# TMC5031 DATASHEET

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


## APPLICATIONS

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

## FEATURES AND BENEFITS

**2-phase** stepper motors

**Drive Capability** up to 2 x 1.1A coil current

**Motion Controller** with **sixPoint™** ramp

**Voltage Range** 4.75… 16V DC

**SPI Interface**

**2x Ref.-Switch input per axis**

**Highest Resolution** 256 microsteps per full step

**Full Protection & Diagnostics**

**stallGuard2™** high precision sensorless motor load detection

**coolStep™** load dependent current control for energy savings up to 75%

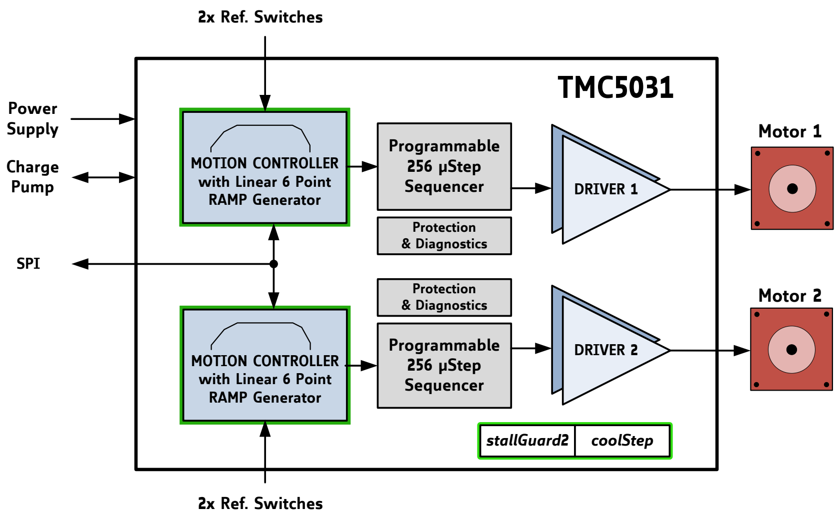
**spreadCycle™** high-precision chopper for best current sine wave form and zero crossing with additional **chopSync2™**

**Compact Size** 7x7mm QFN48 package

## DESCRIPTION

The TMC5031 is a low cost motion controller and driver IC for up to two stepper motors. It combines two flexible ramp motion controllers with energy efficient stepper motor drivers. The drivers support two-phase stepper motors and offer an industry-leading feature set, including high-resolution microstepping, sensorless mechanical load measurement, load-adaptive power optimization, and low-resonance chopper operation. All features are controlled by a standard SPI™ interface. Integrated protection and diagnostic features support robust and reliable operation. High integration, high energy efficiency and small form factor enable miniaturized designs with low external component count for cost-effective and highly competitive solutions.

## BLOCK DIAGRAM

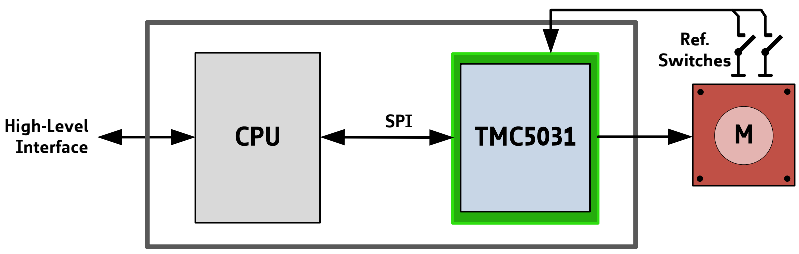



TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

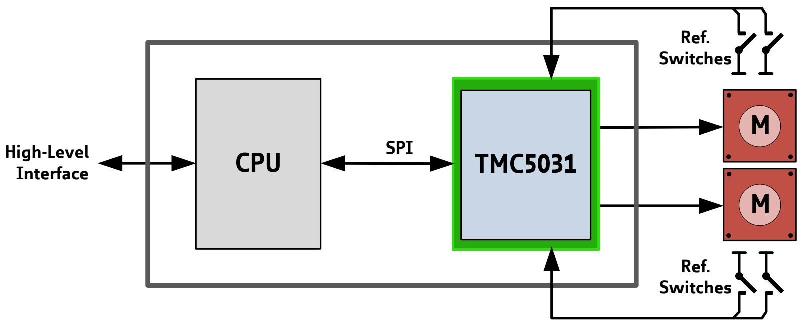
# APPLICATION EXAMPLES: HIGH FLEXIBILITY – MULTIPURPOSE USE

The TMC5031 scores with power density, complete motion controlling features and integrated power stages. It offers a versatility that covers a wide spectrum of applications from battery systems up to embedded applications with 1.1A current per motor. The small form factor keeps costs down and allows for miniaturized layouts. Extensive support at the chip, board, and software levels enables rapid design cycles and fast time-to-market with competitive products. High energy efficiency and reliability from TRINAMIC's coolStep technology deliver cost savings in related systems such as power supplies and cooling.

### Miniaturized Design for up to two Stepper Motors




Two reference switch inputs can be used for each motor. A single CPU controls the whole system, which is highly economical and space saving.




## Order Codes

| Order code | Description | Size |
|---|---|---|
| TMC5031-LA | Dual stallGuard2™ and coolStep™ controller/driver, QFN48 | 7 x 7 $mm^2$ |

# Table of Contents

# 1 Principles of Operation

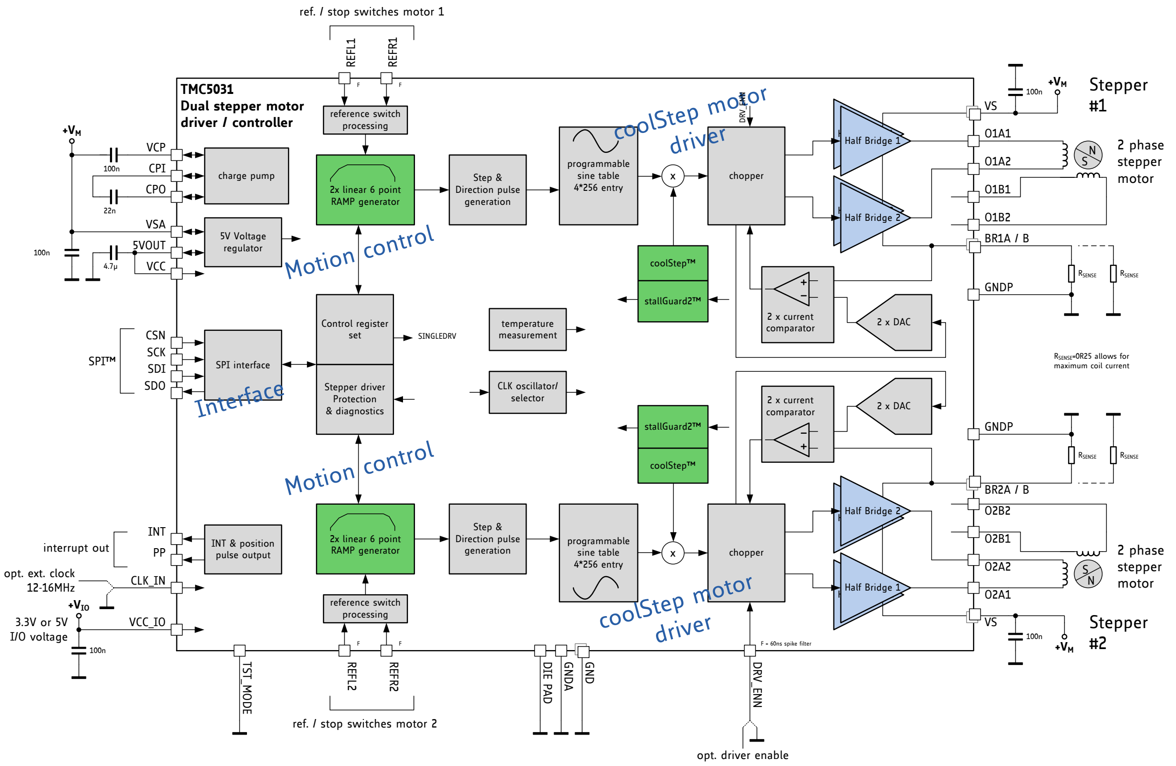



**Figure 1.1 Basic application and block diagram**

The TMC5031 motion controller and driver chip is an intelligent power component interfacing between the CPU and up to two stepper motors. The TMC5031 offers a number of unique enhancements which are enabled by the system-on-chip integration of driver and controller. The sixPoint ramp generator of the TMC5031 uses coolStep and stallGuard2 automatically to optimize every motor movement: TRINAMICs special features contribute toward lower system cost, greater precision, greater energy efficiency, smoother motion, and cooler operation in stepper motor applications. The clear concept and the comprehensive solution save design-in time.

## 1.1 Key Concepts

The TMC5031 implements several advanced features which are exclusive to TRINAMIC products. These features contribute toward greater precision, greater energy efficiency, higher reliability, smoother motion, and cooler operation in many stepper motor applications.

| | |
|---|---|
| ***stallGuard2™*** | High-precision load measurement using the back EMF on the motor coils. |
| ***coolStep™*** | Load-adaptive current control which reduces energy consumption by as much as 75%. |
| ***spreadCycle™*** | High-precision chopper algorithm available as an alternative to the traditional constant off-time algorithm. |

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

## 1.2 SPI Control Interface

The SPI interface is a bit-serial interface synchronous to a bus clock. For every bit sent from the bus master to the bus slave, another bit is sent simultaneously from the slave to the master. Communication between an SPI master and the TMC5031 slave always consists of sending one 40-bit command word and receiving one 40-bit status word.

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

## 1.3 Software

From a software point of view the TMC5031 is a peripheral with a number of control and status registers. Most of them can either be written only or read only, some of the registers allow both read and write access. In case read-modify-write access is desired for a write only register, a shadow register can be realized in master software.

## 1.4 Moving and Controlling the Motor

### 1.4.1 Integrated Motion Controller

The integrated 32 bit motion controller automatically drives the motors to target positions, or accelerates to target velocities. All motion parameters can be changed on the fly with the motion controller recalculating immediately. A minimum set of configuration data consists of acceleration and deceleration values and the maximum motion velocity. A start and stop velocity is supported as well as a second acceleration and deceleration setting. It supports immediate reaction to mechanical reference switches and to the sensorless stall detection stallGuard2.

***Benefits are:***

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

## 1.5 Precision Driver with Programmable Microstepping Wave

Current into the motor coils is controlled using a cycle-by-cycle chopper mode. Two chopper modes are available: a traditional constant off-time mode and the new spreadCycle mode. Constant off-time mode provides higher torque at the highest velocity, while spreadCycle mode offers smoother operation and greater power efficiency over a wide range of speed and load. The spreadCycle chopper scheme automatically integrates a fast decay cycle and guarantees smooth zero crossing performance. Programmable microstep shapes allow optimizing the motor performance.

***Benefits are:***

- Significantly improved microstepping with low cost motors
- Motor runs smooth and quiet
- Reduced mechanical resonances yields improved torque

## 1.6 stallGuard2 – Mechanical Load Sensing

stallGuard2 provides an accurate measurement of the load on the motor. It can be used for stall detection as well as other uses at loads below those which stall the motor, such as coolStep load-adaptive current reduction. This gives more information on the drive allowing functions like sensorless homing and diagnostics of the drive mechanics.

## 1.7 coolStep – Load Adaptive Current Control

coolStep drives the motor at the optimum current. It uses the stallGuard2 load measurement information to adjust the motor current to the minimum amount required in the actual load situation. This saves energy and keeps the components cool, making the drive an efficient and precise solution.

| | | |
|---|---|---|
| *Energy efficiency* | - | power consumption decreased up to 75%. |
| *Motor generates less heat* | - | improved mechanical precision. |
| *Less or no cooling* | - | improved reliability and lower cost infrastructure. |
| *Use of smaller motor* | - | less torque reserve required, lower cost motor. |

Figure 1.2 shows the efficiency gain of a 42mm stepper motor when using coolStep compared to standard operation with 50% of torque reserve. coolStep is enabled above 60RPM in the example.

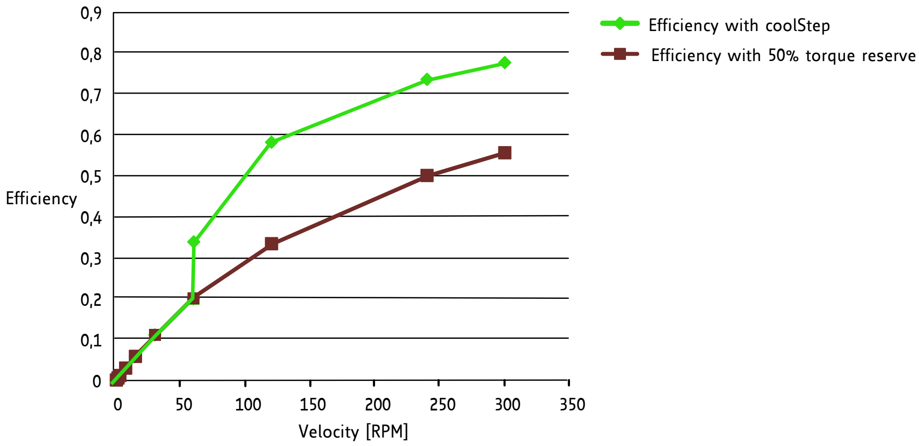



**Figure 1.2 Energy efficiency with coolStep (example)**

# 2 Pin Assignments

## 2.1 Package Outline

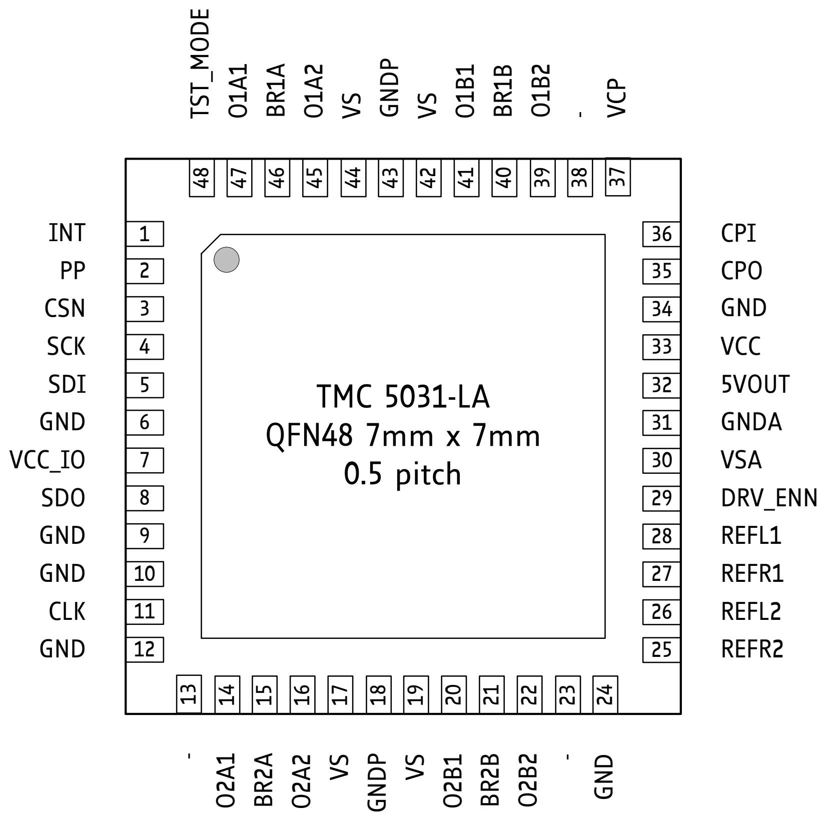



**Figure 2.1 TMC5031 pin assignments.**

## 2.2 Signal Descriptions

| Pin | Number | Type | Function |
|---|---|---|---|
| GND | 6, 9, 10, 12, 24, 34 | GND | Digital ground pin for IO pins and digital circuitry. |
| VCC_IO | 7 | | 3.3V or 5V I/O supply voltage pin for all digital inputs and outputs. May be supplied from 5VOUT pin in stand-alone operation, where no I/O voltage supply is available. |
| VSA | 30 | | Analog high voltage supply for linear regulator and internal references – typically supplied with driver supply voltage. Provide 100nF blocking capacitor to GND. Avoid excessive voltage ripple. |
| GNDA | 31 | GND | Analog GND |
| 5VOUT | 32 | | Output of internal 5V linear regulator, supply voltage for internal analog circuitry and reference for coil current regulators. An external capacitor to GNDA close to the pin is required. 4.7µF ceramic are recommended to keep ripple below a few mV, especially when used to supply VCC. Optional RC filtering can be used to decouple VCC ripple from this pin (3.3 Ω recommended). |
| VCC | 33 | | Digital core power supply. Normally supplied by 5VOUT pin. In case, a different 5V supply is used, or RC-filtering is applied, provide a 470nF or larger blocking capacitor to GND. |

| Pin | Number | Type | Function |
|---|---|---|---|
| DIE_PAD | - | GND | The exposed die attach pad is the thermal cooling pad for the IC and shall be soldered to a ground pad, and be directly electrically tied together with all GND pins. Use a large number of thermally conducting vias to a PCB ground plane for best thermal and electrical performance. The ground plane also acts as a heat spreader to reduce thermal junction to ambient resistance. |

**Table 2.1 Low voltage digital and analog power supply pins**

| Pin | Number | Type | Function |
|---|---|---|---|
| CPO | 35 | O(VCC) | Charge pump driver output. Outputs 5V (GND to VCC) square wave with 1/16 of internal oscillator frequency. |
| CPI | 36 | I(VCP) | Charge pump capacitor input: Provide external 22 nF / 50 V capacitor to CPO. |
| VCP | 37 | | Output of charge pump. Provide external 100 nF capacitor to VS. |

**Table 2.2 Charge pump pins**

| Pin | Number | Type | Function |
|---|---|---|---|
| INT | 1 | O (Z) | Tristate interrupt output based on ramp generator flags 4, 5, 6 & 7. |
| PP | 2 | O (Z) | Tristate position compare output for motor 1 (*poscmp_enable*=1). |
| CSN | 3 | I | Chip select input of SPI interface |
| SCK | 4 | I | Serial clock input of SPI interface |
| SDI | 5 | I | Data input of SPI interface |
| SDO | 8 | O (Z) | Tristate data output of SPI interface (enabled with CSN=0) |
| CLK | 11 | I | Clock input for all internal operations. Tie low to use internal oscillator. A high signal disables the internal oscillator until power down. |
| REFR2 | 25 | I | Right reference switch input for motor 2 |
| REFL2 | 26 | I | Left reference switch input for motor 2 |
| REFR1 | 27 | I | Right reference switch input for motor 1 |
| REFL1 | 28 | I | Left reference switch input for motor 1 |
| DRV_ENN | 29 | I | Enable (not) input for drivers (tie to GND). Switches off all motor outputs (set high for disable). |
| TST_MODE | 48 | I | Test mode input. Puts IC into test mode. Tie to GND for normal operation. |
| - | 13, 23, 38 | N.C. | Unused pins – no internal electrical connection. Leave open or tie to GND for compatibility with future devices. |

**Table 2.3 Digital I/O pins (all related to VCC_IO supply)**

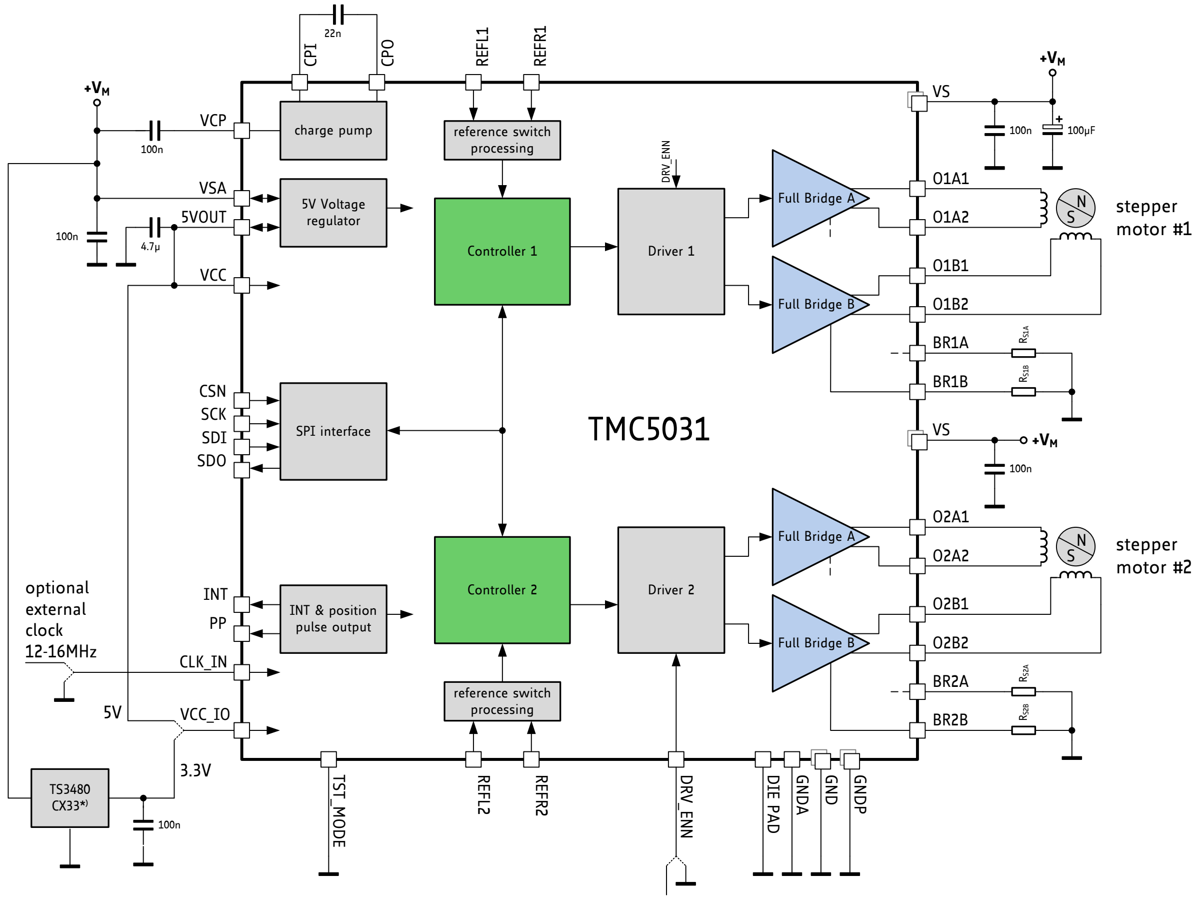
| Pin | Number | Type | Function |
|---|---|---|---|
| O2A1 | 14 | O (VS) | Motor 2 A1 output (stepper motor coil A) |
| BR2A | 15 | | Motor 2 bridge A negative power supply and current sense input. Provide external sense resistor to GND. |
| O2A2 | 16 | O (VS) | Motor 2 A2 output (stepper motor coil A) |
| VS | 17, 19 | | Driver 2 positive power supply. Connect to VS and provide sufficient filtering capacity for chopper current ripple. |
| GNDP | 18 | GND | Power GND for driver 2. Connect to GND. |
| O2B1 | 20 | O (VS) | Motor 2 B1 output (stepper motor coil B) |
| BR2B | 21 | | Motor 2 bridge B negative power supply and current sense input. Provide external sense resistor to GND. |
| O2B2 | 22 | O (VS) | Motor 2 B2 output (stepper motor coil B) |
| O1B2 | 39 | O (VS) | Motor 1 B2 output (stepper motor coil B) |
| BR1B | 40 | | Motor 1 bridge B negative power supply and current sense input. Provide external sense resistor to GND. |
| O1B1 | 41 | O (VS) | Motor 1 B1 output (stepper motor coil B) |
| VS | 42, 44 | | Driver 1 positive power supply. Connect to VS and provide sufficient filtering capacity for chopper current ripple. |
| GNDP | 43 | GND | Power GND for driver 1. Connect to GND. |
| O1A2 | 45 | O (VS) | Motor 1 A2 output (stepper motor coil A) |
| BR1A | 46 | | Motor 1 bridge A negative power supply and current sense input. Provide external sense resistor to GND. |
| O1A1 | 47 | O (VS) | Motor 1 A1 output (stepper motor coil A) |

**Table 2.4 Power driver pins**

# 3 Sample Circuits

The sample circuits show the connection of the external components in different operation and supply modes. The standard application circuit uses a minimum set of additional components in order to operate the motor. The connection of the bus interface and further digital signals is left out for clarity.

## 3.1 Standard Application Circuit




**) For a reliable start-up it is essential that VCC_IO comes up to a minimum of 1.5V before the TMC5031 leaves the reset condition. Therefore, TRINAMIC recommends using a fast-start-up voltage regulator (e.g. TS3480CX33) in a 3.3V environment.*

**Figure 3.1 Standard application circuit**

In order to minimize linear voltage regulator power dissipation of the internal 5V voltage regulator in an application where VM is high, a different (lower) supply voltage can be used for VSA, if available.

### 3.1.1 VCC_IO Requirements

For a reliable start-up it is essential that VCC_IO comes up to a minimum of 1.5V before the TMC5031 leaves the reset condition. The reset condition ends earliest 50µs after the time when VSA exceeds its undervoltage threshold of typically 4.2V, or when 5VOUT exceeds its undervoltage threshold of typically 3.5V, whichever comes last.

#### THERE ARE THREE WAYS TO COME UP TO VCC_IO REQUIREMENTS

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

- An external low drop regulator can be used in a 3.3V environment as shown in Figure 3.1. Note, that most voltage regulators are not suitable for this application because they show a delayed boot up. The following external regulators are proved by TRINAMIC:

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

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

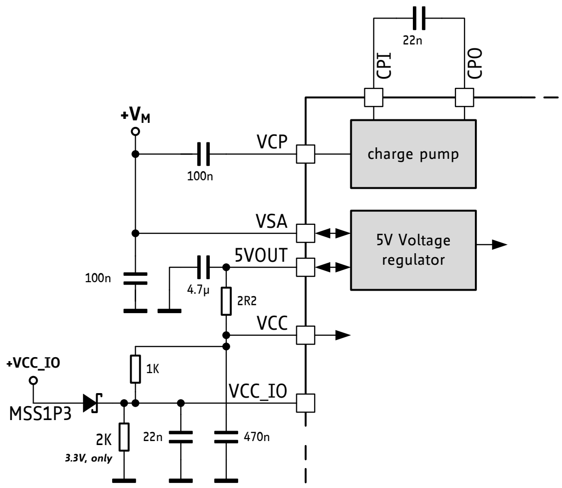



**Figure 3.2 External supply of VCC_IO (showing optional filtering for VCC)**

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

## 3.2 5 V Only Supply

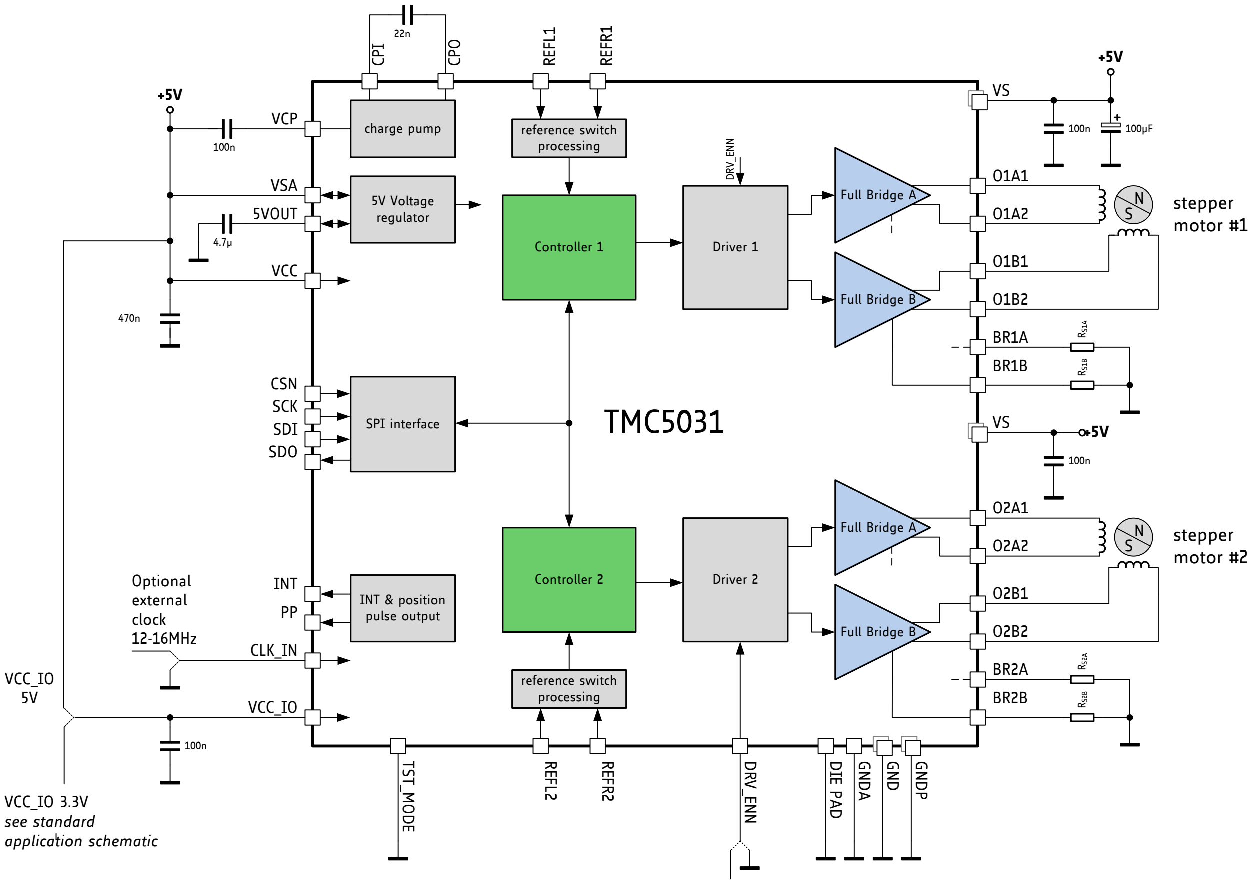



**Figure 3.3 5V only operation**

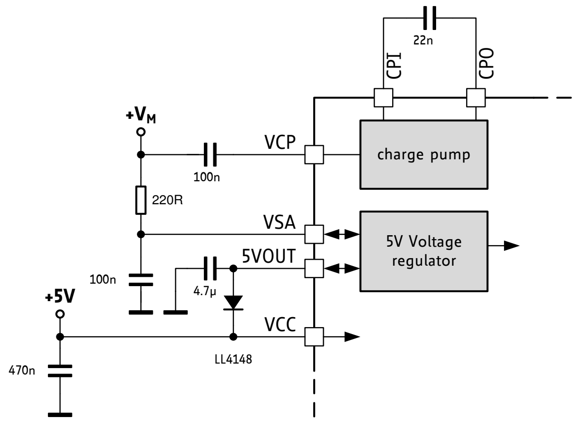
While the standard application circuit is limited to roughly 5.5 V lower supply voltage, a 5 V only application lets the IC run from a normal 5 V +/-10% supply. In this application, linear regulator drop must be minimized. Therefore, the major 5 V load is removed by supplying VCC directly from the external supply.

In order to keep supply ripple away from the analog voltage reference, 5VOUT should have an own filtering capacity and the 5VOUT pin does not become bridged to the 5V supply.

## 3.3 External VCC Supply

Supplying VCC from an external supply is advised, when cooling of the chip is critical, e.g. at high environment temperatures in combination with high supply voltages (16V), as the linear regulator is a major source of on-chip power dissipation. It must be made sure that the external VCC supply comes up before or synchronously with the 5VOUT supply, because otherwise the power-up reset event may be missed by the TMC5031. A diode from 5VOUT to VCC ensures this, in case the external voltage regulator is not a low drop type linear regulator. In order to prevent overload of the internal 5V regulator when using this diode, an additional series resistor has been added to VSA.

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




The diode is mandatory to satisfy power-up conditions!

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

## 3.4 Optimizing Analog Precision

The 5VOUT pin is used as an analog reference for operation of the TMC5031. Performance will degrade when there is voltage ripple on this pin. Most of the high frequency ripple in a TMC5031 design results from the operation of the internal digital logic. The digital logic switches with each edge of the clock signal. Further, ripple results from operation of the charge pump, which operates with roughly 1 MHz and draws current from the VCC pin. In order to keep this ripple as low as possible, an additional filtering capacitor can be put directly next to the VCC pin with vias to the GND plane giving a short connection to the digital GND pins (pin 6 and pin 34). Analog performance is best, when this ripple is kept away from the analog supply pin 5VOUT, using an additional series resistor of 2.2 Ω to 3.3 Ω. The voltage drop on this resistor will be roughly 100 mV ($I_{VCC}$ * R).

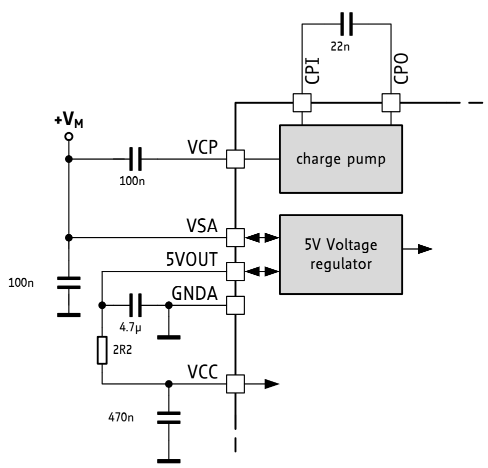



**Figure 3.5 Adding an RC-Filter on VCC for reduced ripple**

# 4 SPI Interface

## 4.1 SPI Datagram Structure

The TMC5031 uses 40 bit SPI™ (Serial Peripheral Interface, SPI is Trademark of Motorola) datagrams for communication with a microcontroller. Microcontrollers which are equipped with hardware SPI are typically able to communicate using integer multiples of 8 bit. The NCS line of the TMC5031 must be handled in a way, that it stays active (low) for the complete duration of the datagram transmission.

Each datagram sent to the TMC5031 is composed of an address byte followed by four data bytes. This allows direct 32 bit data word communication with the register set of the TMC5031. Each register is accessed via 32 data bits even if it uses less than 32 data bits.

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

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

Most registers are write only registers, some can be read additionally, and there are also some read only registers.

<table>
<tr><th colspan="40">TMC5031 SPI DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="40">MSB (transmitted first) 40 bit LSB (transmitted last)</td></tr>
<tr><td colspan="40">39 ... ... 0</td></tr>
<tr><td colspan="8">→ 8 bit address<br>← 8 bit SPI status</td><td colspan="32">←→ 32 bit data</td></tr>
<tr><td colspan="8">39 ... 32</td><td colspan="32">31 ... 0</td></tr>
<tr><td colspan="8">→ to TMC5031:<br>RW + 7 bit address<br>← from TMC5031:<br>8 bit SPI status</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td><td colspan="8">8 bit data</td></tr>
<tr><td colspan="8">39 / 38 ... 32</td><td colspan="8">31 ... 24</td><td colspan="8">23 ... 16</td><td colspan="8">15 ... 8</td><td colspan="8">7 ... 0</td></tr>
<tr><td>W</td><td colspan="7">38...32</td><td colspan="4">31...28</td><td colspan="4">27...24</td><td colspan="4">23...20</td><td colspan="4">19...16</td><td colspan="4">15...12</td><td colspan="4">11...8</td><td colspan="4">7...4</td><td colspan="4">3...0</td></tr>
<tr><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td><td>32</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

### 4.1.1 Selection of Write / Read (WRITE_notREAD)

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

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

Whenever data is read from or written to the TMC5031, the MSBs delivered back contain the SPI status, *SPI_STATUS*, a number of eight selected status bits.

*Example*:

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

| action | data sent to TMC5031 | data received from TMC5031 |
|---|---|---|
| read *X_ACTUAL* | → 0x2100000000 | ← 0xSS & unused data |
| read *X_ACTUAL* | → 0x2100000000 | ← 0xSS & X_ACTUAL |
| write *V_ACTUAL*:= 0x00ABCDEF | → 0xA200ABCDEF | ← 0xSS & X_ACTUAL |
| write *V_ACTUAL*:= 0x00123456 | → 0xA200123456 | ← 0xSS00ABCDEF |

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

### 4.1.2 SPI Status Bits Transferred with Each Datagram Read Back

| ***SPI_STATUS*** **– status flags transmitted with each SPI access in bits 39 to 32** | | |
|---|---|---|
| **Bit** | **Name** | **Comment** |
| 7 | - | reserved (0) |
| 6 | *status_stop_l(2)* | *RAMP_STATUS2*[0] – 1: Signals motor 2 stop left switch status |
| 5 | *status_stop_l(1)* | *RAMP_STATUS1*[0] – 1: Signals motor 1 stop left switch status |
| 4 | *velocity_reached(2)* | *RAMP_STATUS2*[8] – 1: Signals motor 2 has reached its target velocity |
| 3 | *velocity_reached(1)* | *RAMP_STATUS1*[8] – 1: Signals motor 1 has reached its target velocity |
| 2 | *driver_error(2)* | *GSTAT*[2] – 1: Signals driver 2 driver error (clear by reading *GSTAT*) |
| 1 | *driver_error(1)* | *GSTAT*[1] – 1: Signals driver 1 driver error (clear by reading *GSTAT*) |
| 0 | *reset_flag* | *GSTAT*[0] – 1: Signals, that a reset has occurred (clear by reading *GSTAT*) |

### 4.1.3 Data Alignment

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

## 4.2 SPI Signals

The SPI bus on the TMC5031 has four signals:

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

The slave is enabled for an SPI transaction by a low on the chip select input CSN. Bit transfer is synchronous to the bus clock SCK, with the slave latching the data from SDI on the rising edge of SCK and driving data to SDO following the falling edge. The most significant bit is sent first. A minimum of 40 SCK clock cycles is required for a bus transaction with the TMC5031.

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

CSN must be low during the whole bus transaction. When CSN goes high, the contents of the internal shift register are latched into the internal control register and recognized as a command from the master to the slave. If more than 40 bits are sent, only the last 40 bits received before the rising edge of CSN are recognized as the command.

## 4.3 Timing

The SPI interface is synchronized to the internal system clock, which limits the SPI bus clock SCK to half of the system clock frequency. If the system clock is based on the on-chip oscillator, an additional 10% safety margin must be used to ensure reliable data transmission. All SPI inputs as well as the ENN input are internally filtered to avoid triggering on pulses shorter than 20ns. Figure 4.1 shows the timing parameters of an SPI bus transaction, and the table below specifies their values.

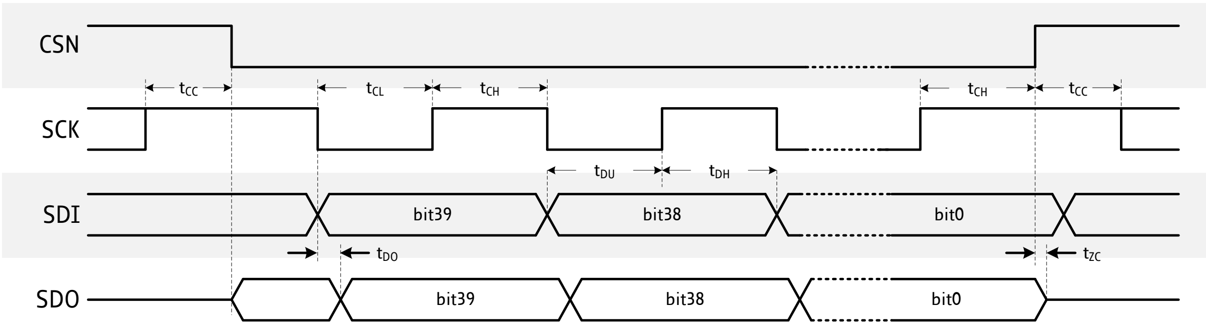



**Figure 4.1 SPI timing**

| **SPI interface timing** | **AC-Characteristics**<br>clock period: $t_{CLK}$ | | | | | |
|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Conditions** | **Min** | **Typ** | **Max** | **Unit** |
| SCK valid before or after change of CSN | $t_{CC}$ | | 10 | | | ns |
| CSN high time | $t_{CSH}$ | *) Min time is for synchronous CLK with SCK high one $t_{CH}$ before CSN high only | $t_{CLK}$ *) | $>2t_{CLK}+10$ | | ns |
| SCK low time | $t_{CL}$ | *) Min time is for synchronous CLK only | $t_{CLK}$ *) | $>t_{CLK}+10$ | | ns |
| SCK high time | $t_{CH}$ | *) Min time is for synchronous CLK only | $t_{CLK}$ *) | $>t_{CLK}+10$ | | ns |
| SCK frequency using internal clock | $f_{SCK}$ | assumes minimum OSC frequency | | | 4 | MHz |
| SCK frequency using external 16MHz clock | $f_{SCK}$ | assumes synchronous CLK | | | 8 | MHz |
| SDI setup time before rising edge of SCK | $t_{DU}$ | | 10 | | | ns |
| SDI hold time after rising edge of SCK | $t_{DH}$ | | 10 | | | ns |
| Data out valid time after falling SCK clock edge | $t_{DO}$ | no capacitive load on SDO | | | $t_{FILT}+5$ | ns |
| SDI, SCK and CSN filter delay time | $t_{FILT}$ | rising and falling edge | 12 | 20 | 30 | ns |

# 5 Register Mapping

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

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

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

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

**OVERVIEW REGISTER MAPPING**

| REGISTER | DESCRIPTION |
|---|---|
| General Configuration Registers | These registers contain<br>- global configuration,<br>- global status flags,<br>- slave address configuration. |
| Ramp Generator Motion Control Register Set | This register set offers registers for<br>- choosing a ramp mode,<br>- choosing velocities,<br>- homing,<br>- acceleration and deceleration, and<br>- target positioning. |
| Ramp Generator Driver Feature Control Register Set | This register set offers registers for<br>- driver current control,<br>- setting thresholds for coolStep operation,<br>- setting thresholds for different chopper modes, and<br>- a reference switch and stallGuard2 event configuration register and (*with separate table*)<br>- a ramp and reference switch status register (*with separate table*). |
| Motor Driver Register Set | This register set offers registers for<br>- setting / reading out microstep table and counter (*see separate table, too*),<br>- chopper and driver configuration (*see separate tables for different motor types, too*),<br>- coolStep and stallGuard2 configuration (*see separate table, too*), and<br>- reading out stallGuard2 values and driver error flags (*see separate table, too*). |

## 5.1 General Configuration Registers

**GENERAL CONFIGURATION REGISTERS (0X00…0X1F)**

| R/W | Addr | n | *Register* | Bit | Description / *bit names* |
|---|---|---|---|---|---|
| RW | 0x00 | 11 | *GCONF* | **Bit** | ***GCONF* – Global configuration flags** |
| | | | | 0..2 | Reserved, set to 0 |
| | | | | 3 | *poscmp_enable*<br>0: Outputs INT and PP are tristated.<br>1: Position compare pulse (PP) and interrupt output (INT) are available<br><br>*Attention – do not leave the ouputs floating in tristate condition, provide an external pull-up or set this bit 1.* |
| | | | | 4..6 | Reserved, set to 0 |
| | | | | 7 | *test_mode*<br>0: Normal operation<br>1: Enable analog test output on pin REFR2 *TEST_SEL* selects the function of REFR2: 0…4: T120, DAC1, VDDH1, DAC2, VDDH2<br><br>*Attention: Not for user, set to 0 for normal operation!* |
| | | | | 8 | *shaft1*<br>1: Inverse motor 1 direction |
| | | | | 9 | *shaft2*<br>1: Inverse motor 2 direction |
| | | | | 10 | *lock_gconf*<br>1: GCONF is locked against further write access. |
| R+C | 0x01 | 4 | *GSTAT* | **Bit** | ***GSTAT* – Global status flags** |
| | | | | 0 | *reset*<br>1: Indicates that the IC has been reset since the last read access to *GSTAT.* |
| | | | | 1 | *drv_err1*<br>1: Indicates, that driver 1 has been shut down due to an error since the last read access. |
| | | | | 2 | *drv_err2*<br>*1:* Indicates, that driver 2 has been shut down due to an error since the last read access. |
| | | | | 3 | *uv_cp*<br>1: Indicates an undervoltage on the charge pump. The driver is disabled in this case. |
| W | 0x03 | 4 | *TEST_SEL* | **Bit** | ***SLAVECONF*** |
| | | | | 3..0 | *TEST_SEL*:<br>selects the function of REFR2 in test mode:<br>0…4: T120, DAC1, VDDH1, DAC2, VDDH2<br><br>*Attention: Not for user, set to 0 for normal operation!* |
| R | 0x04 | 8 + 8 | *INPUT* | **Bit** | ***INPUT*** |
| | | | | 0..6 | Unused, ignore these bits |
| | | | | 7 | Reads the state of the DRV_ENN pin |
| | | | | 31.. 24 | *VERSION*: 0x01=first version of the IC |
| W | 0x05 | 32 | *X_COMPARE* | | Position comparison register for motor 1 position strobe. Activate *poscmp_enable* to get position pulse on output PP.<br><br>*XACTUAL = X_COMPARE*:<br>- Output PP becomes high. It returns to a low state, if the positions mismatch. |

## 5.2 Ramp Generator Registers

*Addresses **Addr** are specified for motor 1 (upper value) and motor 2 (second address).*

### 5.2.1 Ramp Generator Motion Control Register Set

| RAMP GENERATOR MOTION CONTROL REGISTER SET (MOTOR 1: 0X20…0X2D, MOTOR 2: 0X40…0X4D) | | | | | |
|---|---|---|---|---|---|
| **R/W** | **Addr** | **n** | ***Register*** | **Description / *bit names*** | **Range [Unit]** |
| RW | 0x20<br>0x40 | 2 | *RAMPMODE* | *RAMPMODE:*<br>0: Positioning mode (using all A, D and V parameters)<br>1: Velocity mode to positive *VMAX* (using *AMAX* acceleration)<br>2: Velocity mode to negative *VMAX* (using *AMAX* acceleration)<br>3: Hold mode (velocity remains unchanged, unless stop event occurs) | 0…3 |
| RW | 0x21<br>0x41 | 32 | *XACTUAL* | Actual motor position (signed)<br><br>*Hint:* This value normally should only be modified, when homing the drive. In positioning mode, modifying the register content will start a motion. | -2^31…<br>+(2^31)-1 |
| R | 0x22<br>0x42 | 24 | *VACTUAL* | Actual motor velocity from ramp generator (signed) | +-(2^23)-1<br>[µsteps / t] |
| W | 0x23<br>0x43 | 18 | *VSTART* | Motor start velocity (unsigned)<br><br>Set *VSTOP ≥ VSTART!* | 0…(2^18)-1<br>[µsteps / t] |
| W | 0x24<br>0x44 | 16 | *A1* | First acceleration between *VSTART* and *V1* (unsigned) | 0…(2^16)-1<br>[µsteps / ta²] |
| W | 0x25<br>0x45 | 20 | *V1* | First acceleration / deceleration phase target velocity (unsigned)<br><br>0: Disables *A1* and *D1* phase, use *AMAX*, *VMAX* only | 0…(2^20)-1<br>[µsteps / t] |
| W | 0x26<br>0x46 | 16 | *AMAX* | Second acceleration between V1 and VMAX (unsigned)<br><br>This is the acceleration and deceleration value for velocity mode. | 0…(2^16)-1<br>[µsteps / ta²] |
| W | 0x27<br>0x47 | 23 | *VMAX* | Second acceleration phase target velocity *VMAX* > *V1*, *VMAX* > *VSTART* (unsigned)<br><br>This is the target velocity in velocity mode. It can be changed any time during a motion. | 0…(2^23)-512<br>[µsteps / t] |
| W | 0x28<br>0x48 | 16 | *DMAX* | Deceleration between *VMAX* and *V1* (unsigned) | 0…(2^16)-1<br>[µsteps / ta²] |
| W | 0x2A<br>0x4A | 16 | *D1* | Deceleration between *V1* and *VSTOP* (unsigned)<br><br>*Attention: Do not set 0 in positioning mode, even if V1=0!* | 1…(2^16)-1<br>[µsteps / ta²] |
| W | 0x2B<br>0x4B | 18 | *VSTOP* | Motor stop velocity (unsigned)<br><br>*Attention: Set VSTOP ≥ VSTART!*<br><br>*Attention: Do not set 0 in positioning mode!* | 1…(2^18)-1<br>[µsteps / t] |

| RAMP GENERATOR MOTION CONTROL REGISTER SET (MOTOR 1: 0X20…0X2D, MOTOR 2: 0X40…0X4D) | | | | | |
|---|---|---|---|---|---|
| **R/W** | **Addr** | **n** | ***Register*** | **Description / *bit names*** | **Range [Unit]** |
| W | 0x2C<br>0x4C | 16 | *TZEROWAIT* | Waiting time after ramping down to zero velocity before next movement or direction inversion can start and before motor power down starts. Time range is about 0 to 2 seconds.<br><br>This setting avoids excess acceleration e.g. from *VSTOP* to -*VSTART*. | 0…(2^16)-1 * 512 $t_{CLK}$ |
| RW | 0x2D<br>0x4D | 32 | *XTARGET* | Target position for ramp mode (signed). Write a new target position to this register in order to activate the ramp generator positioning in *RAMPMODE*=0. Initialize all velocity, acceleration and deceleration parameters before.<br><br>*Hint:* The position is allowed to wrap around, thus, *XTARGET* value optionally can be treated as an unsigned number.<br><br>*Hint:* The maximum possible displacement is +/-((2^31)-1).<br><br>*Hint:* When increasing *V1, D1 or DMAX during* a motion, rewrite *XTARGET* afterwards in order to trigger a second acceleration phase, if desired. | -2^31…<br>+(2^31)-1 |

## 5.2.2 Ramp Generator Driver Feature Control Register Set

| RAMP GENERATOR DRIVER FEATURE CONTROL REGISTER SET (MOTOR 1: 0X30…0X36, MOTOR 2: 0X50…0X56) | | | | |
|---|---|---|---|---|
| **R/W** | **Addr** | **n** | ***Register*** | **Description / *bit names*** |
| W | 0x30<br>0x50 | 5<br>+<br>5<br>+<br>4 | *IHOLD_IRUN* | **Bit** — ***IHOLD_IRUN*** **– Driver current control**<br>**4..0** — *IHOLD*<br>Standstill current (0=1/32…31=32/32)<br>**12..8** — *IRUN*<br>Motor run current (0=1/32…31=32/32)<br><br>*Hint:* Choose sense resistors in a way, that normal *IRUN* is 16 to 31 for best microstep performance.<br>**19..16** — *IHOLDDELAY*<br>Controls the number of clock cycles for motor power down after a motion as soon as *T_ZEROWAIT* has expired. The smooth transition avoids a motor jerk upon power down.<br><br>0: instant power down<br>1..15: Delay per current reduction step in multiple of 2^18 clocks |
| W | 0x31<br>0x51 | 23 | *VCOOLTHRS* | This is the lower threshold velocity for switching on smart energy coolStep. (unsigned)<br>Set this parameter to disable coolStep at low speeds, where it cannot work reliably.<br><br>*VHIGH* ≥ \|*VACT*\| ≥ VCOOLTHRS:<br>- coolStep is enabled, if configured<br><br>(Only bits 22..8 are used for value and for comparison) |
| W | 0x32<br>0x52 | 23 | *VHIGH* | This velocity setting allows velocity dependent switching into a different chopper mode and fullstepping to maximize torque. (unsigned)<br><br>\|*VACT*\| ≥ *VHIGH*:<br>- coolStep is disabled (motor runs with normal current scale)<br>- If *vhighchm* is set, the chopper switches to *chm*=1 with *TFD*=0 (constant off time with slow decay, only).<br>- chopSync2 is switched off (*SYNC*=0)<br>- If *vhighfs* is set, the motor operates in fullstep mode.<br><br>(Only bits 22..8 are used for value and for comparison) |
| RW | 0x34<br>0x54 | 11 | ***SW_MODE*** | Switch mode configuration<br>*See separate table!* |
| R+C | 0x35<br>0x55 | 14 | ***RAMP_STAT*** | Ramp status and switch event status<br>*See separate table!* |
| R | 0x36<br>0x56 | 32 | *XLATCH* | Ramp generator latch position, latches *XACTUAL* upon a programmable switch event (see *SW_MODE*). |

*time reference t for velocities*: $t = 2^{24} / f_{CLK}$
*time reference ta² for accelerations*: $ta^2 = 2^{41} / (f_{CLK})^2$

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

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

#### 6.2.2.2 *RAMP_STAT* – Ramp and Reference Switch Status Register

| 0x35, 0x55: *RAMP_STAT* – RAMP AND REFERENCE SWITCH STATUS REGISTER | | | |
|---|---|---|---|
| **R/W** | **Bit** | **Name** | **Comment** |
| R | 13 | *status_sg* | 1: Signals an active stallGuard2 input from the coolStep driver, if enabled.<br><br>*Hint:* When polling this flag, stall events may be missed – activate *sg_stop* to be sure not to miss the stall event. |
| R+C | 12 | *second_move* | 1: Signals that the automatic ramp requires moving back in the opposite direction, e.g. due to on-the-fly parameter change<br>(Flag is cleared upon reading) |
| R | 11 | *t_zerowait_ active* | 1: Signals, that *T_ZEROWAIT* is active after a motor stop. During this time, the motor is in standstill. |
| R | 10 | *vzero* | 1: Signals, that the actual velocity is 0. |
| R | 9 | *position_ reached* | 1: Signals, that the target position is reached.<br>This flag becomes set while *X_ACTUAL* and *X_TARGET* match. |
| R | 8 | *velocity_ reached* | 1: Signals, that the target velocity is reached.<br>This flag becomes set while *V_ACTUAL* and *VMAX* match. |
| R+C | 7 | *event_pos_ reached* | 1: Signals, that the target position has been reached (*pos_reached* becoming active).<br>(Flag and interrupt condition are cleared upon reading)<br>This bit is ORed to the *interrupt output* signal. |
| R+C | 6 | *event_stop_ sg* | 1: Signals an active StallGuard2 stop event.<br>(Flag and interrupt condition are cleared upon reading)<br>This bit is ORed to the *interrupt output* signal. |
| R | 5 | *event_stop_r* | 1: Signals an active stop right condition due to stop switch.<br>The stop condition and the interrupt condition can be removed by setting *RAMP_MODE* to hold mode or by commanding a move to the opposite direction. In *soft_stop* mode, the condition will remain active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor will continue motion.<br>This bit is ORed to the *interrupt output* signal. |
| | 4 | *event_stop_l* | 1: Signals an active stop left condition due to stop switch.<br>The stop condition and the interrupt condition can be removed by setting *RAMP_MODE* to hold mode or by commanding a move to the opposite direction. In *soft_stop* mode, the condition will remain active until the motor has stopped motion into the direction of the stop switch. Disabling the stop switch or the stop function also clears the flag, but the motor will continue motion.<br>This bit is ORed to the *interrupt output* signal. |
| R+C | 3 | *status_latch_r* | 1: Latch right ready<br>(enable position latching using *SWITCH_MODE* settings *latch_r_active* or *latch_r_inactive*)<br>(Flag is cleared upon reading) |
| | 2 | *status_latch_l* | 1: Latch left ready<br>(enable position latching using *SWITCH_MODE* settings *latch_l_active* or *latch_l_inactive*)<br>(Flag is cleared upon reading) |
| R | 1 | *status_stop_r* | Reference switch right status (1=active) |
| | 0 | *status_stop_l* | Reference switch left status (1=active) |

## 5.3 Motor Driver Registers

| **Motor driver register set (Motor 1: 0x60…0x6F, Motor 2: 0x70…0x7F)** | | | | | |
|---|---|---|---|---|---|
| **R/W** | **Addr** | **n** | ***Register*** | **Description / *bit names*** | **Range [Unit]** |
| W | 0x60<br>0x70 | 32 | *MSLUT1[0]*<br>*MSLUT2[0]*<br><br>microstep table entries 0…31 | Each bit gives the difference between microstep x and x+1 when combined with the corresponding *MSLUTSEL W* bits:<br>0: *W*= %00: -1<br>%01: +0<br>%10: +1<br>%11: +2<br>1: *W*= %00: +0<br>%01: +1<br>%10: +2<br>%11: +3<br>This is the differential coding for the first quarter of a wave. Start values for *CUR_A* and *CUR_B* are stored for *MSCNT* position 0 in *START_SIN* and *START_SIN90_120*.<br>*ofs31, ofs30, …, ofs01, ofs00*<br>…<br>*ofs255, ofs254, …, ofs225, ofs224* | 32x 0 or 1<br>*reset default= sine wave table* |
| W | 0x61<br>…<br>0x67<br>0x71<br>…<br>0x77 | 7<br>x<br>32 | *MSLUT1[1…7]*<br>*MSLUT2[1…7]*<br><br>microstep table entries 32…255 | | 7x<br>32x 0 or 1<br>*reset default= sine wave table* |
| W | 0x68<br>0x78 | 32 | ***MSLUTSEL1***<br>***MSLUTSEL2*** | This register defines four segments within each quarter *MSLUT* wave. Four 2 bit entries determine the meaning of a 0 and a 1 bit in the corresponding segment of *MSLUT*.<br>*See separate table!* | *0<X1<X2<X3*<br>*reset default= sine wave table* |
| W | 0x69<br>0x79 | 8<br>+<br>8 | *MSLUTSTART* | bit 7… 0: *START_SIN*<br>bit 23… 16: *START_SIN90_120*<br>*START_SIN* gives the absolute current at microstep table entry 0.<br>*START_SIN90_120* gives the absolute current for microstep table entry at positions 256.<br><br>Start values are transferred to the microstep registers *CUR_A* and *CUR_B*, whenever the reference position *MSCNT*=0 is passed. | *START_SIN*<br>*reset default =0*<br><br>*START_SIN90_120*<br>*reset default =247* |
| R | 0x6A<br>0x7A | 10 | *MSCNT* | Microstep counter. Indicates actual position in the microstep table for *CUR_A*. *CUR_B* uses an offset of 256.<br>*Hint:* Move to a position where *MSCNT* is zero before re-initializing *MSLUTSTART* or *MSLUT* and *MSLUTSEL*. | |
| R | 0x6B<br>0x7B | 9<br>+<br>9 | *MSCURACT* | bit 8… 0: *CUR_A* (signed):<br>Actual microstep current for motor phase A as read from MSLUT (not scaled by current)<br>bit 24… 16: *CUR_B* (signed):<br>Actual microstep current for motor phase B as read from *MSLUT* (not scaled by current) | |
| RW | 0x6C<br>0x7C | 32 | ***CHOPCONF*** | chopper and driver configuration<br>*See separate table!* | |
| W | 0x6D<br>0x7D | 25 | ***COOLCONF*** | coolStep smart current control register and stallGuard2 configuration<br>*See separate table!* | |

| MOTOR DRIVER REGISTER SET (MOTOR 1: 0X60…0X6F, MOTOR 2: 0X70…0X7F) | | | | | |
|---|---|---|---|---|---|
| **R/W** | **Addr** | **n** | ***Register*** | **Description / *bit names*** | **Range [Unit]** |
| R | 0x6F<br>0x7F | 32 | ***DRV_ STATUS*** | stallGuard2 value and driver error flags<br>*See separate table!* | |

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

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

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

### 5.3.1 *MSLUTSEL* – Look up Table Segmentation Definition

| 0X68, 0X78: *MSLUTSEL* – LOOK UP TABLE SEGMENTATION DEFINITION | | | |
|---|---|---|---|
| **Bit** | **Name** | **Function** | **Comment** |
| 31 | *X3* | LUT segment 3 start | The sine wave look up table can be divided into up to four segments using an individual step width control entry *Wx*. The segment borders are selected by *X1*, *X2* and *X3*.<br><br>Segment 0 goes from 0 to *X1*-1.<br>Segment 1 goes from *X1* to *X2*-1.<br>Segment 2 goes from *X2* to *X3*-1.<br>Segment 3 goes from *X3* to 255.<br><br>For defined response the values shall satisfy:<br>0<*X1*<*X2*<*X3* |
| 30 | | | |
| 29 | | | |
| 28 | | | |
| 27 | | | |
| 26 | | | |
| 25 | | | |
| 24 | | | |
| 23 | *X2* | LUT segment 2 start | |
| 22 | | | |
| 21 | | | |
| 20 | | | |
| 19 | | | |
| 18 | | | |
| 17 | | | |
| 16 | | | |
| 15 | *X1* | LUT segment 1 start | |
| 14 | | | |
| 13 | | | |
| 12 | | | |
| 11 | | | |
| 10 | | | |
| 9 | | | |
| 8 | | | |
| 7 | *W3* | LUT width select from *ofs(X3)* to *ofs255* | Width control bit coding *W0…W3*:<br>%00: MSLUT entry 0, 1 select: -1, +0<br>%01: MSLUT entry 0, 1 select: +0, +1<br>%10: MSLUT entry 0, 1 select: +1, +2<br>%11: MSLUT entry 0, 1 select: +2, +3 |
| 6 | | | |
| 5 | *W2* | LUT width select from *ofs(X2)* to *ofs(X3-1)* | |
| 4 | | | |
| 3 | *W1* | LUT width select from *ofs(X1)* to *ofs(X2-1)* | |
| 2 | | | |
| 1 | *W0* | LUT width select from *ofs00* to *ofs(X1-1)* | |
| 0 | | | |

### 5.3.2 *CHOPCONF* – Chopper Configuration

| 0x6C, 0x7C: CHOPCONF – Chopper Configuration | | | |
|---|---|---|---|
| **Bit** | **Name** | **Function** | **Comment** |
| 31 | - | reserved | set to 0 |
| 30 | *diss2g* | short to GND protection disable | 0: Short to GND protection is on<br>1: Short to GND protection is disabled |
| 29 | - | reserved | set to 0 |
| 28 | - | reserved | set to 0 |
| 27 | - | reserved | set to 0 |
| 26 | - | reserved | set to 0 |
| 25 | - | reserved | set to 0 |
| 24 | - | reserved | set to 0 |
| 23 | *sync3* | SYNC<br>PWM synchronization clock | This register allows synchronization of the chopper for both phases of a two phase motor in order to avoid the occurrence of a beat, especially at low motor velocities. It is automatically switched off above VHIGH.<br>%0000: Chopper sync function chopSync off<br>%0001 … %1111:<br>Synchronization with $f_{SYNC} = f_{CLK}/(sync*64)$<br>Hint: Set TOFF to a low value, so that the chopper cycle is ended, before the next sync clock pulse occurs. Set for the double desired chopper frequency for chm=0, for the desired base chopper frequency for chm=1. |
| 22 | *sync2* | | |
| 21 | *sync1* | | |
| 20 | *sync0* | | |
| 19 | *vhighchm* | high velocity chopper mode | This bit enables switching to *chm*=1 and *fd*=0, when *VHIGH* is exceeded. This way, a higher velocity can be achieved. Can be combined with *vhighfs*=1. If set, the *TOFF* setting automatically becomes doubled during high velocity operation in order to avoid doubling of the chopper frequency. |
| 18 | *vhighfs* | high velocity fullstep selection | This bit enables switching to fullstep, when *VHIGH* is exceeded. Switching takes place only at 45° position. The fullstep target current uses the current value from the microstep table at the 45° position. |
| 17 | *vsense* | sense resistor voltage based current scaling | 0: Low sensitivity, high sense resistor voltage<br>1: High sensitivity, low sense resistor voltage |
| 16 | *tbl1* | TBL<br>blank time select | %00 … %11:<br>Set comparator blank time to 16, 24, 36 or 54 clocks<br>Hint: %10 is recommended for most applications |
| 15 | *tbl0* | | |
| 14 | *chm* | chopper mode | 0: Standard mode (spreadCycle)<br>1: Constant off time with fast decay time. Fast decay time is also terminated when the negative nominal current is reached. Fast decay is after on time. |
| 13 | *rndtf* | random *TOFF* time | 0: Chopper off time is fixed as set by *TOFF*<br>1: Random mode, *TOFF* is random modulated by $d_{NCLK}$= -12 … +3 clocks. |
| 12 | *disfdcc* | fast decay mode | chm=1:<br>disfdcc=1 disables current comparator usage for termination of the fast decay cycle |
| 11 | *fd3* | *TFD* [3] | chm=1:<br>MSB of fast decay time setting TFD |

| 0X6C, 0X7C: CHOPCONF – CHOPPER CONFIGURATION | | | | |
|---|---|---|---|---|
| **Bit** | **Name** | **Function** | **Comment** | |
| 10 | *hend3* | *HEND*<br>hysteresis low value<br>*OFFSET*<br>sine wave offset | *chm*=0 | %0000 … %1111:<br>Hysteresis is -3, -2, -1, 0, 1, …, 12<br>(1/512 of this setting adds to current setting)<br>This is the hysteresis value which becomes used for the hysteresis chopper. |
| 9 | *hend2* | | | |
| 8 | *hend1* | | | |
| 7 | *hend0* | | *chm*=1 | %0000 … %1111:<br>Offset is -3, -2, -1, 0, 1, …, 12<br>This is the sine wave offset and 1/512 of the value becomes added to the absolute value of each sine wave entry. |
| 6 | *hstrt2* | *HSTRT*<br>hysteresis start value added to *HEND* | *chm*=0 | %000 … %111:<br>Add 1, 2, …, 8 to hysteresis low value *HEND*<br>(1/512 of this setting adds to current setting)<br>Attention: Effective *HEND*+*HSTRT* ≤ 16.<br>Hint: Hysteresis decrement is done each 16 clocks |
| 5 | *hstrt1* | | | |
| 4 | *hstrt0* | | | |
| | | *TFD* [2..0]<br>fast decay time setting | *chm*=1 | Fast decay time setting (MSB: *fd3*):<br>%0000 … %1111:<br>Fast decay time setting *TFD* with<br>*NCLK*= 32**HSTRT* (%0000: slow decay only) |
| 3 | *toff3* | *TOFF* off time<br>and driver enable | Off time setting controls duration of slow decay phase<br>*NCLK*= 12 + 32**TOFF*<br>%0000: Driver disable, all bridges off<br>%0001: 1 – use only with TBL ≥ 2<br>%0010 … %1111: 2 … 15 | |
| 2 | *toff2* | | | |
| 1 | *toff1* | | | |
| 0 | *toff0* | | | |

### 5.3.3 *COOLCONF* – Smart Energy Control coolStep and stallGuard2

| 0x6D, 0x7D: *COOLCONF* – Smart Energy control coolStep and stallGuard2 | | | |
|---|---|---|---|
| **Bit** | **Name** | **Function** | **Comment** |
| … | - | reserved | set to 0 |
| 24 | *sfilt* | stallGuard2 filter enable | 0: Standard mode, high time resolution for stallGuard2<br>1: Filtered mode, stallGuard2 signal updated for each four fullsteps only to compensate for motor pole tolerances |
| 23 | - | reserved | set to 0 |
| 22 | *sgt6* | stallGuard2 threshold value | This signed value controls stallGuard2 level for stall output and sets the optimum measurement range for readout. A lower value gives a higher sensitivity. Zero is the starting value working with most motors.<br>-64 to +63: A higher value makes stallGuard2 less sensitive and requires more torque to indicate a stall. |
| 21 | *sgt5* | | |
| 20 | *sgt4* | | |
| 19 | *sgt3* | | |
| 18 | *sgt2* | | |
| 17 | *sgt1* | | |
| 16 | *sgt0* | | |
| 15 | *seimin* | minimum current for smart current control | 0: 1/2 of current setting (*IRUN*)<br>1: 1/4 of current setting (*IRUN*) |
| 14 | *sedn1* | current down step speed | %00: For each 32 stallGuard2 values decrease by one<br>%01: For each 8 stallGuard2 values decrease by one<br>%10: For each 2 stallGuard2 values decrease by one<br>%11: For each stallGuard2 value decrease by one |
| 13 | *sedn0* | | |
| 12 | - | reserved | set to 0 |
| 11 | *semax3* | stallGuard2 hysteresis value for smart current control | If the stallGuard2 result is equal to or above (*SEMIN*+*SEMAX*+1)*32, the motor current becomes decreased to save energy.<br>%0000 … %1111: 0 … 15 |
| 10 | *semax2* | | |
| 9 | *semax1* | | |
| 8 | *semax0* | | |
| 7 | - | reserved | set to 0 |
| 6 | *seup1* | current up step width | Current increment steps per measured stallGuard2 value<br>%00 … %11: 1, 2, 4, 8 |
| 5 | *seup0* | | |
| 4 | - | reserved | set to 0 |
| 3 | *semin3* | minimum stallGuard2 value for smart current control and smart current enable | If the stallGuard2 result falls below *SEMIN**32, the motor current becomes increased to reduce motor load angle.<br>%0000: smart current control coolStep off<br>%0001 … %1111: 1 … 15 |
| 2 | *semin2* | | |
| 1 | *semin1* | | |
| 0 | *semin0* | | |

### 5.3.4 DRV_STATUS – stallGuard2 Value and Driver Error Flags

**0X6F, 0X7F: DRV_STATUS – STALLGUARD2 VALUE AND DRIVER ERROR FLAGS**

| Bit | Name | Function | Comment |
|---|---|---|---|
| 31 | *stst* | standstill indicator | This flag indicates motor stand still in each operation mode. |
| 30 | *olb* | open load indicator phase B | 1: Open load detected on phase A or B.<br>*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. |
| 29 | *ola* | open load indicator phase A | |
| 28 | *s2gb* | short to ground indicator phase B | 1: Short to GND detected on phase A or B.<br>The driver becomes disabled. The flags stay active, until the driver is disabled by software or by the ENN input. |
| 27 | *s2ga* | short to ground indicator phase A | |
| 26 | *otpw* | overtemperature pre-warning flag | 1: Overtemperature pre-warning threshold is exceeded.<br>The overtemperature pre-warning flag is common for both drivers. |
| 25 | *ot* | overtemperature flag | 1: Overtemperature limit has been reached. Drivers become disabled until *otpw* is also cleared due to cooling down of the IC.<br>The overtemperature flag is common for both drivers. |
| 24 | *stallGuard* | stallGuard2 status | 1: Motor stall detected (*SG_RESULT*=0) |
| 23<br>22<br>21 | - | reserved | Ignore these bits |
| 20<br>19<br>18<br>17<br>16 | *CS ACTUAL* | actual motor current / smart energy current | Actual current control scaling, for monitoring smart energy current scaling controlled via settings in register *COOLCONF*. |
| 15 | *fsactive* | full step active indicator | 1: Indicates that the driver has switched to fullstep as defined by chopper mode settings and velocity thresholds. |
| 14<br>13<br>12<br>11<br>10 | - | reserved | Ignore these bits |
| 9<br>8<br>7<br>6<br>5<br>4<br>3<br>2<br>1<br>0 | *SG_ RESULT* | stallGuard2 result respectively PWM on time for coil A in stand still for motor temperature detection | Mechanical load measurement:<br>The stallGuard2 result gives a means to measure mechanical motor load. A higher value means lower mechanical load. A value of 0 signals highest load. With optimum *SGT* setting, this is an indicator for a motor stall. The stall detection compares *SG_RESULT* to 0 in order to detect a stall. *SG_RESULT* is used as a base for coolStep operation, by comparing it to a programmable upper and a lower limit.<br><br>Temperature measurement:<br>In standstill, no stallGuard2 result can be obtained. *SG_RESULT* shows the chopper on-time for motor coil A instead. If the motor is moved to a determined microstep position at a certain current setting, a comparison of the chopper on-time can help to get a rough estimation of motor temperature. As the motor heats up, its coil resistance rises and the chopper on-time increases. |

# 6 Current Setting

The internal 5 V supply voltage available at the pin 5VOUT is used as a reference for the coil current regulation based on the sense resistor voltage measurement. The desired maximum motor current is set by selecting an appropriate value for the sense resistor. The sense resistor voltage range can be selected by the *vsense* bit in *CHOPCONF*. The low sensitivity setting (high sense resistor voltage, *vsense*=0) brings best and most robust current regulation, while high sensitivity (low sense resistor voltage, *vsense*=1) reduces power dissipation in the sense resistor. The high sensitivity setting reduces the power dissipation in the sense resistor by nearly half.

After choosing the *vsense* setting and selecting the sense resistor, the currents to both coils are scaled by the 5-bit current scale parameters (*IHOLD*, *IRUN*). The sense resistor value is chosen so that the maximum desired current (or slightly more) flows at the maximum current setting (*IRUN* = %11111).

Using the internal sine wave table, which has the amplitude of 248, the RMS motor current can be calculated by:

$$I_{RMS} = \frac{CS+1}{32} * \frac{V_{FS}}{R_{SENSE}} * \frac{1}{\sqrt{2}}$$

The momentary motor current is calculated by:

$$I_{MOT} = \frac{CUR_{A/B}}{248} * \frac{CS+1}{32} * \frac{V_{FS}}{R_{SENSE}}$$

*CS* is the current scale setting as set by the *IHOLD* and *IRUN* and coolStep.
$V_{FS}$ is the full scale voltage as determined by *vsense* control bit (please refer to electrical characteristics, $V_{SRTL}$ and $V_{SRTH}$).
$CUR_{A/B}$ is the actual value from the internal sine wave table.

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

## 6.1 Sense Resistors

Sense resistors should be carefully selected. The full motor current flows through the sense resistors. They also see the switching spikes from the MOSFET bridges. A low-inductance type such as film or composition resistors is required to prevent spikes causing ringing on the sense voltage inputs leading to unstable measurement results. A low-inductance, low-resistance PCB layout is essential. Any common GND path for the two sense resistors must be avoided, because this would lead to coupling between the two current sense signals. A massive ground plane is best. Please also refer to layout considerations in chapter 15.3.

The sense resistor needs to be able to conduct the peak motor coil current in motor standstill conditions, unless standby power is reduced. Under normal conditions, the sense resistor sees a bit less than the coil RMS current, because no current flows through the sense resistor during the slow decay phases.

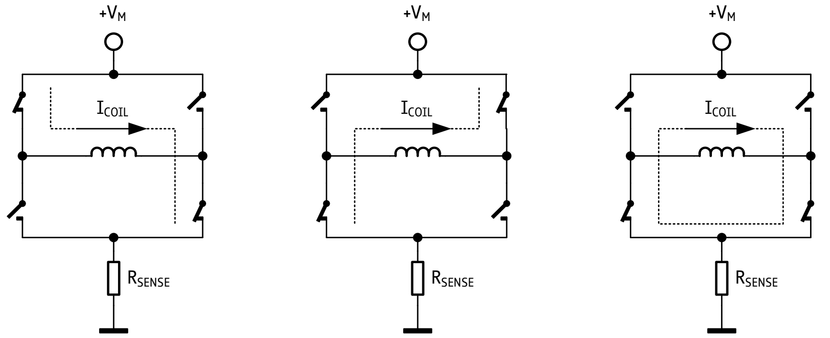
The peak sense resistor power dissipation is:

$$P_{RSMAX} = {I_{COIL}}^2 * R_{SENSE}$$

For high current applications, power dissipation is halved by using the low *vsense* setting and using an adapted resistance value. Please be aware, that in this case any voltage drop in PCB traces has a larger influence on the result. A compact layout with massive ground plane is best to avoid parasitic resistance effects.

# 7 Chopper Operation

The currents through both motor coils are controlled using choppers. The choppers work independently of each other. In Figure 7.1 the different chopper phases are shown.




On Phase: current flows in direction of target current

Fast Decay Phase: current flows in opposite direction of target current

Slow Decay Phase: current re-circulation

**Figure 7.1 Chopper phases**

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

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

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

The chopper frequency is an important parameter for a chopped motor driver. A too low frequency might generate audible noise. A higher frequency reduces current ripple in the motor, but with a too high frequency magnetic losses may rise. Also power dissipation in the driver rises with increasing frequency due to the increased influence of switching slopes causing dynamic dissipation. Therefore, a compromise needs to be found. Most motors are optimally working in a frequency range of 20 kHz to 40 kHz. The chopper frequency is influenced by a number of parameter settings as well as by the motor inductivity and supply voltage.

*A chopper frequency in the range of 20 kHz to 40 kHz gives a good result for most motors. A higher frequency leads to increased switching losses. It is advised to check the resulting frequency and to work below 50 kHz.*

Three parameters are used for controlling both chopper modes:

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

## 7.1 spreadCycle 2-Phase Motor Chopper

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

Each chopper cycle is comprised of an on phase, a slow decay phase, a fast decay phase and a second slow decay phase (see Figure 7.2). The slow decay phases limit the maximum chopper frequency and are important for low motor and driver power dissipation. The hysteresis start setting limits the chopper frequency by forcing the driver to introduce a minimum amount of current ripple into the motor coils. The motor inductance limits the ability of the chopper to follow a changing motor current. The duration of the on phase and the fast decay phase must be longer than the blanking time, because the current comparator is disabled during blanking. This requirement is satisfied by choosing a positive value for the hysteresis as can be estimated by the following calculation:

$$dI_{COILBLANK} = V_M * \frac{t_{BLANK}}{L_{COIL}}$$

$$dI_{COILSD} = R_{COIL} * I_{COIL} * \frac{2 * t_{SD}}{L_{COIL}}$$

Where:
$dI_{COILBLANK}$ is the coil current change during the blanking time
$dI_{COILSD}$ is the coil current change during the slow decay time
$t_{SD}$ is the slow decay time
$t_{BLANK}$ is the blank time (as set by *TBL*),
$V_M$ is the motor supply voltage,
$I_{COIL}$ is the peak motor coil current at the maximum motor current setting CS,
$R_{COIL}$ and $L_{COIL}$ are motor coil inductivity and motor coil resistance.

With this, a lower limit for the start hysteresis setting can be determined:

$$Hysteresis\ Start \geq (dI_{COILBLANK} + dI_{COILSD}) * \frac{2 * 248}{I_{COIL}} * \frac{CS + 1}{32}$$

*Example:*

For a 42mm stepper motor with 7.5 mH, 4.5 Ω phase and 1 A RMS current at *IRUN*=31, i.e. 1.41 A peak current, at 24 V with a blank time of 1.5 µs:

$$dI_{COILBLANK} = 24\ V * \frac{2\ \mu s}{7.5\ mH} = 6.4\ mA$$

$$dI_{COILSD} = 4.5\ \Omega * 1.41\ A * \frac{2 * 5\ \mu s}{7.5\ mH} = 8.5\ mA$$

With this, the minimum hysteresis start setting is 5.2. A value in the range 6 to 10 can be used.

***An Excel calculation spreadsheet is provided for the ease of use.***

As experiments show, the setting is quite independent of the motor, because higher current motors typically also have a lower coil resistance. Choosing a medium default value for the hysteresis (for example, effective *HSTART*+*HEND*=10) normally fits most applications. The setting can be optimized by experimenting with the motor: A too low setting will result in reduced microstep accuracy, while a too high setting will lead to more chopper noise and motor power dissipation. When measuring the sense resistor voltage in motor standstill at a medium coil current with an oscilloscope, a too low setting shows a fast decay phase not longer than the blanking time. When the fast decay time becomes slightly longer than the blanking time, the setting is optimum. You can reduce the off-time setting, if this is hard to reach.

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

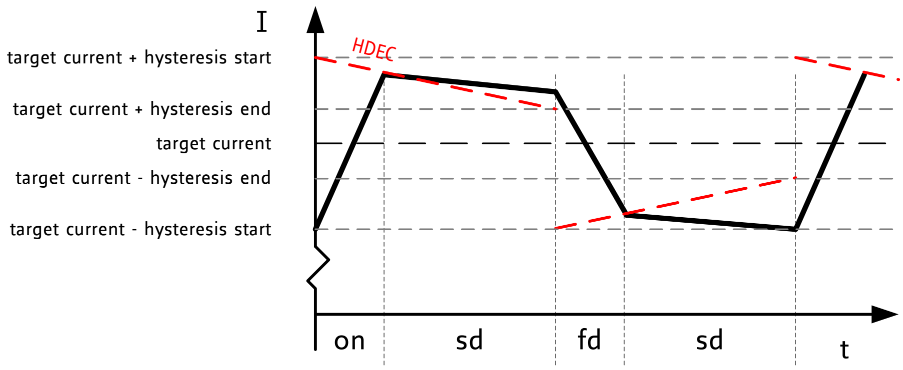



**Figure 7.2 spreadCycle chopper scheme showing coil current during a chopper cycle**

Two parameters control spreadCycle mode:

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

*Example:*

In the example above a hysteresis start of 7 has been chosen. You might decide to not use hysteresis decrement. In this case set:

*HEND*=10 (sets an effective end value of 7)
*HSTRT*=0 (sets minimum hysteresis)

In order to take advantage of the variable hysteresis, we can set hysteresis end to about half of the start value, e.g. 4. The resulting configuration register values are as follows:

*HEND*=7 (sets an effective end value of 4)
*HSTRT*=2 (sets an effective start value of hysteresis end +3)

## 7.2 Classic 2-Phase Motor Constant Off Time Chopper

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

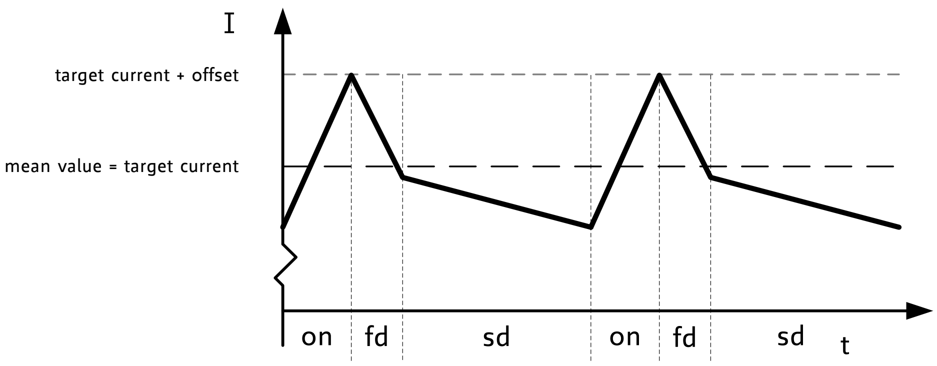



**Figure 7.3 Classic const. off time chopper with offset showing coil current**

After tuning the fast decay time, the offset should be tuned for a smooth zero crossing. This is necessary because the fast decay phase makes the absolute value of the motor current lower than the target current (see Figure 7.4). If the zero offset is too low, the motor stands still for a short moment during current zero crossing. If it is set too high, it makes a larger microstep. Typically, a positive offset setting is required for smoothest operation.

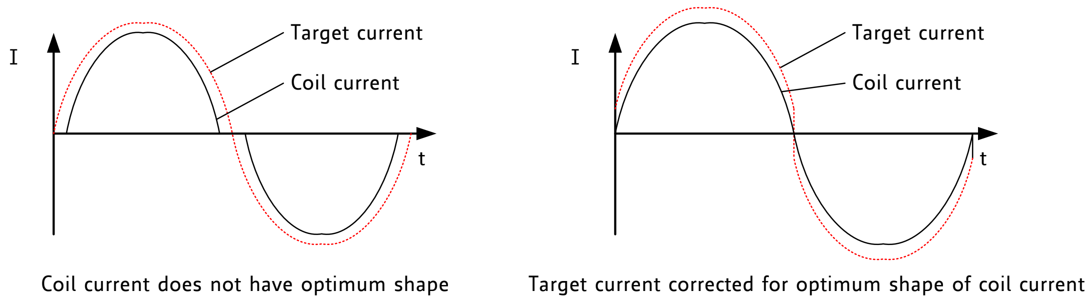



**Figure 7.4 Zero crossing with classic chopper and correction using sine wave offset**

Three parameters control constant off-time mode:

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

## 7.3 Random Off Time

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

*A common factor, which can cause motor noise, is a bad PCB layout causing coupling of both sense resistor voltages (please refer layouts hint in chapter 15.3).*

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

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

## 7.4 chopSync2 for Quiet Motors

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

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

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

*Example:*

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

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

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

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

# 8 Driver Diagnostic Flags

The TMC5031 drivers supply a complete set of diagnostic and protection capabilities, like short to GND protection and undervoltage detection. A detection of an open load condition allows testing if a motor coil connection is interrupted. See the *DRV_STATUS* table for details.

## 8.1 Temperature Measurement

The TMC5031 integrates a two level temperature sensor (120°C prewarning and 150°C thermal shutdown) for diagnostics and for protection of the IC against excess heat. The heat is mainly generated by the voltage regulator and the motor driver stages. The central temperature detector can detect heat accumulation on the chip, i.e. due to missing convection cooling or rising environment temperature. It cannot detect overheating of the power transistors in all cases, e.g. with bad PCB layout, because heat transfer between power transistors and temperature sensor depends on the PCB layout and environmental conditions. Most critical situations, where the driver MOSFETs could be overheated, are avoided when enabling the short to GND protection. For many applications, the overtemperature prewarning will indicate an abnormal operation situation and can be used to initiate user warning or power reduction measures like motor current reduction. If continuous operation in hot environments is necessary, a more precise processor based temperature measurement should be used to realize application specific overtemperature detection. The thermal shutdown is just an emergency measure and temperature rising to the shutdown level should be prevented by design.

After triggering the overtemperature sensor (*ot* flag), the driver remains switched off until the system temperature falls below the prewarning level (*otpw*) to avoid continuous heating to the shut down level.

## 8.2 Short to GND Protection

The TMC5031 power stages are protected against a short circuit condition by an additional measurement of the current flowing through the highside MOSFETs. This is important, as most short circuit conditions result from a motor cable insulation defect, e.g. when touching the conducting parts connected to the system ground. The short detection is protected against spurious triggering, e.g. by ESD discharges, by retrying three times before switching off the motor.

Once a short condition is safely detected, the corresponding driver bridge becomes switched off, and the *s2ga* or *s2gb* flag becomes set. In order to restart the motor, the user must intervene by disabling and re-enabling the driver. It should be noted, that the short to GND protection cannot protect the system and the power stages for all possible short events, as a short event is rather undefined and a complex network of external components may be involved. Therefore, short circuits should basically be avoided.

## 8.3 Open Load Diagnostics

Interrupted cables are a common cause for systems failing, e.g. when connectors are not firmly plugged. The TMC5031 detects open load conditions by checking, if it can reach the desired motor coil current. This way, also undervoltage conditions, high motor velocity settings or short and overtemperature conditions may cause triggering of the open load flag, and inform the user, that motor torque may suffer. In motor stand still, open load cannot be measured, as the coils might eventually have zero current.

In order to safely detect an interrupted coil connection, read out the open load flags at low or nominal motor velocity operation, only. However, the *ola* and *olb* flags have just informative character and do not cause any action of the driver.

# 9 Ramp Generator

The TMC5031 integrates a new type of ramp generator, which offers faster machine operation compared to the classical linear acceleration ramps. The sixPoint ramp generator allows adapting the acceleration ramps to the torque curves of a stepper motor and uses two different acceleration settings each for the acceleration phase and for the deceleration phase. See Figure 9.2.

## 9.1 Real World Unit Conversion

The TMC5031 uses its internal or external clock signal as a time reference for all internal operations. Thus, all time, velocity and acceleration settings are referenced to $f_{CLK}$. For best stability and reproducibility, it is recommended to use an external quartz oscillator as a time base, or to provide a clock signal from a microcontroller.

The units of a TMC5031 register content are written as register[5031].

| PARAMETER VS. UNITS | | |
|---|---|---|
| **Parameter / Symbol** | **Unit** | **calculation / description / comment** |
| $f_{CLK}$[Hz] | [Hz] | clock frequency of the TMC5031 in [Hz] |
| s | [s] | second |
| US | µstep | |
| FS | fullstep | |
| µstep velocity v[Hz] | µsteps / s | v[Hz] = v[5031] * ( $f_{CLK}$[Hz]/2 / 2^23 ) |
| µstep acceleration a[Hz/s] | µsteps / s^2 | a[Hz/s] = a[5031] * $f_{CLK}$[Hz]^2 / (512*256) / 2^24 |
| USC microstep count | counts | microstep resolution in number of microsteps (i.e. the number of microsteps between two fullsteps – normally 256) |
| rotations per second v[rps] | rotations / s | v[rps] = v[µsteps/s] / USC / FSC<br>FSC: motor fullsteps per rotation, e.g. 200 |
| rps acceleration a[rps/s^2] | rotations / s^2 | a[rps/s^2] = a[µsteps/s^2] / USC / FSC |
| ramp steps[µsteps] = rs | µsteps | rs = (v[5031])^2 / a[5031] / 2^8<br>microsteps during linear acceleration ramp (assuming acceleration from 0 to v) |

## 9.2 Ramp Generator Functionality

For the ramp generator register set, please refer to the chapter 5.2.

### 9.2.1 Ramp Mode

The ramp generator delivers two phase acceleration and two phase deceleration ramps with additional programmable start and stop velocities (see Figure 9.1).

*Note!*
The start velocity can be set to zero, if not used.
The stop velocity can be set to one, if not used.
Take care to always set *VSTOP* identical to or above *VSTART*. This ensures that even a short motion can be terminated successfully at the target position.

The two different sets of acceleration and deceleration can be combined freely. *A common transition speed V1 allows for velocity dependent switching between both acceleration and deceleration settings*. A typical use case will use lower acceleration and deceleration values at higher velocities, as the motors torque declines at higher velocity. When considering friction in the system, it becomes clear, that typically deceleration of the system is quicker than acceleration. Thus, deceleration values can be higher in many applications. This way, operation speed of the motor in time critical applications can be maximized.

As target positions and ramp parameters may be changed any time during the motion, the motion controller will always use the optimum (fastest) way to reach the target, while sticking to the constraints set by the user. This way it might happen, that the motion becomes automatically stopped, crosses zero and drives back again. This case is flagged by the special flag *second_move.*

## 9.2.2 Start and Stop Velocity

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

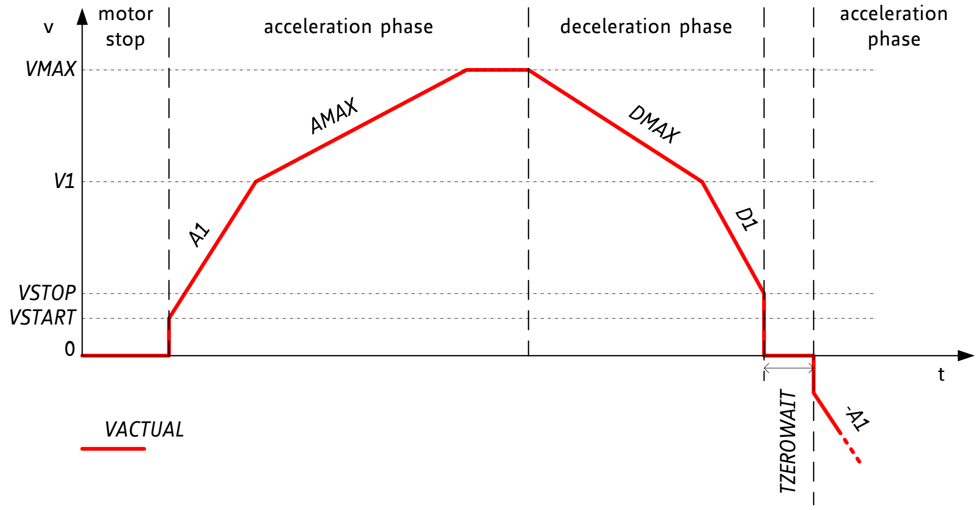



**Figure 9.1 Ramp generator velocity trace showing consequent move in negative direction**

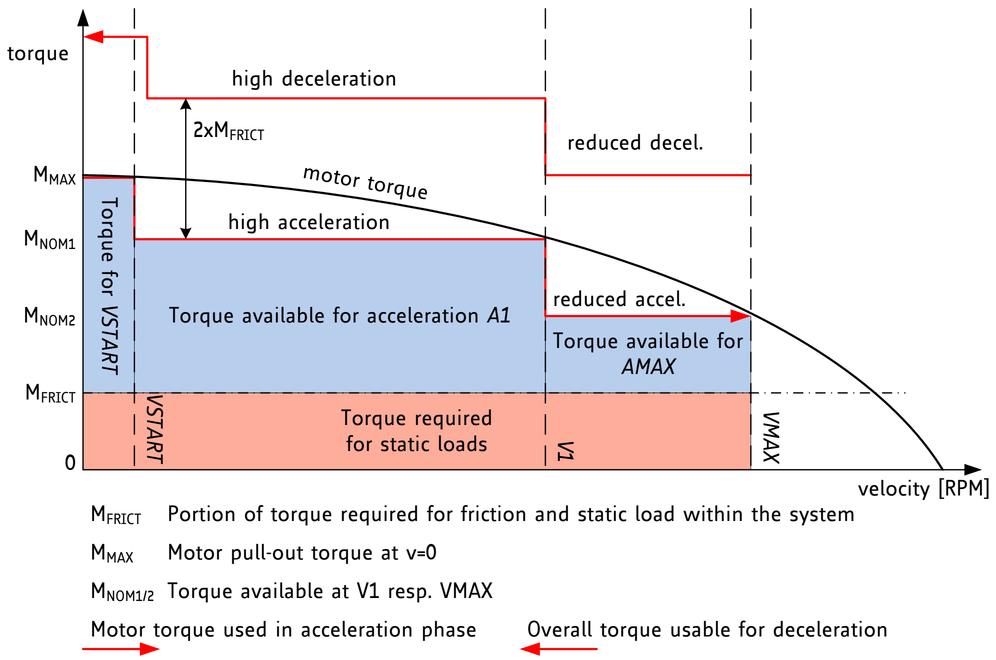



**Figure 9.2 Illustration of optimized motor torque usage with TMC5031 ramp generator**

### 9.2.3 Velocity Mode

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

In order to decelerate the motor to stand still, it is sufficient to set *VMAX* to zero. The flag *vzero* signals standstill of the motor. The flag *velocity_reached* always signals, that the target velocity has been reached.

## 9.3 Velocity Thresholds

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

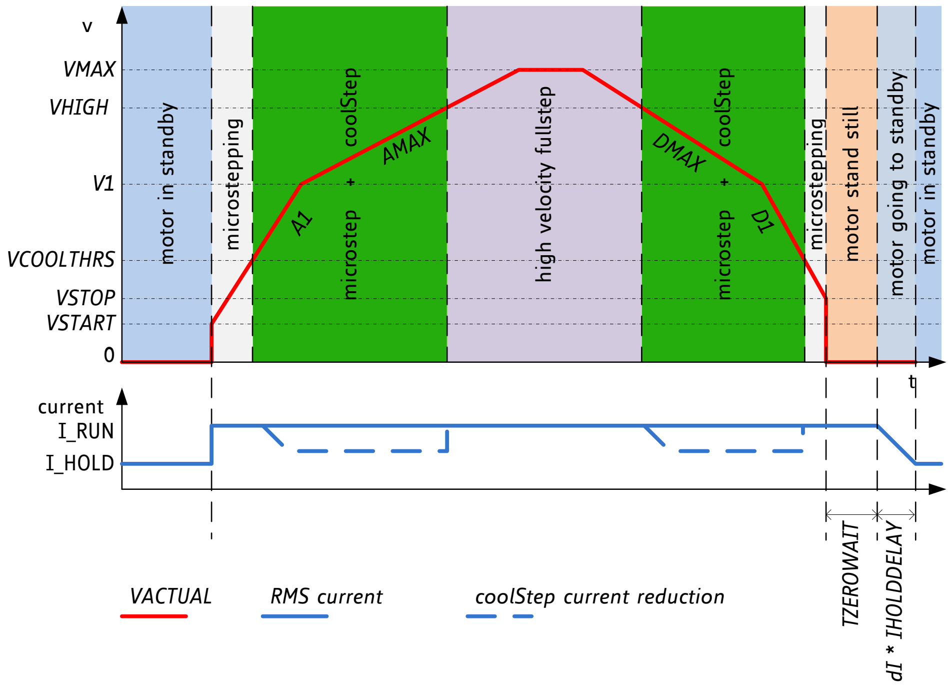



**Figure 9.3 Ramp generator velocity dependent motor control**

Since it is not necessary to differentiate the velocity to the last detail, the velocity thresholds use a reduced number of bits for comparison and the lower eight bits of the compare values become ignored.

## 9.4 Reference Switches

Prior to normal operation of the drive an absolute reference position must be set. The reference position can be found using a mechanical stop which can be detected by stall detection, or by a reference switch.

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

- it can be stopped abruptly, when a switch is hit. This is useful in an emergency case.
- Or the motor can be softly decelerated to zero using deceleration settings.

*Note:*
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.

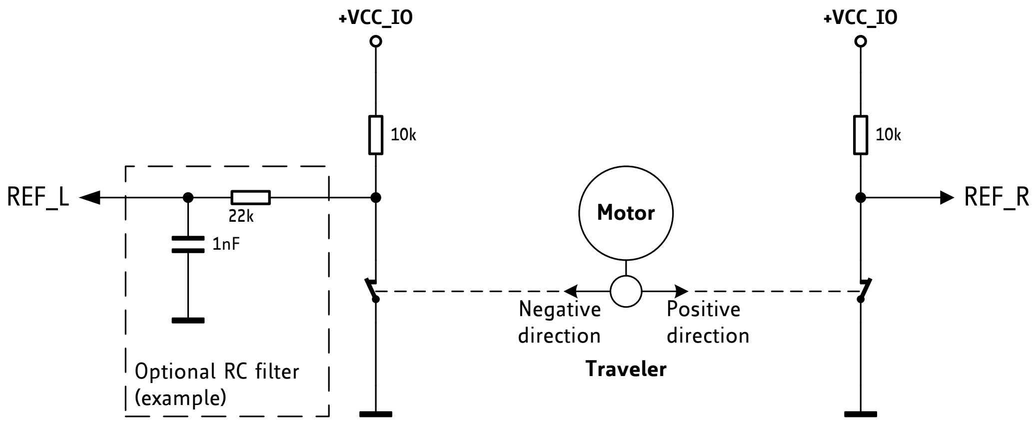



**Figure 9.4 Using reference switches (example)**

Normally open or normally closed switches can be used by programming the switch polarity or selecting the pull-up or pull-down resistor configuration. A normally closed switch is failsafe with respect to an interrupt of the switch connection. Switches which can be used are:

- mechanical switches,
- photo interrupters, or
- hall sensors.

Be careful to select resistors matching your switch requirements!
In case of long cables additional RC filtering might be required near the TMC5031 reference inputs. Adding an RC filter will also reduce the danger of destroying the logic level inputs by wiring faults, but it will add a certain delay which should be considered with respect to the application.

### Implementing a Homing Procedure

- Make sure, that the switch is not pressed.
- Activate position latching upon the desired switch event and activate motor (soft) stop upon active switch.
- Start a motion ramp into the direction of the switch. (Move to a more negative position for a left switch, to a more positive position for a right switch). You may timeout this motion by using a position ramping command.
- As soon as the switch is hit, the position becomes latched and the motor is stopped. Wait until the motor is in standstill again.
- Switch the ramp generator to hold mode and calculate the difference between the latched position and the actual position.
- Write the calculated difference into the actual position register. Now, the homing is finished. A move to position 0 will bring back the motor exactly to the switching point.

# 10 stallGuard2 Load Measurement

stallGuard2 provides an accurate measurement of the load on the motor. It can be used for stall detection as well as other uses at loads below those which stall the motor, such as coolStep load-adaptive current reduction. The stallGuard2 measurement value changes linearly over a wide range of load, velocity, and current settings, as shown in Figure 10.1. At maximum motor load, the value goes to zero or near to zero. This corresponds to a load angle of 90° between the magnetic field of the coils and magnets in the rotor. This also is the most energy-efficient point of operation for the motor.

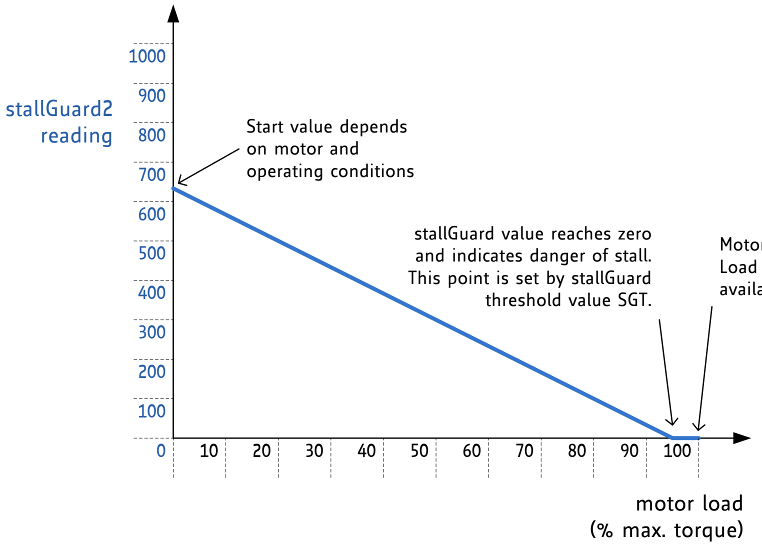



**Figure 10.1 Function principle of stallGuard2**

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

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

## 10.1 Tuning the stallGuard2 Threshold SGT

The stallGuard2 value *SG* is affected by motor-specific characteristics and application-specific demands on load and velocity. Therefore the easiest way to tune the stallGuard2 threshold *SGT* for a specific motor type and operating conditions is interactive tuning in the actual application.

The procedure is:

1. Operate the motor at the normal operation velocity for your application and monitor *SG*.
2. Apply slowly increasing mechanical load to the motor. If the motor stalls before *SG* reaches zero, decrease *SGT*. If *SG* reaches zero before the motor stalls, increase *SGT*. A good *SGT* starting value is zero. *SGT* is signed, so it can have negative or positive values.
3. Now enable *sg_stop* and make sure, that the motor is safely stopped whenever it is stalled. Increase *SGT* if the motor becomes stopped before a stall occurs.
4. The optimum setting is reached when *SG* is between 0 and roughly 100 at increasing load shortly before the motor stalls, and *SG* increases by 100 or more without load. *SGT* in most cases can be tuned for a certain motion velocity or a velocity range. Make sure, that the setting works reliable in a certain range (e.g. 80% to 120% of desired velocity) and also under extreme motor conditions (lowest and highest applicable temperature).

*SG goes to zero when the motor stalls and the ramp generator can be programmed to stop the motor upon a stall event by enabling sg_stop in SW_MODE.*

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

| |
|---|
| *Note!* <br> Application Note 002 *Parameterization of stallGuard2 & coolStep* is available on www.trinamic.com. |

### 10.1.1 Variable Velocity Operation

The *SGT* setting chosen as a result of the previously described *SGT* tuning (chapter 0) can be used for a certain velocity range. Outside this range, a stall may not be detected safely, and coolStep might not give the optimum result.

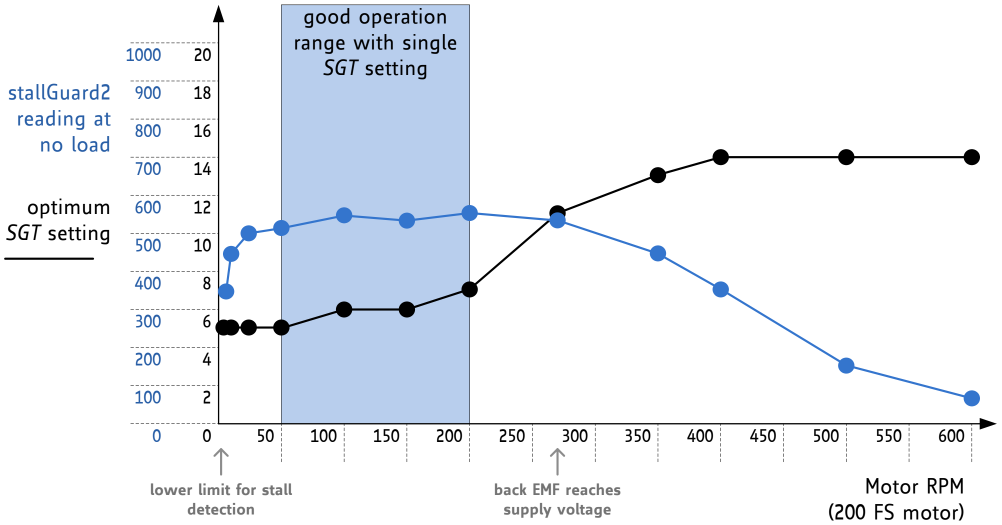



**Figure 10.2 Example: Optimum SGT setting and stallGuard2 reading with an example motor**

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

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

### 10.1.2 Small Motors with High Torque Ripple and Resonance

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

### 10.1.3 Temperature Dependence of Motor Coil Resistance

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

### 10.1.4 Accuracy and Reproducibility of stallGuard2 Measurement

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

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

## 10.2 stallGuard2 Measurement Frequency and Filtering

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

## 10.3 Detecting a Motor Stall

To safely detect a motor stall the stall threshold must be determined using a specific *SGT* setting. Therefore, you need to determine the maximum load the motor can drive without stalling and to monitor the *SG* value at this load, e.g. some value within the range 0 to 100. The stall threshold should be a value safely within the operating limits, to allow for parameter stray. The response at an *SGT* setting at or near 0 gives some idea on the quality of the signal: Check the *SG* value without load and with maximum load. They should show a difference of at least 100 or a few 100, which shall be large compared to the offset. If you set the *SGT* value in a way, that a reading of 0 occurs at maximum motor load, the stall can be automatically detected by the motion controller to issue a motor stop.

## 10.4 Limits of stallGuard2 Operation

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

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

# 11 coolStep Operation

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

## 11.1 User Benefits

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

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

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

## 11.2 Setting up for coolStep

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

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

**FIGURE 11.1 SHOWS THE OPERATING REGIONS OF COOLSTEP:**

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

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

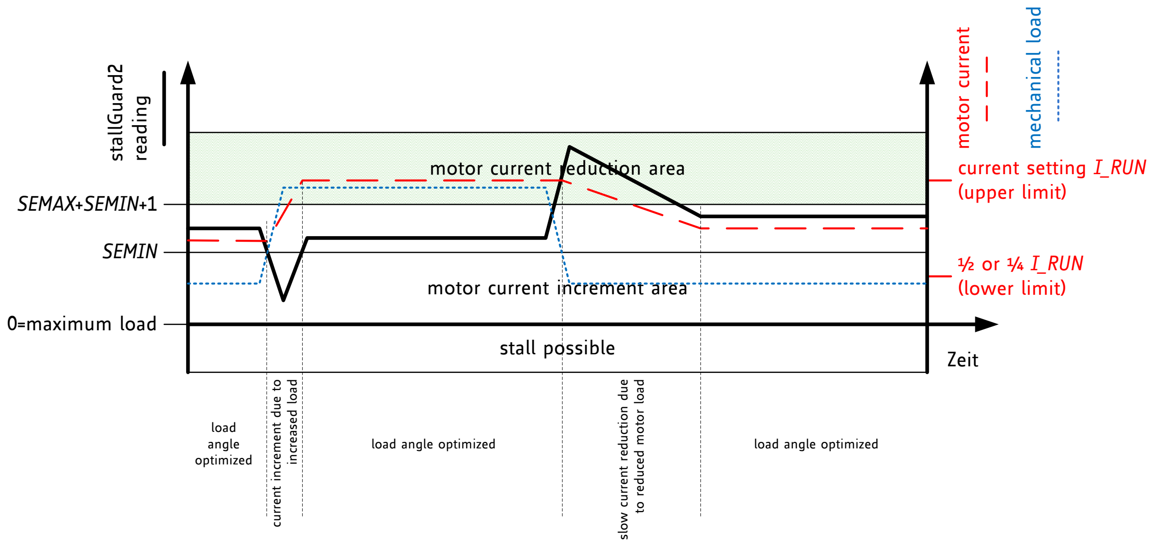



**Figure 11.1 coolStep adapts motor current to the load**

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

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

## 11.3 Tuning coolStep

Before tuning coolStep, first tune the stallGuard2 threshold level *SGT*, which affects the range of the load measurement value *SG*. coolStep uses *SG* to operate the motor near the optimum load angle of +90°.

The current increment speed is specified in *SEUP*, and the current decrement speed is specified in *SEDN*. They can be tuned separately because they are triggered by different events that may need different responses. The encodings for these parameters allow the coil currents to be increased much more quickly than decreased, because crossing the lower threshold is a more serious event that may require a faster response. If the response is too slow, the motor may stall. In contrast, a slow response to crossing the upper threshold does not risk anything more serious than missing an opportunity to save power.

| |
|---|
| coolStep operates between limits controlled by the current scale parameter *IRUN* and the *seimin* bit. |

### 11.3.1 Response Time

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

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

### 11.3.2 Low Velocity and Standby Operation

Because coolStep is not able to measure the motor load in standstill and at very low RPM, a lower velocity threshold is provided in the ramp generator. It should be set to an application specific default value. Below this threshold the normal current setting via *IRUN* respectively *IHOLD* is valid. An upper threshold is provided by the *VHIGH* setting. Both thresholds can be set as a result of the stallGuard2 tuning process.

# 12 Sine-Wave Look-up Table

Each of the TMC5031 drivers provides a programmable look-up table for storing the microstep current wave. As a default, the tables are pre-programmed with a sine wave, which is a good starting point for most stepper motors. Reprogramming the table to a motor specific wave allows drastically improved microstepping especially with low-cost motors.

## 12.1 User Benefits

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

## 12.2 Microstep Table

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

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

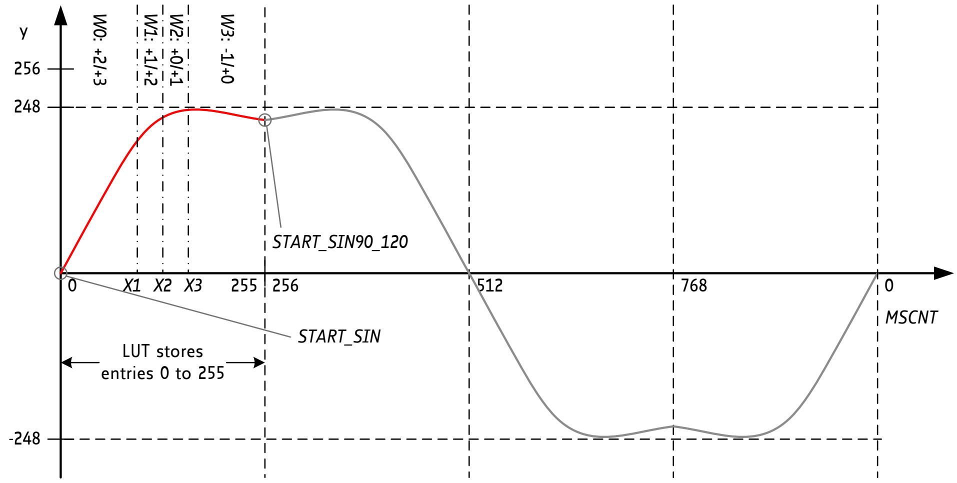



**Figure 12.1 LUT programming example**

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

Two registers control the starting values of the tables:

- As the starting value at zero is not necessarily 0 (it might be 1 or 2), it can be programmed into the starting point register *START_SIN*.
- In the same way, the start of the second wave for the second motor coil needs to be stored in *START_SIN90_120*. This register stores the resulting table entry for a phase shift of 90° for 2-phase stepper motors.

| Hints:<br>Refer chapter 5.3 for the register set and for the default table function stored in the drivers. The default table is a good base for realizing an own table.<br>The TMC5031-EVAL will come with a calculation tool for own waves. |
|---|

# 13 Clock Oscillator and Clock Input

The clock is the timing reference for all functions: the chopper, the velocity, the acceleration control, etc. Many parameters are scaled with the clock frequency, thus a precise reference allows a more deterministic result. The on-chip clock oscillator provides timing in case no external clock is easily available.

### Using the Internal Clock

Directly tie the CLK input to GND near to the TMC5031 if the internal clock oscillator is to be used. The internal clock can be calibrated by driving the ramp generator at a certain velocity setting. Reading out position values via the interface and comparing the resulting velocity to the remote masters' clock gives a time reference. This allows scaling acceleration and velocity settings as a result. The temperature dependency and ageing of the internal clock is comparatively low.

*In case well defined velocity settings and precise motor chopper operation are desired, it is supposed to work with an external clock source.*

### Using an External Clock

When an external clock is available, a frequency of 12 MHz to 16 MHz is recommended for optimum performance. The duty cycle of the clock signal is uncritical, as long as minimum high or low input time for the pin is satisfied (refer to electrical characteristics). Up to 18 MHz can be used, when the clock duty cycle is 50%. Make sure, that the clock source supplies clean CMOS output logic levels and steep slopes when using a high clock frequency. The external clock input is enabled with the first positive polarity seen on the CLK input.

*Attention*:
Switching off the external clock frequency prevents the driver from operating normally. Therefore be careful to switch off the motor drivers before switching off the clock (e.g. using the enable input), because otherwise the chopper would stop and the motor current level could rise uncontrolled. The short to GND detection stays active even without clock, if enabled.

## 13.1 Considerations on the Frequency

A higher frequency allows faster step rates, faster SPI operation and higher chopper frequencies. On the other hand, it may cause more electromagnetic emission of the system and causes more power dissipation in the TMC5031 digital core and voltage regulator. Generally a frequency of 12 MHz to 16 MHz should be sufficient for most applications. For reduced requirements concerning the motor dynamics, a clock frequency of down to 8 MHz can be considered.

# 14 Absolute Maximum Ratings

The maximum ratings may not be exceeded under any circumstances. Operating the circuit at or near more than one maximum rating at a time for extended periods shall be avoided by application design.

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Supply voltage | $V_{VS}$ | -0.5 | 18 | V |
| I/O supply voltage | $V_{VIO}$ | -0.5 | 5.5 | V |
| digital VCC supply voltage (if not supplied by internal regulator) | $V_{VCC}$ | -0.5 | 5.5 | V |
| Logic input voltage | $V_I$ | -0.5 | $V_{VIO}$+0.5 | V |
| Maximum current to / from digital pins and analog low voltage I/Os | $I_{IO}$ | | +/-10 | mA |
| 5V regulator output current (internal plus external load) | $I_{5VOUT}$ | | 50 | mA |
| 5V regulator continuous power dissipation ($V_{VM}$-5V) * $I_{5VOUT}$ | $P_{5VOUT}$ | | 1 | W |
| Power bridge repetitive output current ($T_J \leq 105°C$) | $I_{Ox}$ | | 2.0 | A |
| Power bridge repetitive output current ($T_J \leq 125°C$) | $I_{Ox}$ | | 1.5 | A |
| Power bridge repetitive output current ($T_J = 150°C$) | $I_{Ox}$ | | 0.8 | A |
| Junction temperature | $T_J$ | -50 | 150 | °C |
| Storage temperature | $T_{STG}$ | -55 | 150 | °C |
| ESD-Protection for interface pins (Human body model, HBM) | $V_{ESDAP}$ | | 4 (tbd.) | kV |
| ESD-Protection for handling (Human body model, HBM) | $V_{ESD}$ | | 1 (tbd.) | kV |

# 15 Electrical Characteristics

## 15.1 Operational Range

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Junction temperature | $T_J$ | -40 | 125 | °C |
| Supply voltage (using internal +5V regulator) | $V_{VS}$ | 5.5 | 16 | V |
| Supply voltage (internal +5V regulator bridged: $V_{VCC}$=$V_{VSA}$) | $V_{VS}$ | 4.7 | 5.4 | V |
| I/O supply voltage | $V_{VIO}$ | 3.00 | 5.25 | V |
| VCC voltage when using optional external source (supplies digital logic and charge pump) | $V_{VCC}$ | 4.75 | 5.25 | V |
| Peak output current per motor coil output (sine wave peak) | $I_{Ox}$ | | 1.1 | A |
| Peak output current per motor coil output (sine wave peak) Limit $T_J \leq 105°C$, e.g. with 50% duty cycle at 3s on / 3s off. | $I_{Ox}$ | | 1.5 | A |

## 15.2 DC Characteristics and Timing Characteristics

DC characteristics contain the spread of values guaranteed within the specified supply voltage range unless otherwise specified. Typical values represent the average value of all parts measured at +25°C. Temperature variation also causes stray to some values. A device with typical values will not leave Min/Max range within the full temperature range.

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

| **Motor driver section** | **DC- and Timing-Characteristics**<br>$V_{VS}$ = 16.0V | | | | | |
|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Conditions** | **Min** | **Typ** | **Max** | **Unit** |
| $RDS_{ON}$ lowside MOSFET | $R_{ONL}$ | measure at 100mA, 25°C, static state | | 0.4 | 0.5 | Ω |
| $RDS_{ON}$ highside MOSFET | $R_{ONH}$ | measure at 100mA, 25°C, static state | | 0.5 | 0.6 | Ω |
| slope, MOSFET turning on | $t_{SLPON}$ | measured at 700mA load current | | 120 | 250 | ns |
| slope, MOSFET turning off | $t_{SLPOFF}$ | measured at 700mA load current | | 220 | 450 | ns |
| Current sourcing, driver off | $I_{OIDLE}$ | $O_{XX}$ pulled to GND | 120 | 180 | 250 | µA |

| **Charge pump** | **DC-Characteristics** | | | | | |
|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Conditions** | **Min** | **Typ** | **Max** | **Unit** |
| Charge pump output voltage | $V_{VCP}$-$V_{VS}$ | operating, typical $f_{chop}$<40kHz | 4.0 | $V_{5VOUT}$ - 0.4 | $V_{5VOUT}$ | V |
| Charge pump voltage threshold for undervoltage detection | $V_{VCP}$-$V_{VS}$ | using internal 5V regulator voltage | 3.3 | 3.6 | 3.8 | V |
| Charge pump frequency | $f_{CP}$ | | | 1/16 $f_{CLKOSC}$ | | |

| **Linear regulator** | **DC-Characteristics** | | | | | |
|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Conditions** | **Min** | **Typ** | **Max** | **Unit** |
| Output voltage | $V_{5VOUT}$ | $I_{5VOUT}$ = 0mA<br>$T_J$ = 25°C | 4.75 | 5.0 | 5.25 | V |
| Output resistance | $R_{5VOUT}$ | Static load | | 3 | | Ω |
| Deviation of output voltage over the full temperature range | $V_{5VOUT(DEV)}$ | $I_{5VOUT}$ = 30mA<br>$T_J$ = full range | | 30 | 100 | mV |

| Clock oscillator and input | Timing-Characteristics | | | | | |
|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Conditions** | **Min** | **Typ** | **Max** | **Unit** |
| Clock oscillator frequency | $f_{CLKOSC}$ | $t_J$=-50°C | 8.8 | 12.4 | 17.9 | MHz |
| Clock oscillator frequency | $f_{CLKOSC}$ | $t_J$=50°C | 9.4 | 13.2 | 18.8 | MHz |
| Clock oscillator frequency | $f_{CLKOSC}$ | $t_J$=150°C | 9.6 | 13.4 | 18.9 | MHz |
| External clock frequency (operating) | $f_{CLK}$ | | 8 | 12-16 | 18 | MHz |
| External clock high / low level time | $t_{CLK}$ | CLK driven to 0.1 $V_{VIO}$ / 0.9 $V_{VIO}$ | 25 | | | ns |

| Detector levels | DC-Characteristics | | | | | |
|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Conditions** | **Min** | **Typ** | **Max** | **Unit** |
| $V_{VS}$ undervoltage threshold for RESET | $V_{UV}$ | $V_{VS}$ rising | 3.8 | 4.2 | 4.6 | V |
| $V_{5VOUT}$ undervoltage threshold for RESET | $V_{UV}$ | $V_{5VOUT}$ rising | | 3.5 | | V |
| Short to GND detector threshold ($V_{VSP}$ - $V_{Ox}$) | $V_{OS2G}$ | | 1.5 | 2.2 | 3 | V |
| Short to GND detector delay (high side switch on to short detected) | $t_{S2G}$ | High side output clamped to $V_{SP}$-3V | 0.8 | 1.3 | 2 | µs |
| Overtemperature prewarning | $t_{OTPW}$ | Temperature rising | 100 | 120 | 140 | °C |
| Overtemperature shutdown | $t_{OT}$ | Temperature rising | 135 | 150 | 170 | °C |

| Sense resistor voltage levels | DC-Characteristics | | | | | |
|---|---|---|---|---|---|---|
| **Parameter** | **Symbol** | **Conditions** | **Min** | **Typ** | **Max** | **Unit** |
| Sense input peak threshold voltage (low sensitivity) | $V_{SRTL}$ | *vsense*=0<br>*csactual*=31<br>*sin_x*=248<br>Hyst.=0; $I_{BRxy}$=0 | | 325 | | mV |
| sense input peak threshold voltage (high sensitivity) | $V_{SRTH}$ | *vsense*=1<br>*csactual*=31<br>*sin_x*=248<br>Hyst.=0; $I_{BRxy}$=0 | | 180 | | mV |
| Internal resistance from pin BRxy to internal sense comparator (additional to sense resistor) | $R_{BRxy}$ | | | 20 | | mΩ |

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

## 15.3 Thermal Characteristics

The following table shall give an idea on the thermal resistance of the QFN-48 package. The thermal resistance for a four layer board will provide a good idea on a typical application. The single layer board example is kind of a worst case condition, as the typical application will require a 4 layer board. Actual thermal characteristics will depend on the PCB layout, PCB type and PCB size.

A thermal resistance of 23°C/W for a typical board means, that the package is capable of continuously dissipating 4W at an ambient temperature of 25°C with the die temperature staying below 125°C.

| **Parameter** | **Symbol** | **Conditions** | **Typ** | **Unit** |
|---|---|---|---|---|
| Thermal resistance junction to ambient on a single layer board | $R_{TJA}$ | Single signal layer board (1s) as defined in JEDEC EIA JESD51-3 (FR4, 76.2mm x 114.3mm, d=1.6mm) | 80 | K/W |
| Thermal resistance junction to ambient on a multilayer board | $R_{TMJA}$ | Dual signal and two internal power plane board (2s2p) as defined in JEDEC EIA JESD51-5 and JESD51-7 (FR4, 76.2mm x 114.3mm, d=1.6mm) | 23 | K/W |
| Thermal resistance junction to ambient on a multilayer board with air flow | $R_{TMJA1}$ | Identical to $R_{TMJA}$, but with air flow 1m/s | 20 | K/W |
| Thermal resistance junction to board | $R_{TJB}$ | PCB temperature measured within 1mm distance to the package | 10 | K/W |
| Thermal resistance junction to case | $R_{TJC}$ | Junction temperature to heat slug of package | 3 | K/W |

The thermal resistance in an actual layout can be tested by checking for the heat up caused by the standby power consumption of the chip. When no motor is attached, all power seen on the power supply is dissipated within the chip.

*Note:*
A spread-sheet for calculating TMC5031 power dissipation is available on www.trinamic.com.

# 16 Layout Considerations

## 16.1 Exposed Die Pad

The TMC5031 uses its die attach pad to dissipate heat from the drivers and the linear regulator to the board. For best electrical and thermal performance, use a reasonable amount of solid, thermally conducting vias between the die attach pad and the ground plane. The printed circuit board should have a solid ground plane spreading heat into the board and providing for a stable GND reference.

## 16.2 Wiring GND

All signals of the TMC5031 are referenced to their respective GND. Directly connect all GND pins under the TMC5031 to a common ground area (GND, GNDP, GNDA and die attach pad). The GND plane right below the die attach pad should be treated as a virtual star point. For practical reasons, this has to be the PCB GND layer, not the PCB top layer.

*Attention*!
Especially, the sense resistors are susceptible to GND differences and GND ripple voltage, as the microstep current steps make up for voltages down to 0.5 mV. No current other than the sense resistor current should flow on their connections to GND and to the TMC5031. Optimally place them close to the TMC5031, with one or more vias to the GND plane for each sense resistor. The two sense resistors for one coil should not share a common ground connection trace or vias, as also PCB traces have a certain resistance.

## 16.3 Supply Filtering

The 5VOUT output voltage ceramic filtering capacitor (4.7 µF recommended) should be placed as close as possible to the 5VOUT pin, with its GND return going directly to the GNDA pin. Use as short and as thick connections as possible. For best microstepping performance and lowest chopper noise an additional filtering capacitor can be used for the VCC pin to GND, to avoid charge pump and digital part ripple influencing motor current regulation. Therefore place a ceramic filtering capacitor (470nF recommended) as close as possible (1-2mm distance) to the VCC pin with GND return going to the ground plane. VCC can be coupled to 5VOUT using a 2.2 Ω or 3.3 Ω resistor in order to supply the digital logic from 5VOUT while keeping ripple away from this pin.

A 100 nF filtering capacitor should be placed as close as possible to the VSA pin to ground plane. The motor supply pins VS should be decoupled with an electrolytic capacitor (47 µF or larger is recommended) and a ceramic capacitor, placed close to the device.

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

## 16.4 Layout Example

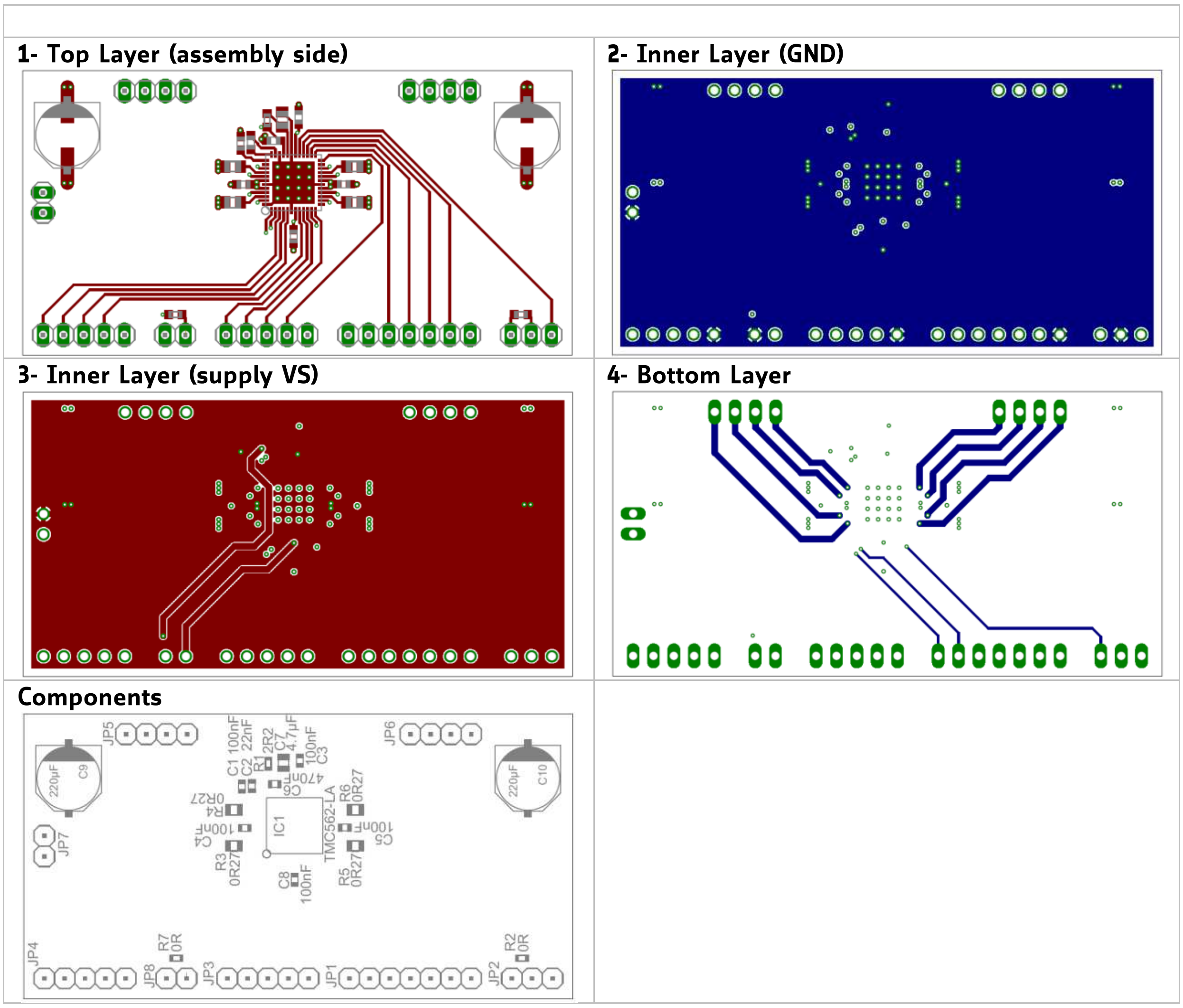



**Figure 16.1 Layout example**

# 17 Package Mechanical Data

## 17.1 Dimensional Drawings

*Attention: Drawings not to scale.*

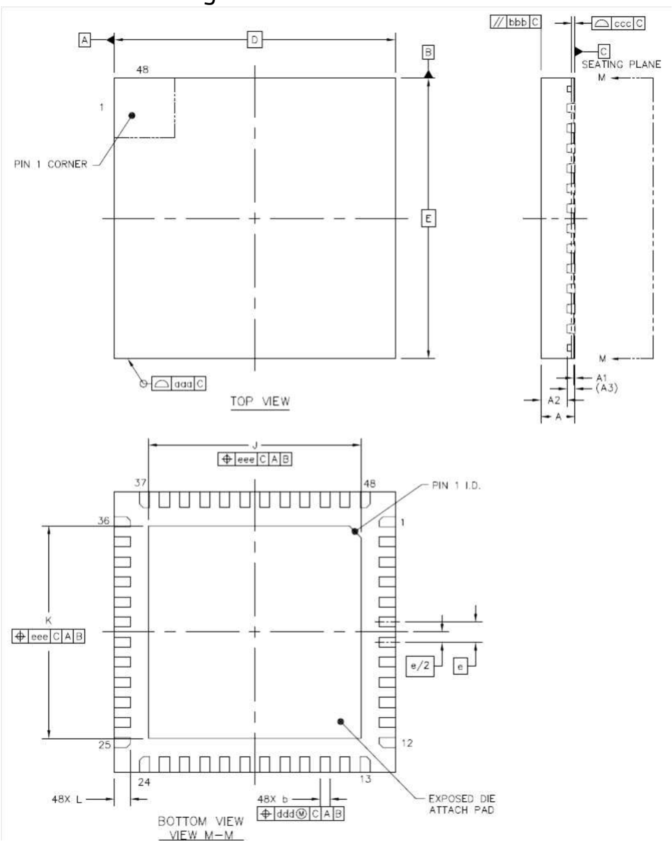



**Figure 17.1 Dimensional drawings**

| Parameter | Ref | Min | Nom | Max |
|---|---|---|---|---|
| total thickness | A | 0.80 | 0.85 | 0.90 |
| stand off | A1 | 0.00 | 0.035 | 0.05 |
| mold thickness | A2 | - | 0.65 | 0.67 |
| lead frame thickness | A3 | | 0.203 | |
| lead width | b | 0.2 | 0.25 | 0.3 |
| body size X | D | | 7.0 | |
| body size Y | E | | 7.0 | |
| lead pitch | e | | 0.5 | |
| exposed die pad size X | J | 5.2 | 5.3 | 5.4 |
| exposed die pad size Y | K | 5.2 | 5.3 | 5.4 |
| lead length | L | 0.35 | 0.4 | 0.45 |
| package edge tolerance | aaa | | | 0.1 |
| mold flatness | bbb | | | 0.1 |
| coplanarity | ccc | | | 0.08 |
| lead offset | ddd | | | 0.1 |
| exposed pad offset | eee | | | 0.1 |

## 17.2 Package Codes

| Type | Package | Temperature range | Code & marking |
|---|---|---|---|
| TMC5031 | QFN48 (RoHS) | -40°C ... +125°C | TMC5031-ES |

# 18 Getting Started

Please refer to the TMC5031-EVAL evaluation board to allow a quick start with the device, and in order to allow interactive tuning of the device setup in your application. It will guide you through the process of correctly setting up all registers. The following example gives a minimum set of accesses allowing moving a motor.

## 18.1 Initialization Examples

Initialization SPI datagram example sequence to enable and initialize driver 1 for operation:

```
SPI send: 0x8000000008;  // GCONF=8: Enable PP and INT outputs
SPI send: 0xEC00010445;  // CHOPCONF: TOFF=5, HSTRT=4, HEND=8, TBL=2, CHM=0 (spreadCycle)
SPI send: 0xB000011F05;  // IHOLD_IRUN: IHOLD=5, IRUN=31 (max. current), IHOLDDELAY=1
SPI send: 0xA600001388;  // AMAX=5000
SPI send: 0xA700004E20;  // VMAX=20000
SPI send: 0xA000000001;  // RAMPMODE=1 (positive velocity)

// Now motor 1 should start rotating

SPI send: 0x2100000000;  // Query X Actual – The next read access delivers X Actual
SPI read;                // Read X Actual
```

*The configuration parameters should be tuned to the motor and application for optimum performance.*

# 19 Disclaimer

TRINAMIC Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Motion Control GmbH & Co. KG. Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

Information given in this data sheet is believed to be accurate and reliable. However no responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties which may result from its use.

Specifications are subject to change without notice.

All trademarks used are property of their respective owners.

# 20 ESD Sensitive Device

The TMC5031 is an ESD sensitive CMOS device sensitive to electrostatic discharge. Take special care to use adequate grounding of personnel and machines in manual handling. After soldering the devices to the board, ESD requirements are more relaxed. Failure to do so can result in defect or decreased reliability.

# 21 Table of Figures

# 22 Revision History

| Version | Date | Author<br>BD – Bernhard Dwersteg<br>SD – Sonja Dwersteg | Description |
|---|---|---|---|
| 1.04 | 2012_NOV-18 | BD / SD | First version of product TMC5031 datasheet based on TMC562 prototype datasheet V1.04 |
| 1.05 | 2013_FEB-22 | JP | Product Image changed |
| 1.06 | 2013-MAR-25 | SD | - Chapter 15.3 (thermal characteristics) added.<br>- Chapter 10.1 (tuning the stallGuard2 threshold) updated.<br>- CSACTUAL in DRV_STATUS corrected (chapter 5.3.4).<br>- Interrupt output remark in *RAMP_STAT* for *status_latch_l* and *status_latch_r* removed. Description *event_stop_l* and *event_stop_r* updated (chapter 6.2.2.2)<br>- Description of the reference switch actions improved.<br>- *SW_MODE* register updated.<br>- Order codes updated.<br>- Consecutive numbering of the document corrected. |
| 1.07 | 2013-APR-30 | SD | New description of VCC_IO requirements. |

**Table 22.1 Documentation revisions**

# 23 References

[AN001] Trinamic Application Note 001 - Parameterization of spreadCycle™, www.trinamic.com
[AN002] Trinamic Application Note 002 - Parameterization of stallGuard2™ & coolStep™, www.trinamic.com