD	BITS	DESCRIPTION	DECODE
X3	31:24	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*	

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
X2	23:16	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*	
X1	15:8	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*	
W3	7:6	LUT width select from *ofs*(*X3*) to *ofs255* Width control bit coding *W0*…*W3*: %00: MSLUT entry 0, 1 select: -1, +0 %01: MSLUT entry 0, 1 select: +0, +1 %10: MSLUT entry 0, 1 select: +1, +2 %11: MSLUT entry 0, 1 select: +2, +3	0x0: Current MSLUT entry 0 (1): -1 (+0) to sinewave. 0x1: Current MSLUT entry 0 (1): +0 (+1) to sinewave. 0x2: Current MSLUT entry 0 (1): +1 (+2) to sinewave. 0x3: Current MSLUT entry 0 (1): +2 (+3) to sinewave.
W2	5:4	LUT width select from *ofs*(*X2*) to *ofs*(*X3*-1) Width control bit coding *W0*…*W3*: %00: MSLUT entry 0, 1 select: -1, +0 %01: MSLUT entry 0, 1 select: +0, +1 %10: MSLUT entry 0, 1 select: +1, +2 %11: MSLUT entry 0, 1 select: +2, +3	0x0: Current MSLUT entry 0 (1): -1 (+0) to sinewave. 0x1: Current MSLUT entry 0 (1): +0 (+1) to sinewave. 0x2: Current MSLUT entry 0 (1): +1 (+2) to sinewave. 0x3: Current MSLUT entry 0 (1): +2 (+3) to sinewave.
W1	3:2	LUT width select from *ofs*(*X1*) to *ofs*(*X2*-1) Width control bit coding *W0*…*W3*: %00: MSLUT entry 0, 1 select: -1, +0 %01: MSLUT entry 0, 1 select: +0, +1	0x0: Current MSLUT entry 0 (1): -1 (+0) to sinewave. 0x1: Current MSLUT entry 0 (1): +0 (+1) to sinewave. 0x2: Current MSLUT entry 0 (1): +1 (+2) to sinewave. 0x3: Current MSLUT entry 0 (1): +2 (+3) to sinewave.

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
		%10: MSLUT entry 0, 1 select: +1, +2 %11: MSLUT entry 0, 1 select: +2, +3	
W0	1:0	LUT width select from *ofs00* to *ofs*(*X1*-1) Width control bit coding *W0*…*W3*: %00: MSLUT entry 0, 1 select: -1, +0 %01: MSLUT entry 0, 1 select: +0, +1 %10: MSLUT entry 0, 1 select: +1, +2 %11: MSLUT entry 0, 1 select: +2, +3	0x0: Current MSLUT entry 0 (1): -1 (+0) to sinewave. 0x1: Current MSLUT entry 0 (1): +0 (+1) to sinewave. 0x2: Current MSLUT entry 0 (1): +1 (+2) to sinewave. 0x3: Current MSLUT entry 0 (1): +2 (+3) to sinewave.

MSLUTSTART (0x69)

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

BIT	31	30	29	28	27	26	25	24
Field	OFFSET_SIN90[7:0]							
Reset	0x00							
Access Type	Write, Read							
BIT	23	22	21	20	19	18	17	16
Field	START_SIN90[7:0]							
Reset	0xF7							
Access Type	Write, Read							
BIT	15	14	13	12	11	10	9	8
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	7	6	5	4	3	2	1	0
Field	START_SIN[7:0]							
Reset	0x00							

Downloaded from Arrow.com.

Access Type	Write, Read

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

MSCNT (0x6A)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	MSCNT[9:8]	
Reset	–	–	–	–	–	–	0b0000000000	
Access Type	–	–	–	–	–	–	Read Only	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MSCNT[7:0]							
Reset	0b0000000000							
Access Type	Read Only							

Downloaded from Arrow.com.

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

MSCURACT (0x6B)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	CUR_A[8]
Reset	–	–	–	–	–	–	–	0b011110111
Access Type	–	–	–	–	–	–	–	Read Only
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	CUR_A[7:0]							
Reset	0b011110111							
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	–	–	–	–	–	–	CUR_B[8]
Reset	–	–	–	–	–	–	–	0b000000000
Access Type	–	–	–	–	–	–	–	Read Only
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CUR_B[7:0]							
Reset	0b000000000							
Access Type	Read Only							

Downloaded from Arrow.com.

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

CHOPCONF (0x6C)

BIT	31	30	29	28	27	26	25	24
Field	–	–	dedge	intpol	MRES[3:0]			
Reset	–	–	0b0	0b1	0x0			
Access Type	–	–	Write, Read	Write, Read	Write, Read			
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	TPFD[3:0]				–	–	–	TBL[1]
Reset	0x4				–	–	–	0d2
Access Type	Write, Read				–	–	–	Write, Read
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	TBL[0]	chm	–	disfdcc	fd3	HEND_OFFSET[3:1]		
Reset	0d2	0b0	–	0b0	0b0	0x2		
Access Type	Write, Read	Write, Read	–	Write, Read	Write, Read	Write, Read		
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	HEND_OFFSET[0]	HSTRT_TFD210[2:0]			TOFF[3:0]			
Reset	0x2	0b101			0x0			
Access Type	Write, Read	Write, Read			Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
dedge	29	Enable double edge step pulses.	0x0: Disabled 0x1: Enable step impulse at each step edge to reduce step frequency requirement.

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
intpol	28	Interpolation to 256 microsteps.	0x0: No interpolation 0x1: The actual microstep resolution (*MRES*) is extrapolated to 256 microsteps for smoothest motor operation (useful for STEP and DIR operation, only).
MRES	27:24	Microstep resolution selection. A reduced microstep resolution can be configured, especially for STEP and DIR operation. The resolution gives the number of microstep entries per sine quarter wave. The driver automatically uses microstep positions, which result in a symmetrical wave when choosing and switching to a lower microstep resolution. Step width = 2^{MRES} [microsteps]	0x0: Native 256 microstep setting. Normally use this setting with the internal motion controller. 0x1: 128 0x2: 64 0x3: 32 0x4: 16 0x5: 8 0x6: 4 0x7: 2 0x8: FULLSTEP 0x9: FULLSTEP 0xA: FULLSTEP 0xB: FULLSTEP 0xC: FULLSTEP 0xD: FULLSTEP 0xE: FULLSTEP 0xF: FULLSTEP
TPFD	23:20	Passive fast decay time. *TPFD* allows dampening of motor mid-range resonances. Passive fast decay time setting controls duration of the fast decay phase inserted after bridge polarity change. N_{CLK} = 8µs × *TPFD* %0000: Disable %0001 … %1111: 1 … 15	
TBL	16:15	Blank time selection for current measurement. **Note:** 1 or 2 is recommended for most applications.	0x0: ~1.25µs 0x1: ~1.75µs 0x2: ~2.5µs 0x3: ~3.625µs
chm	14	Chopper mode	0x0: Standard mode (SpreadCycle) 0x1: Constant off-time with fast decay time. Fast decay time is also terminated when the negative nominal current is reached. Fast decay is after on-time.
disfdcc	12	Fast decay mode. With *chm* = 1: *disfdcc* = 1 disables current comparator usage for termination of the fast decay cycle.	0x0: Comparator terminates fast decay cycle 0x1: Fixed fast decay cycle
fd3	11	TFD[3] With *chm* = 1: MSB of fast decay time setting *TFD.*	
HEND_OFFSET	10:7	With *chm* = 0: *HEND* hysteresis low value. %0000 … %1111: Hysteresis is -3, -2, -1, 0, 1, …, 12. (1/512 of this setting adds to current setting). This is the hysteresis value used for the hysteresis chopper.	

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
		With *chm* = 1: *OFFSET* sine wave offset. %0000 … %1111: Offset is -3, -2, -1, 0, 1, …, 12. This is the sine wave offset and 1/512 of the value is added to the absolute value of each sine wave entry.	
HSTRT_TFD210	6:4	With *chm* = 0: *HSTRT* hysteresis start value added to *HEND.* %000 … %111: Add 1, 2, …, 8 to hysteresis low value HEND. (1/512 of this setting adds to current setting). **Attention:** Effective *HEND* + *HSTRT* ≤ 16. **Note:** Hysteresis decrement is done each 16 clocks. With *chm* = 1: *TFD* [2..0] fast decay time setting. Fast decay time setting (MSB: *fd3*): %0000 … %1111: Fast decay time setting *TFD* with N_{CLK} = 32 x *TFD* (%0000: slow decay only).	
TOFF	3:0	*TOFF* off-time and driver enable Off-time setting controls duration of slow decay phase. N_{CLK} = 24 + 32 x *TOFF* %0000: Driver disable, all bridges off. %0001: 1 – use only with *TBL* ≥ 2. %0010 … %1111: 2 … 15	0x0: Driver disabled and all bridges off. 0x1: Slow decay phase duration ~3.5 us 0x2: Slow decay phase duration ~5.5 us 0x3: Slow decay phase duration ~7.5 us 0x4: Slow decay phase duration ~9.5 us 0x5: Slow decay phase duration ~11.5 us 0x6: Slow decay phase duration ~13.5 us 0x7: Slow decay phase duration ~15.5 us 0x8: Slow decay phase duration ~17.5 us 0x9: Slow decay phase duration ~19.5 us 0xA: Slow decay phase duration ~21.5 us 0xB: Slow decay phase duration ~23.5 us 0xC: Slow decay phase duration ~25.5 us 0xD: Slow decay phase duration ~27.5 us 0xE: Slow decay phase duration ~29.5 us 0xF: Slow decay phase duration ~31.5 us

COOLCONF (0x6D)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	sfilt
Reset	–	–	–	–	–	–	–	0b0
Access Type	–	–	–	–	–	–	–	Write, Read

Downloaded from Arrow.com.

BIT	23	22	21	20	19	18	17	16
Field	thigh_sg_off	sgt[6:0]						
Reset		0b0000000						
Access Type	Write, Read	Write, Read						
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	seimin	sedn[2:0]			semax[3:0]			
Reset	0b0	0b000			0x0			
Access Type	Write, Read	Write, Read			Write, Read			
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	–	seup[1:0]		–	semin[3:0]			
Reset	–	0b00		–	0x0			
Access Type	–	Write, Read		–	Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
sfilt	24	StallGuard2 filter enable.	0x0: Standard mode, high time resolution for StallGuard. 0x1: Filtered mode, StallGuard signal updated for each four fullsteps only to compensate for motor pole tolerances.
thigh_sg_off	23	Stop-on-stall handling when exceeding *VHIGH.*	0x0: Exceeding *VHIGH* only disables CoolStep. 0x1: Exceeding *VHIGH* disables stop-on-stall function as well as CoolStep.
sgt	22:16	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.	
seimin	15	Minimum current for smart current control.	0x0: 1/2 of current setting (*IRUN*). 0x1: 1/4 of current setting (*IRUN*).
sedn	14:12	Current down step speed.	0x0: For each 8 StallGuard2 values, decrease by one. 0x1: For each 4 StallGuard2 values, decrease by one. 0x2: For each 2 StallGuard2 values, decrease by one. 0x3: For each StallGuard2 value, decrease by one. 0x4: For each StallGuard2 value, decrease by two. 0x5: For each StallGuard2 value, decrease by four.

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
			0x6: For each StallGuard2 value, decrease by eight. 0x7: For each StallGuard2 value, decrease by sixteen.
semax	11:8	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.	0x0: Reduce current when (32 × SEMIN + 32) <= SG_RESULT. 0x1: Reduce current when (32 × SEMIN + 64) <= SG_RESULT. 0x2: Reduce current when (32 × SEMIN + 96) <= SG_RESULT. 0x3: Reduce current when (32 × SEMIN + 128) <= SG_RESULT. 0x4: Reduce current when (32 × SEMIN + 160) <= SG_RESULT. 0x5: Reduce current when (32 × SEMIN + 192) <= SG_RESULT. 0x6: Reduce current when (32 × SEMIN + 224) <= SG_RESULT. 0x7: Reduce current when (32 × SEMIN + 256) <= SG_RESULT. 0x8: Reduce current when (32 × SEMIN + 288) <= SG_RESULT. 0x9: Reduce current when (32 × SEMIN + 320) <= SG_RESULT. 0xA: Reduce current when (32 × SEMIN + 352) <= SG_RESULT. 0xB: Reduce current when (32 × SEMIN + 384) <= SG_RESULT. 0xC: Reduce current when (32 × SEMIN + 416) <= SG_RESULT. 0xD: Reduce current when (32 × SEMIN + 448) <= SG_RESULT. 0xE: Reduce current when (32 × SEMIN + 480) <= SG_RESULT. 0xF: Reduce current when (32 × SEMIN + 512) <= SG_RESULT.
seup	6:5	Current up step width. Current increment steps per measured StallGuard2 value.	0x0: 8 0x1: 16 0x2: 32 0x3: 64
semin	3:0	Minimum StallGuard2 value for smart current control and enable. If the StallGuard2 result falls below *SEMIN* × 32, the motor current is increased to reduce motor load angle.	0x0: CoolStep smart current control off. 0x1: Increase current on SG_RESULT < 32. 0x2: Increase current on SG_RESULT < 64. 0x3: Increase current on SG_RESULT < 96. 0x4: Increase current on SG_RESULT < 128. 0x5: Increase current on SG_RESULT < 160. 0x6: Increase current on SG_RESULT < 192. 0x7: Increase current on SG_RESULT < 224. 0x8: Increase current on SG_RESULT < 256. 0x9: Increase current on SG_RESULT < 288. 0xA: Increase current on SG_RESULT < 320. 0xB: Increase current on SG_RESULT < 352. 0xC: Increase current on SG_RESULT < 384. 0xD: Increase current on SG_RESULT < 416. 0xE: Increase current on SG_RESULT < 448. 0xF: Increase current on SG_RESULT < 480.

DRV_STATUS (0x6F)

BIT	31	30	29	28	27	26	25	24
Field	stst	olb	ola	s2gb	s2ga	otpw	ot	stallguard
Reset	0b0	0b0	0b0	0b0	0b0	0b0	0b0	0b0

Downloaded from Arrow.com.

Access Type	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	CS_ACTUAL[7:0]							
Reset	0b00000							
Access Type	Read Only							
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	–	stealth	s2vsb	s2vsa	ov	seq_stopped	SG_RESULT[9:8]	
Reset	–	0b0	0b0	0b0			0b0000000000	
Access Type	–	Read Only	Read Only	Read Only	Read Only	Read Only	Read Only	
BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SG_RESULT[7:0]							
Reset	0b0000000000							
Access Type	Read Only							

BITFIELD	BITS	DESCRIPTION	DECODE
stst	31	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.	0x0: Motor moving. 0x1: Motor in standstill.
olb	30	Open load indicator phase B.	0x0: Normal operation 0x1: Open load detected on phase B. **Note:** This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion only.
ola	29	Open load indicator phase A.	0x0: Normal operation 0x1: Open load detected on phase A. **Hint:** This is just an informative flag. The driver takes no action upon it. False detection may occur in fast motion and standstill. Check during slow motion only.
s2gb	28	Short-to-ground indicator phase B.	0x0: Normal operation 0x1: Short-to-GND detected on phase B. The driver is disabled. The flags stay active until the driver is disabled by software ($TOFF$ = 0) or by the ENN input.

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
s2ga	27	Short-to-ground indicator phase A.	0x0: Normal operation 0x1: Short-to-GND detected on phase A. The driver is disabled. The flags stay active until the driver is disabled by software ($TOFF$ = 0) or by the ENN input.
otpw	26	Overtemperature prewarning flag.	0x0: Normal operation 0x1: Overtemperature prewarning threshold is exceeded. The overtemperature prewarning flag is common for both bridges.
ot	25	Overtemperature flag.	0x0: Normal operation 0x1: Overtemperature limit is reached. Drivers are disabled until the IC has cooled down. The overtemperature flag is common for both bridges.
stallguard	24	StallGuard2/StallGuard+ status.	0x0: Normal operation 0x1: Motor stall detected by StallGuard2 (in SpreadCycle operation), respectively, by StallGuard+ (in StealthChop+ operation).
CS_ACTUAL	23:16	Actual motor target current (actual CoolStep/CoolStep+ current of CoolStep function is enabled). Use for monitoring CoolStep and *IRUN* and *IHOLD* current scaling.	
stealth	14	StealthChop+ indicator.	0x0: SpreadCycle mode 0x1: StealthChop+ mode
s2vsb	13	Short-to-supply indicator phase B.	0x0: No error 0x1: Short-to-supply detected on phase B. The driver is disabled. The flags stay active until the driver is disabled by software ($TOFF$ = 0) or by the ENN input.
s2vsa	12	Short-to-supply indicator phase A.	0x0: No error 0x1: Short-to-supply detected on phase A. The driver is disabled. The flags stay active until the driver is disabled by software ($TOFF$ = 0) or by the ENN input.
ov	11	Overvoltage flag indicator. Flag is set if *ADC_VSUPPLY* >= *OVERVOLTAGE_VTH.*	0x0: Threshold not yet reached. 0x1: Threshold reached.
seq_stopped	10	1: sequencer is stopped by ENC_A.	0x0: Sequencer running. 0x1: Sequencer stopped.
SG_RESULT	9:0	StallGuard2 result or StallGuard+ result (depending on chopper mode) or PWM on-time for coil A in standstill with SpreadCycle chopper for motor temperature estimation. Mechanical load measurement for SpreadCycle: StallGuard2 can measure mechanical motor load. A higher value means lower mechanical load. For StallGuard2, a value of 0 signals highest load. With optimum *SGT* setting, this is an indicator for a motor stall. The stall detection compares *SG_RESULT* to 0 to detect a stall. *SG_RESULT* is used as a base for CoolStep operation by comparing it to a programmable upper limit and a lower limit. StallGuard2 works best with fine microstep operation.	

Downloaded from Arrow.com.

BITFIELD	BITS	DESCRIPTION	DECODE
		In standstill, no StallGuard2 result can be obtained. *SG_RESULT* shows the chopper on-time for motor coil A instead. In combination with a sufficiently high-resistive motor, this measurement can be used to estimate motor coil temperature: move the motor to a determined microstep position at a certain current setting to get a rough estimation of motor temperature by reading the chopper on-time. As the motor heats up, its coil resistance rises and the chopper on-time increases. A more precise measurement of coil resistance, and with this estimation of motor temperature, is available with StalthChop+ In StallGuard+ (indicated by flag *stealth*), *SGP_RESULT* is mirrored to *SG_RESULT*. In this mode, the range is 10-bit signed, rather than unsigned for StallGuard2. For StallGuard+ specifics, see *SGP_RESULT*.	

PWMCONF (0x70)

BIT	31	30	29	28	27	26	25	24
Field	–	–	–	–	–	–	–	–
Reset	–	–	–	–	–	–	–	–
Access Type	–	–	–	–	–	–	–	–
BIT	**23**	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Field	–	–	–	–	SD_ON_MEAS_HI[3:0]			
Reset	–	–	–	–	0x0F			
Access Type	–	–	–	–	Write, Read			
BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Field	SD_ON_MEAS_LO[3:0]				–	–	–	–
Reset	0x0E				–	–	–	–
Access Type	Write, Read				–	–	–	–

Downloaded from Arrow.com.

BIT	7	6	5	4	3	2	1	0
Field	OL_THRSH[1:0]		FREEWHEEL[1:0]		PWM_FREQ[3:0]			
Reset	0x0		0b00		0x00			
Access Type	Write, Read		Write, Read		Write, Read			

BITFIELD	BITS	DESCRIPTION	DECODE
SD_ON_MEAS_HI	19:16	Higher hysteresis duty cycle threshold to switch ADC sample point from measurement during slow decay time to measurement during ON-time. *SD_ON_MEAS_HI > SD_ON_MEAS_LO*	
SD_ON_MEAS_LO	15:12	Lower hysteresis duty cycle threshold to switch ADC sample point from measurement during ON time to measurement during slow-decay time. *SD_ON_MEAS_HI > SD_ON_MEAS_LO*	
OL_THRSH	7:6	Open load sensititvity for StealthChop+ operation (during motor rotation). Selects the minimum current value to be reached minimum one time within each half electrical rotation to avoid detection of open load.	0x0: 0.125 x target current 0x1: 0.25 x target current 0x2: 0.5 x target current 0x3: 0.75 x target current
FREEWHEEL	5:4	Selects StealthChop+ behaviour during standstill when motor current setting is zero (I_HOLD = 0).	0x0: Normal operation 0x1: Freewheeling 0x2: Passive braking using LS drivers. 0x3: Passive braking using HS drivers.
PWM_FREQ	3:0	PWM frequency selection	0x0: 19.5kHz 0x1: 24.4kHz 0x2: 29.3kHz 0x3: 34.2kHz 0x4: 39.1kHz 0x5: 44.0kHz 0x6: 48.8kHz 0x7: 53.7kHz 0x8: 58.6kHz 0x9: Reserved (58.6kHz) 0xA: Reserved (58.6kHz) 0xB: Reserved (58.6kHz) 0xC: Reserved (58.6kHz) 0xD: Reserved (58.6kHz) 0xE: Reserved (58.6kHz) 0xF: Reserved (58.6kHz)

Downloaded from Arrow.com.

Ordering Information

PART NUMBER	TEMPERATURE RANGE	PIN-PACKAGE
TMC5262AFV+	-40°C to +125°C	30L FC2QFN - 6mm x 6mm
TMC5262AFV+T	-40°C to +125°C	30L FC2QFN - 6mm x 6mm

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

T Denotes tape-and-reel.

Downloaded from Arrow.com.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	09/25	Initial release	—

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.

Downloaded from Arrow.com.