Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

TMC2041 DATASHEET

Dual step/direction driver for up to two 2-phase bipolar stepper motors. stallGuard for sensorless homing. SPI, UART (single wire) Configuration and Diagnostics Interface.

coolStep™

stallGuard2™

Applications

Office Automation
Antenna Positioning
3D printers
Battery powered applications
Printer and Scanner
Pumps and Valves
Medical Applications
Office and Laboratory equipment

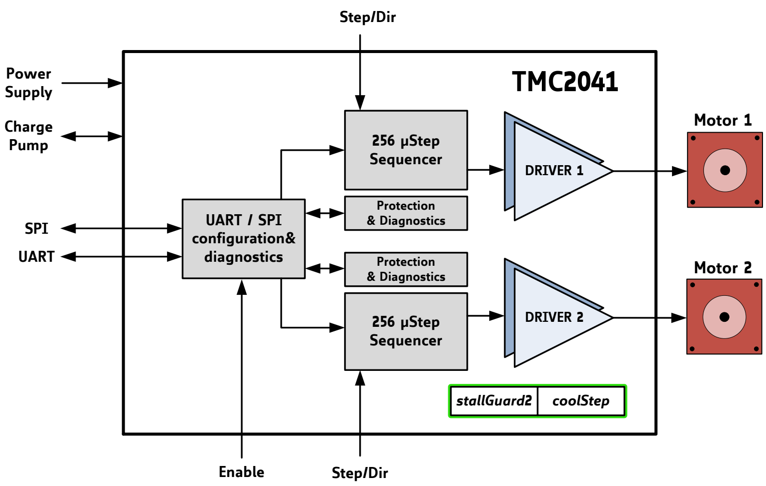
Features and Benefits

Two 2-phase stepper motors
Drive Capability up to 2x 1.1A coil current (2x 1.5A peak)
Parallel Option for one motor at 2.2A (3A peak)
Voltage Range 4.75… 26V DC
SPI & Single Wire UART for configuration and diagnostics
Highest Resolution up to 256 microsteps per full step
microPlyer™ microstep interpolation
spreadCycle™ highly dynamic motor control chopper
stallGuard2™ high precision sensorless motor load detection
coolStep™ current control for energy savings up to 75%
Full Protection & Diagnostics
Compact Size 7x7mm² QFN48 package

Description

The TMC2041 is a compact, dual stepper motor driver IC with serial interfaces for configuration and diagnostics. It is pin compatible to the fully featured TMC5041 and TMC5072 drivers with internal motion controller. The TMC2041 is intended for all applications, where an internal motion controller is not desired, and ramping is done in a microcontroller. Based on TRINAMICs high-performance spreadCycle chopper, the driver allows precise and smooth motor operation. It offers coolStep for energy savings and stallGuard for sensorless stall detection. The complete set of protection and diagnostic functionality ensures reliable operation. High integration, high energy efficiency and a small form factor enable miniaturized and scalable systems for cost effective solutions.

Block Diagram



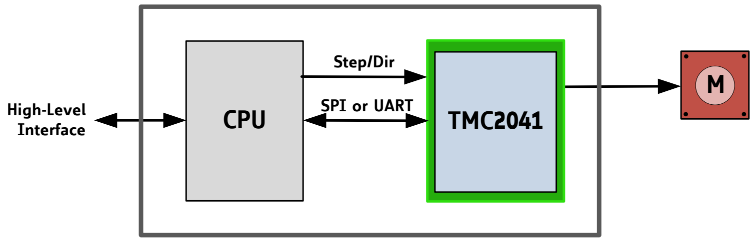
TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

TRINAMIC
MOTION CONTROL

APPLICATION EXAMPLES: HIGH FLEXIBILITY – MULTIPURPOSE USE

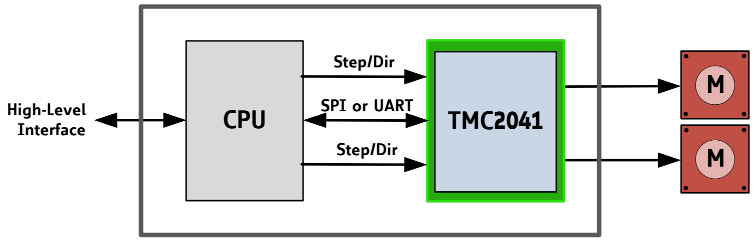
The TMC2041 scores with power density and sensorless homing. It features serial interfaces for advanced monitoring and configuration options. The small form factor keeps costs down and allows for miniaturized layouts. Extensive support at the chip, board, and software levels enables rapid design cycles and fast time-to-market with competitive products. High energy efficiency and reliability deliver cost savings in related systems such as power supplies and cooling.

Step/Dir for up to two Stepper Motors

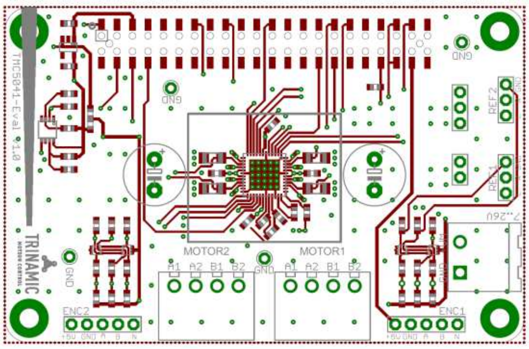


The stepper motor driver outputs are switched in parallel. This way, up to 2.2A RMS motors can be driven.

Step/Dir for up to two Stepper Motors



In this application, a single CPU controls two motors using a Step and Direction interface per motor. It initially configures the drivers by programming current settings and chopper, and run and hold current using either the 4 wire SPI interface, or the single wire UART interface. During operation the interface allows access to status information like stallGuard sensorless load measurement.



TMC2041-EVAL Evaluation Board

Evaluation & development platform

The TMC2041-EVAL is part of TRINAMICs universal evaluation board system which provides a convenient handling of the hardware as well as a user-friendly software tool for evaluation. The TMC2041 evaluation board system consists of three parts: STARTRAMPE (base board), ESELSBRÜCKE (connector board including several test points), and TMC2041-EVAL.

Order Codes

Order code	Description	Size [mm²]
TMC2041-LA	Dual axis step/dir driver, QFN-48	7 x 7
TMC2041-EVAL	Evaluation board for TMC2041	85 x 55
STARTRAMPE	Baseboard for TMC2041-EVAL and further evaluation boards	85 x 55
ESELSBRÜCKE	Connector board for plug-in evaluation board system	61 x 38

Table of Contents

1 Principles of Operation

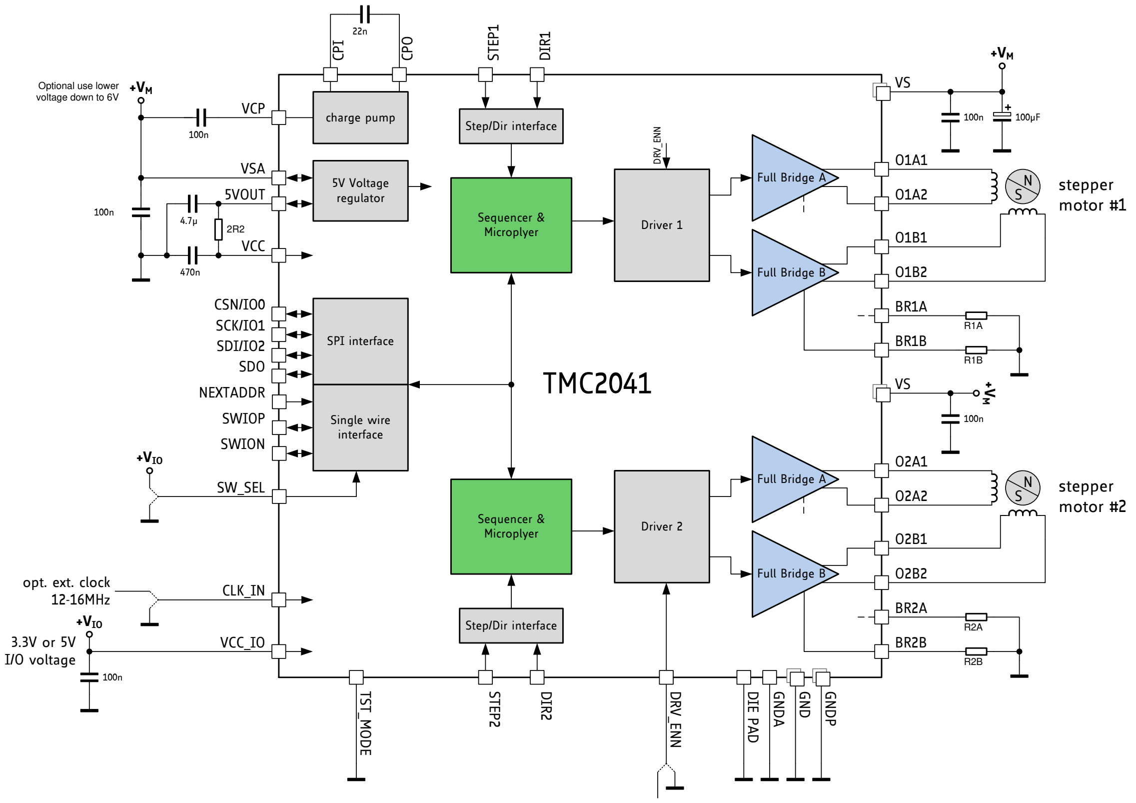


Figure 1.1 Basic application and block diagram

The TMC2041 driver chip is a highly integrated step & direction stepper driver for two stepper motors. The driver, chopper logic, and a 256 microstep sequencer are integrated into the TMC2041. It is pin compatible to the TMC5041 and TMC5072, which provide internal ramping. The TMC2041 offers a number of unique enhancements over similar products. It features automatic standstill current reduction and coolStep for enhanced motor efficiency and provides stallGuard2 for sensorless homing.

1.1 Key Concepts

The TMC2041 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.

spreadCycle™	High-precision chopper algorithm available as an alternative to the traditional constant off-time algorithm.
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%.

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

1.2 Control Interfaces

The TMC2041 supports both, an SPI and a UART based single wire interface with CRC checking. Selection of the actual interface is done via the configuration pin SW_SEL, which can be hardwired to GND or VCC_IO depending on the desired interface. From a software point of view the TMC2041 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.

1.2.1 SPI 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 TMC2041 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.

1.2.2 UART Interface

The single wire interface allows differential operation similar to RS485 (using SWIOP and SWION) or single wire interfacing (leaving open SWION). It can be driven by any standard UART. No baud rate configuration is required. An optional ring mode allows chaining of slaves to optimize interfacing for applications with regularly distributed drives.

1.3 Moving and Controlling the Motor

1.3.1 STEP/DIR Interface

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

1.4 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.

1.5 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.

Benefits are:

- *Energy efficiency*	power consumption decreased up to 75%
- *Motor generates less heat*	improved mechanical precision
- *Less or no cooling*	improved reliability
- *Use of smaller motor*	less torque reserve required → cheaper motor does the job

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.

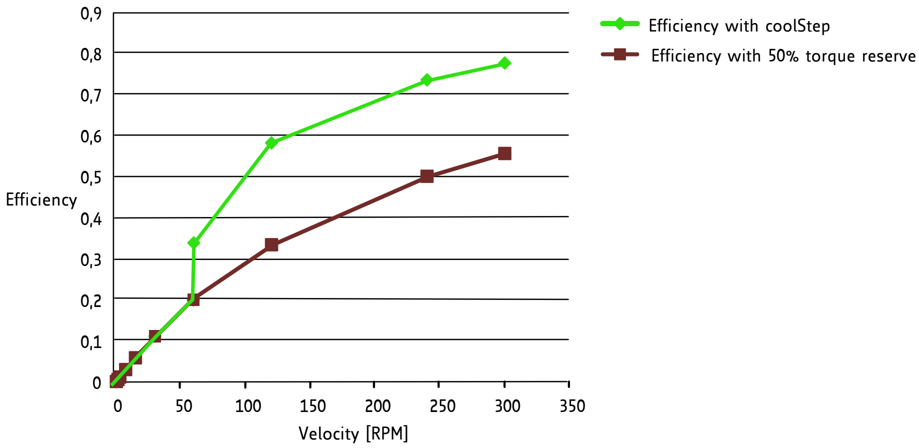


Figure 1.2 Energy efficiency with coolStep (example)

2 Pin Assignments

2.1 Package Outline

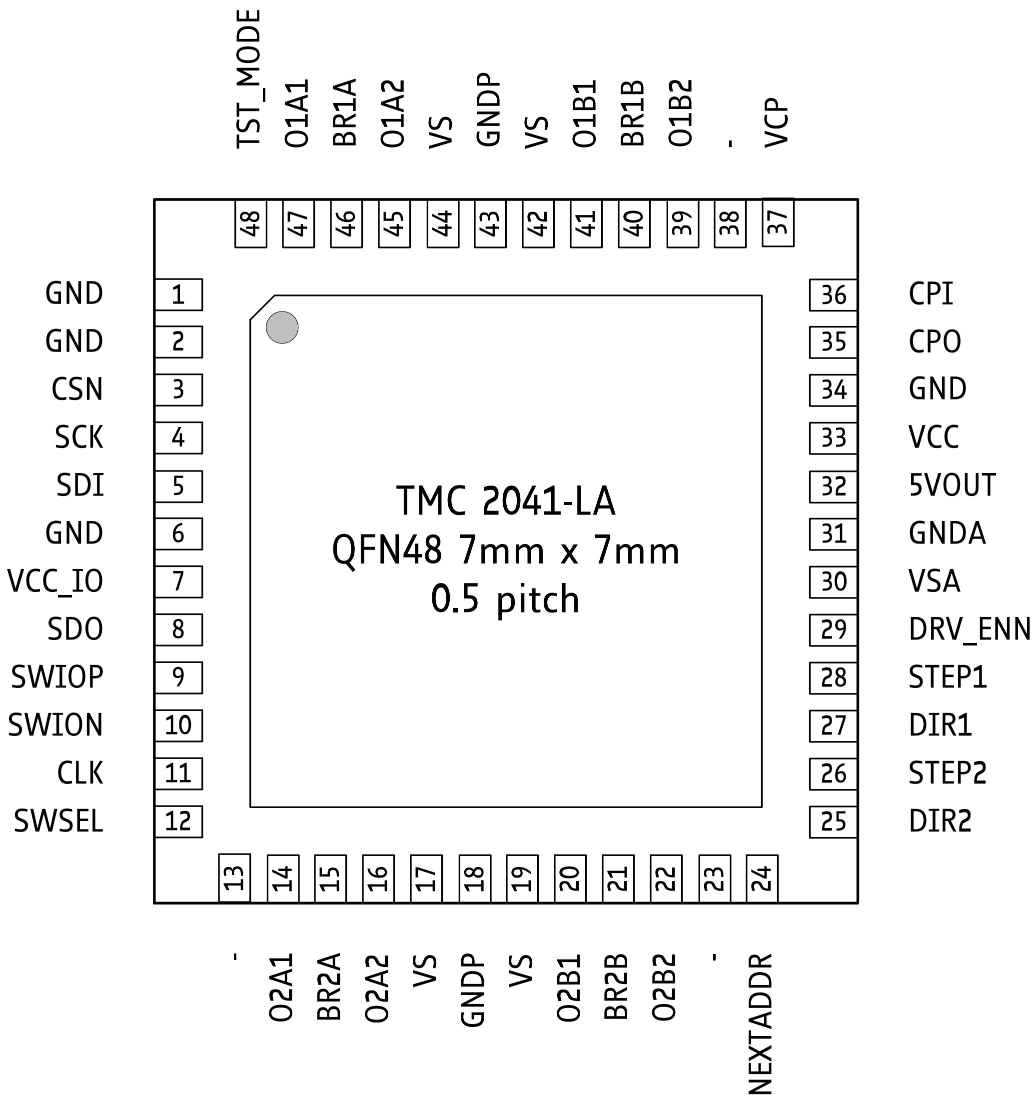


Figure 2.1 TMC2041 pin assignments.

2.2 Signal Descriptions

Pin	Number	Type	Function
GND	6, 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 pins.
VSA	30		Analog supply voltage for 5V regulator – typically supplied with driver supply voltage. An additional 100nF capacitor to GND (GND plane) is recommended for best performance.
GNDA	31	GND	Analog GND. Tie to GND plane.
5VOUT	32		Output of internal 5V regulator. Attach 2.2µF or larger ceramic capacitor to GNDA near to pin for best performance. Use to supply VCC of chip.

Pin	Number	Type	Function
VCC	33		5V supply input for digital circuitry within chip and charge pump. Attach 470nF capacitor to GND (GND plane). Typically supplied by 5VOUT. A 2.2Ω resistor is recommended for decoupling noise from 5VOUT. When using an external supply, make sure, that VCC comes up before or in parallel to 5VOUT or VCC_IO, whichever comes up later!
DIE_PAD	-	GND	Connect the exposed die pad to a GND plane. Provide as many as possible vias for heat transfer to GND plane.

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 22nF or 33nF / 50V capacitor to CPO.
VCP	37		Output of charge pump. Provide external 100nF capacitor to VS.

Table 2.2 Charge pump pins

Pin	Number	Type	Function
GND	1	I	unused input, tie to GND
GND	2	I	unused input, tie to GND
CSN/IO0	3	I/O	Chip select input of SPI interface, programmable IO in UART mode
SCK/IO1	4	I/O	Serial clock input of SPI interface, programmable IO in UART mode
SDI/IO2	5	I/O	Data input of SPI interface, programmable IO in UART mode
SDO	8	I/O	Data output of SPI interface (Tristate, enabled with CSN=0), mode configuration input in UART mode (0 = Normal mode, 1 = Single wire ring mode – SWIO_P is input, SWIO_N is output)
SWIOP	9	I/O	Single wire I/O (positive). Serial input in ring mode. Multi-purpose input in SPI mode or encoder 1 N input.
SWION	10	I/O	Single wire I/O (negative) for differential mode. Leave open in non-differential mode when operating at 5V IO voltage or tie to desired threshold voltage. Serial output in ring mode. Multi-purpose input in SPI mode or encoder 2 N input.
CLK	11	I	Clock input. Tie to GND using short wire for internal clock or supply external clock. The first high signal disables the internal oscillator until power down.
SWSEL	12	I	Interface selection input. Tie to GND for SPI mode, tie to VCC_IO for single wire (UART) interface mode.
NEXTADDR	24	I	Address increment (if tied high) for single wire (UART) mode. General purpose input in SPI mode
DIR2	25	I	Right reference switch input for motor 2, optional DIR input for STEP/DIR operation of motor 2 or encoder 2 B input
STEP2	26	I	Left reference switch input for motor 2, optional STEP input for STEP/DIR operation of motor 2
DIR1	27	I	Right reference switch input for motor 1, optional DIR input for STEP/DIR operation of motor 1 or encoder 2 A input
STEP1	28	I	Left reference switch input for motor 1, optional STEP input for STEP/DIR operation of motor 1
DRV_ENN	29	I	Enable input for motor drivers. The power stage becomes switched off (all motor outputs floating) when this pin becomes driven to a high level. Tie to GND for normal operation.
TST_MODE	48	I	Test mode input. Tie to GND using short wire.
-	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 coil A output 1
BR2A	15		Sense resistor connection for motor 2 coil A. Place sense resistor to GND near pin.
O2A2	16	O (VS)	Motor 2 coil A output 2
VS	17, 19		Motor supply voltage. Provide filtering capacity near pin with shortest loop to nearest GNDP pin (respectively via GND plane).
GNDP	18	GND	Power GND. Connect to GND plane near pin.
O2B1	20	O (VS)	Motor 2 coil B output 1
BR2B	21		Sense resistor connection for motor 2 coil B. Place sense resistor to GND near pin.
O2B2	22	O (VS)	Motor 2 coil B output 2
O1B2	39	O (VS)	Motor 1 coil B output 2
BR1B	40		Sense resistor connection for motor 1 coil B. Place sense resistor to GND near pin.
O1B1	41	O (VS)	Motor 1 coil B output 1
VS	42, 44		Motor supply voltage. Provide filtering capacity near pin with shortest loop to nearest GNDP pin (respectively via GND plane).
GNDP	43	GND	Power GND. Connect to GND plane near pin.
O1A2	45	O (VS)	Motor 1 coil A output 2
BR1A	46		Sense resistor connection for motor 1 coil A. Place sense resistor to GND near pin.
O1A1	47	O (VS)	Motor 1 coil A output 1

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 connection of the bus interface and further digital signals is left out for clarity.

3.1 Standard Application Circuit

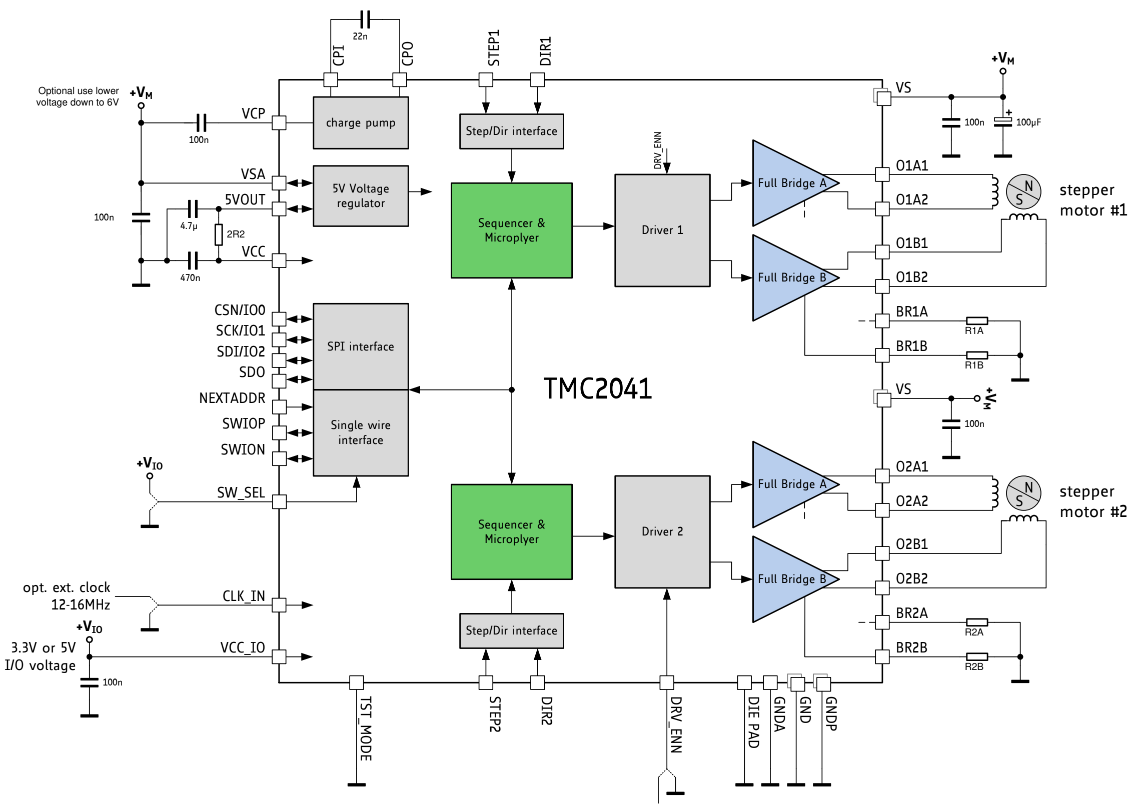


Figure 3.1 Standard application circuit

The standard application circuit uses a minimum set of additional components in order to operate the motor. Use low ESR capacitors for filtering the power supply which are capable to cope with the current ripple. The current ripple often depends on the power supply and cable length. The VCC_IO voltage can be supplied from 5VOUT, or from an external source, e.g. a low drop 3.3V regulator. In order to minimize linear voltage regulator power dissipation of the internal 5V voltage regulator in applications where VM is high, a different (lower) supply voltage can be used for VSA, if available. For example, many applications provide a 12V supply in addition to a higher supply voltage like 24V. Using the 12V supply for VSA will reduce the power dissipation of the internal 5V regulator to about 37% of the dissipation caused by supply with the full motor voltage. For best motor chopper performance, an optional R/C-filter de-couples 5VOUT from digital noise cause by power drawn from VCC.

Basic layout hints
Place sense resistors and all filter capacitors as close as possible to the related IC pins. Use a solid common GND for all GND connections, also for sense resistor GND. Connect 5VOUT filtering capacitor directly to 5VOUT and GNDA pin. See layout hints for more details. Low ESR electrolytic capacitors are recommended for VS filtering.

Attention
In case VSA is supplied by a different voltage source, make sure that VSA does not exceed VS by more than one diode drop upon power up or power down.

3.2 5 V Only Supply

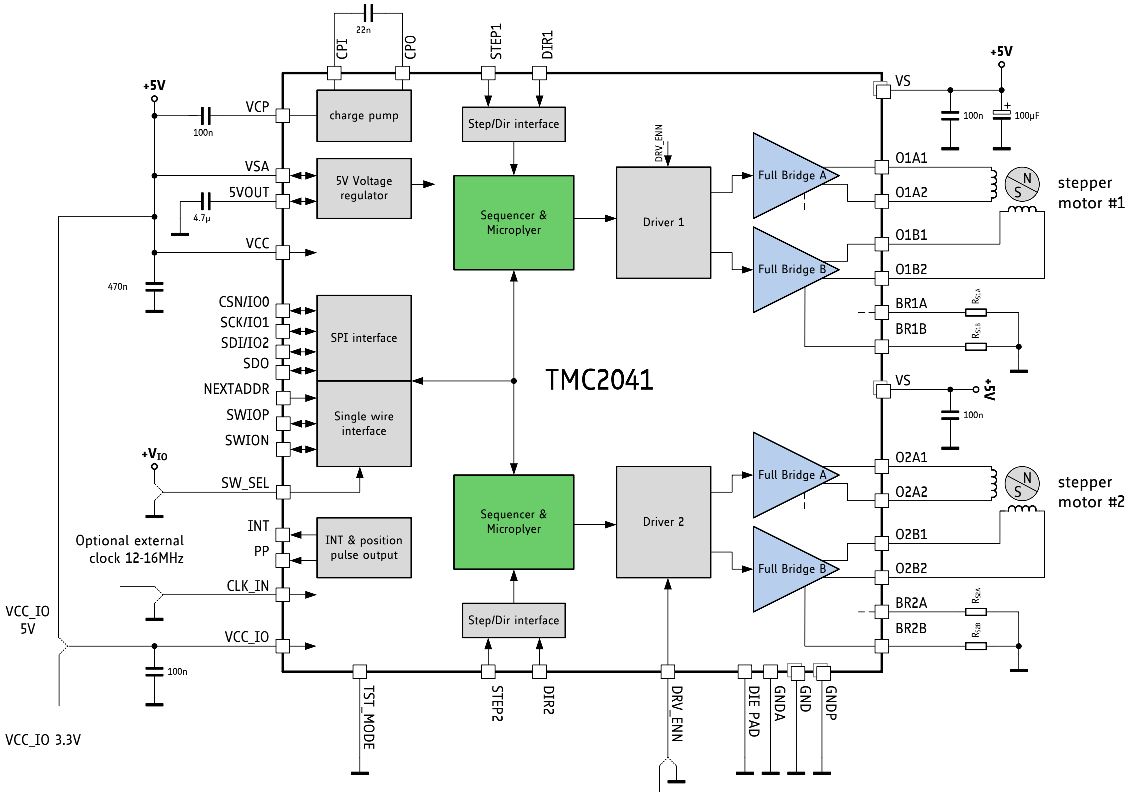


Figure 3.2 5V only operation

While the standard application circuit is limited to roughly 5.5V lower supply voltage, a 5V only application lets the IC run from a normal 5V +/-5% supply. In this application, linear regulator drop must be minimized. Therefore, the major 5V 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 One Motor with High Current

The TMC2041 supports double motor current for a single driver by paralleling both power stages. In order to operate in this mode, activate the flag *single_driver* in the global configuration register *GCONF*. This register can be locked for subsequent write access.

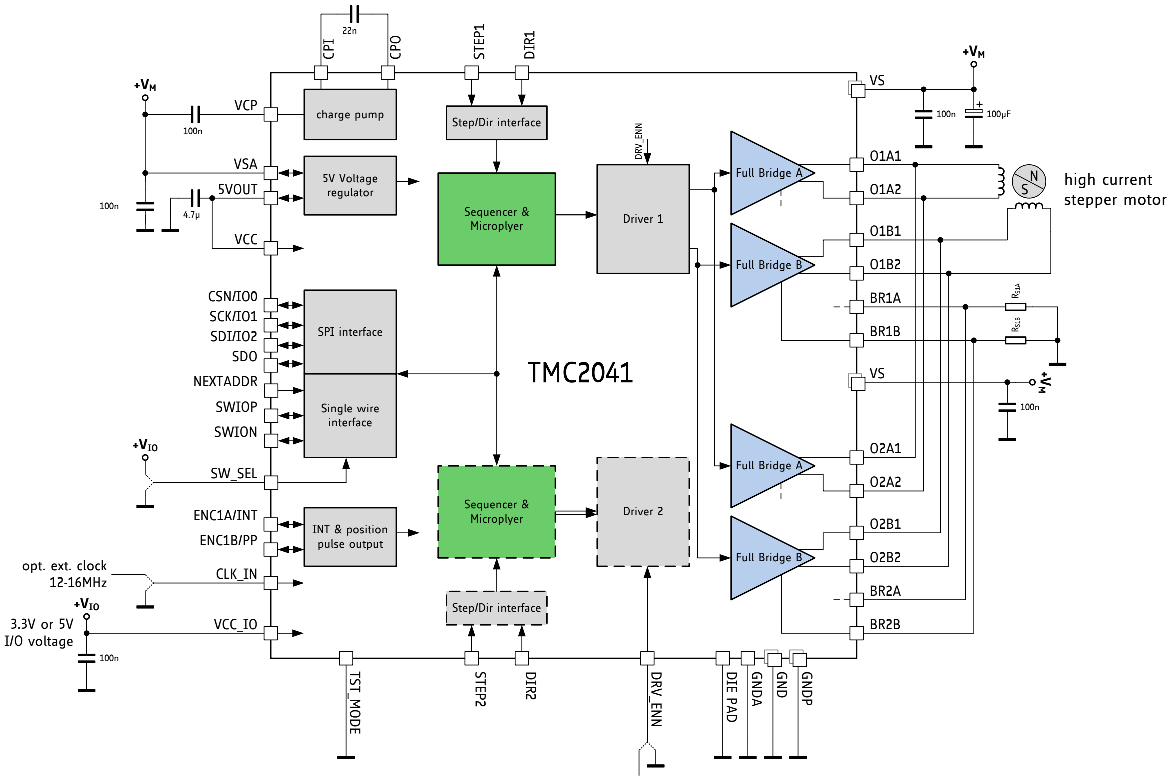


Figure 3.3 Driving a single motor with high current

3.4 External 5V Power Supply

When an external 5V power supply is available, the power dissipation caused by the internal linear regulator can be eliminated. This especially is beneficial in high voltage applications, and when thermal conditions are critical.

3.4.1 Internal Regulator Bridged

In case a clean external 5V supply is available, it can be used for complete supply of analog and digital part (Figure 3.4). The circuit will benefit from a well regulated supply, e.g. when using a +/-1% regulator. A precise supply guarantees increased motor current precision, because the voltage at 5VOUT directly is the reference voltage for all internal units of the driver, especially for motor current control. For best performance, the power supply should have low ripple to give a precise and stable supply at 5VOUT pin with remaining ripple well below 5mV. Some switching regulators have a higher remaining ripple, or different loads on the supply may cause lower frequency ripple. In this case, increase capacity attached to 5VOUT. In case the external supply voltage has poor stability or low frequency ripple, this would affect the precision of the motor current regulation as well as add chopper noise.

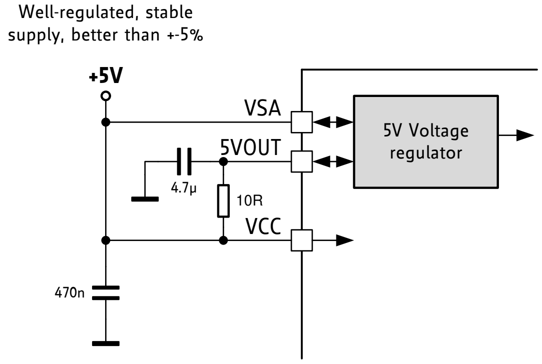


Figure 3.4 Using an external 5V supply to bypass internal regulator

3.5 Optimizing Analog Precision

The 5VOUT pin is used as an analog reference for operation of the TMC2041. Performance will degrade when there is voltage ripple on this pin. Most of the high frequency ripple in a TMC2041 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. The voltage drop on this resistor will be roughly 100 mV (I_{VCC} * R).

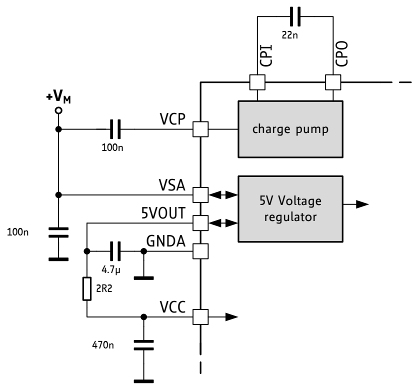


Figure 3.5 RC-Filter on VCC for reduced ripple

3.6 Driver Protection and EME Circuitry

Some applications have to cope with ESD events caused by motor operation or external influence. Despite ESD circuitry within the driver chips, ESD events occurring during operation can cause a reset or even a destruction of the motor driver, depending on their energy. Especially plastic housings and belt drive systems tend to cause ESD events. It is best practice to avoid ESD events by attaching all conductive parts, especially the motors themselves to PCB ground, or to apply electrically conductive plastic parts. In addition, the driver can be protected up to a certain degree against ESD events or live plugging / pulling the motor, which also causes high voltages and high currents into the motor connector terminals. A simple scheme uses capacitors at the driver outputs to reduce the dV/dt caused by ESD events. Larger capacitors will bring more benefit concerning ESD suppression, but cause additional current flow in each chopper cycle, and thus increase driver power dissipation, especially at high supply voltages. The values shown are example values – they might be varied between 100pF and 1nF. The capacitors also dampen high frequency noise injected from digital parts of the circuit and thus reduce electromagnetic emission. A more elaborate scheme uses LC filters to de-couple the driver outputs from the motor connector. Varistors in between of the coil terminals eliminate coil overvoltage caused by live plugging. Optionally protect all outputs by a varistor against ESD voltage.

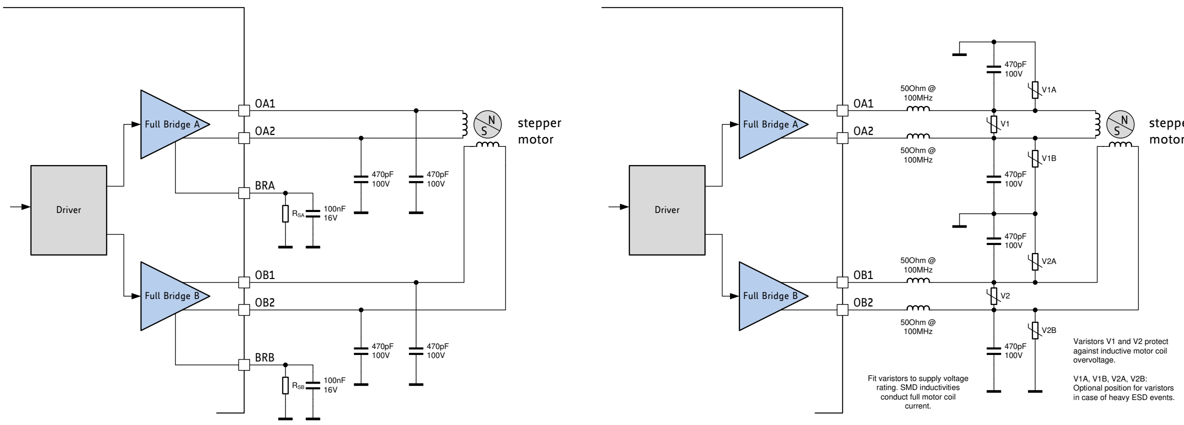


Figure 3.6 Simple ESD enhancement and more elaborate motor output protection

4 SPI Interface

4.1 SPI Datagram Structure

The TMC2041 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 TMC2041 must be handled in a way, that it stays active (low) for the complete duration of the datagram transmission.

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

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

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

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

<table>
<tr><th colspan="40">SPI DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="14">MSB (transmitted first)</td><td colspan="12">40 bit</td><td colspan="14">LSB (transmitted last)</td></tr>
<tr><td colspan="20">39 …</td><td colspan="20">… 0</td></tr>
<tr><td colspan="8">→ 8 bit address
← 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 TMC2041:
RW + 7 bit address
← from TMC2041:
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 TMC2041, 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 (*XACTUAL*) with the address 0x21, the address byte has to be set to 0x21 in the access preceding the read access. For a write access to the register (*VACTUAL*), the address byte has to be set to 0x80 + 0x22 = 0xA2. For read access, the data bit might have any value (-). So, one can set them to 0.

action	data sent to TMC2041	data received from TMC2041
read *CHOPCONF1*	→ `0x6C00000000`	← `0xSS & unused data`
read *CHOPCONF1*	→ `0x6C00000000`	← `0xSS & CHOPCONF1`
write CHOPCONF1:= `0x00ABCDEF`	→ `0xEC00ABCDEF`	← `0xSS & CHOPCONF1`
write *CHOPCONF1*:= `0x00123456`	→ `0xEC00123456`	← `0xSS00ABCDEF`

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

4.1.2 SPI Status Bits Transferred with Each Datagram Read Back

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

***SPI_STATUS* – status flags transmitted with each SPI access in bits 39 to 32**		
Bit	**Name**	**Comment**
7	-	reserved (0)
6	-	reserved (0)
5	-	reserved (0)
4	-	reserved (0)
3	-	reserved (0)
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 TMC2041 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 TMC2041.

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.

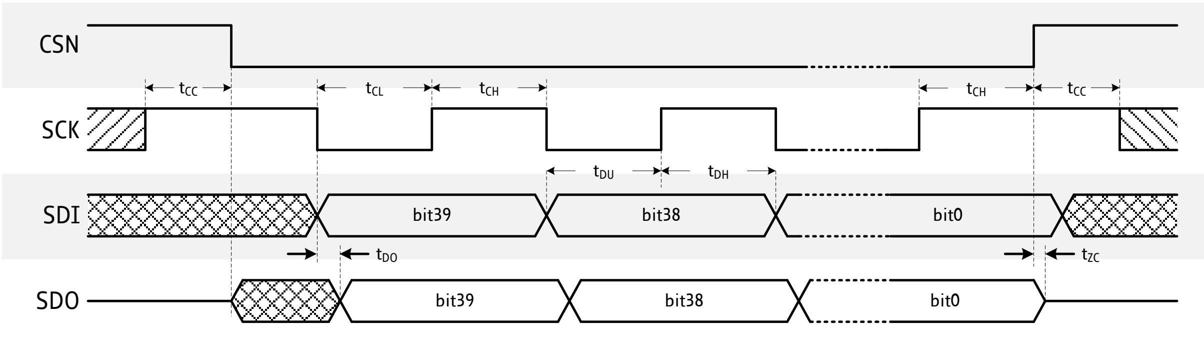


Figure 4.1 SPI timing

Hint
Usually this SPI timing is referred to as SPI MODE 3

SPI interface timing	**AC-Characteristics** 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 UART Single Wire Interface

The UART single wire interface allows the control of the TMC2041 with any microcontroller UART. It shares transmit and receive line like an RS485 based interface. Data transmission is secured using a cyclic redundancy check, so that increased interface distances (e.g. over cables between two PCBs) can be bridged without the danger of wrong or missed commands even in the event of electro-magnetic disturbance. The automatic baud rate detection and an advanced addressing scheme make this interface easy and flexible to use.

5.1 Datagram Structure

5.1.1 Write Access

<table>
<tr><th colspan="20">UART WRITE ACCESS DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="20">each byte is LSB…MSB, highest byte transmitted first</td></tr>
<tr><td colspan="20">0 … 63</td></tr>
<tr><th colspan="8">sync + reserved</th><th colspan="3">8 bit slave address</th><th colspan="3">RW + 7 bit register addr.</th><th colspan="3">32 bit data</th><th colspan="3">CRC</th></tr>
<tr><td colspan="8">0…7</td><td colspan="3">8…15</td><td colspan="3">16…23</td><td colspan="3">24…55</td><td colspan="3">56…63</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td colspan="4">Reserved (don't cares but included in CRC)</td><td colspan="3">SLAVEADDR</td><td colspan="2">register address</td><td>1</td><td colspan="3">data bytes 3, 2, 1, 0 (high to low byte)</td><td colspan="3">CRC</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>…</td><td>15</td><td>16</td><td>…</td><td>23</td><td>24</td><td>…</td><td>55</td><td>56</td><td>…</td><td>63</td></tr>
</table>

A sync nibble precedes each transmission to and from the TMC2041 and is embedded into the first transmitted byte, followed by an addressing byte. Each transmission allows a synchronization of the internal baud rate divider to the master clock. The actual baud rate is adapted and variations of the internal clock frequency are compensated. Thus, the baud rate can be freely chosen within the valid range. Each transmitted byte starts with a start bit (logic 0, low level on SWIOP) and ends with a stop bit (logic 1, high level on SWIOP). The bit time is calculated by measuring the time from the beginning of start bit (1 to 0 transition) to the end of the sync frame (1 to 0 transition from bit 2 to bit 3). All data is transmitted byte wise. The 32 bit data words are transmitted with the highest byte first.

A minimum baud rate of 9000 baud is permissible, assuming 20 MHz clock (worst case for low baud rate). Maximum baud rate is $f_{CLK}/16$ due to the required stability of the baud clock.

The slave address is determined by the register *SLAVEADDR*. If the external address pin NEXTADDR is set, the slave address becomes incremented by one.

The communication becomes reset if a pause time of longer than 63 bit times between the start bits of two successive bytes occurs. This timing is based on the last correctly received datagram. In this case, the transmission needs to be restarted after a failure recovery time of minimum 12 bit times of bus idle time. This scheme allows the master to reset communication in case of transmission errors. Any pulse on an idle data line below 16 clock cycles will be treated as a glitch and leads to a timeout of 12 bit times, for which the data line must be idle. Other errors like wrong CRC are also treated the same way. This allows a safe re-synchronization of the transmission after any error conditions. Remark, that due to this mechanism, an abrupt reduction of the baud rate to less than 15 percent of the previous value is not possible.

Each accepted write datagram becomes acknowledged by the receiver by incrementing an internal cyclic datagram counter (8 bit). Reading out the datagram counter allows the master to check the success of an initialization sequence or single write accesses. Read accesses do not modify the counter.

5.1.2 Read Access

<table>
<tr><th colspan="17">UART READ ACCESS REQUEST DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="17">each byte is LSB…MSB, highest byte transmitted first</td></tr>
<tr><th colspan="8">sync + reserved</th><th colspan="3">8 bit slave address</th><th colspan="3">RW + 7 bit register address</th><th colspan="3">CRC</th></tr>
<tr><td colspan="8">0…7</td><td colspan="3">8…15</td><td colspan="3">16…23</td><td colspan="3">24…31</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td colspan="4">Reserved (don't cares but included in CRC)</td><td colspan="3">SLAVEADDR</td><td colspan="2">register address</td><td>0</td><td colspan="3">CRC</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>…</td><td>15</td><td>16</td><td>…</td><td>23</td><td>24</td><td>…</td><td>31</td></tr>
</table>

The read access request datagram structure is identical to the write access datagram structure, but uses a lower number of user bits. Its function is the addressing of the slave and the transmission of the desired register address for the read access. The TMC2041 responds with the same baud rate as the master uses for the read request.

In order to ensure a clean bus transition from the master to the slave, the TMC2041 does not immediately send the reply to a read access, but it uses a programmable delay time after which the first reply byte becomes sent following a read request. This delay time can be set in multiples of eight bit times using *SENDDELAY* time setting (default=8 bit times) according to the needs of the master. In a multi-slave system, set *SENDDELAY* to min. 2 for all slaves. Otherwise a non-addressed slave might detect a transmission error upon read access to a different slave.

<table>
<tr><th colspan="23">UART READ ACCESS REPLY DATAGRAM STRUCTURE</th></tr>
<tr><td colspan="23">each byte is LSB…MSB, highest byte transmitted first</td></tr>
<tr><td colspan="23">0 …… 63</td></tr>
<tr><th colspan="8">sync + reserved</th><th colspan="3">8 bit slave address</th><th colspan="3">RW + 7 bit register addr.</th><th colspan="3">32 bit data</th><th colspan="3">CRC</th></tr>
<tr><td colspan="8">0…7</td><td colspan="3">8…15</td><td colspan="3">16…23</td><td colspan="3">24…55</td><td colspan="3">56…63</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td colspan="4">reserved (0)</td><td colspan="3">0xFF</td><td colspan="2">register address</td><td>0</td><td colspan="3">data bytes 3, 2, 1, 0 (high to low byte)</td><td colspan="3">CRC</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>…</td><td>15</td><td>16</td><td>…</td><td>23</td><td>24</td><td>…</td><td>55</td><td>56</td><td>…</td><td>63</td></tr>
</table>

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

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

ERRATA IN READ ACCESS

A known bug in the UART interface implementation affects read access to registers that change during the access. While the SPI interface takes a snapshot of the read register before transmission, the UART interface transfers the register directly MSB to LSB without taking a snapshot. This may lead to inconsistent data when reading out a register that changes during the transmission. Further, the CRC sent from the driver may be incorrect in this case (but must not), which will lead to the master repeating the read access. As a workaround, it is advised not to read out quickly changing registers like *XACTUAL*, *MSCNT* or *X_ENC* during a motion, but instead first stop the motor or check the *position_reached* flag to become active, and read out these values afterwards. If possible, use *X_LATCH* and *ENC_LATCH* for a safe readout during motion (e.g. for homing). As the encoder cannot be guaranteed to stand still during motor stop, only a dual read access and check for identical result ensures correct *X_ENC* read data. Therefore it is advised to use the latching function instead. Use the *vzero* and *velocity_reached* flag rather than reading *VACTUAL*.

5.2 CRC Calculation

An 8 bit CRC polynomial is used for checking both read and write access. It allows detection of up to eight single bit errors. The CRC8-ATM polynomial with an initial value of zero is applied LSB to MSB, including the sync- and addressing byte. The sync nibble is assumed to always be correct. The TMC2041 responds only to correctly transmitted datagrams containing its own slave address. It increases its datagram counter for each correctly received write access datagram.

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

SERIAL CALCULATION EXAMPLE

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

C-CODE EXAMPLE FOR CRC CALCULATION

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

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

5.3 UART Signals

The UART interface on the TMC2041 has following signals:

TMC2041 UART INTERFACE SIGNALS	
SWIOP	Non-inverted data input and output
SWION	Inverted data input and output for use in differential transmission. Can be left open in a 5V IO voltage system. Tie to the half IO level voltage for best performance in a 3.3V single wire non-differential application.
NEXTADDR	Address increment pin for sequential addressing scheme
SDO	Configuration input: Tie low for standard mode!

In UART mode (SW_SEL high) the slave checks the single wire SWIOP and SWION for correctly received datagrams with its own address continuously. Both signals are switched as input during this time. It adapts to the baud rate based on the sync nibble, as described before. In case of a read access, it switches on its output drivers on SWIOP and SWION and sends its response using the same baud rate.

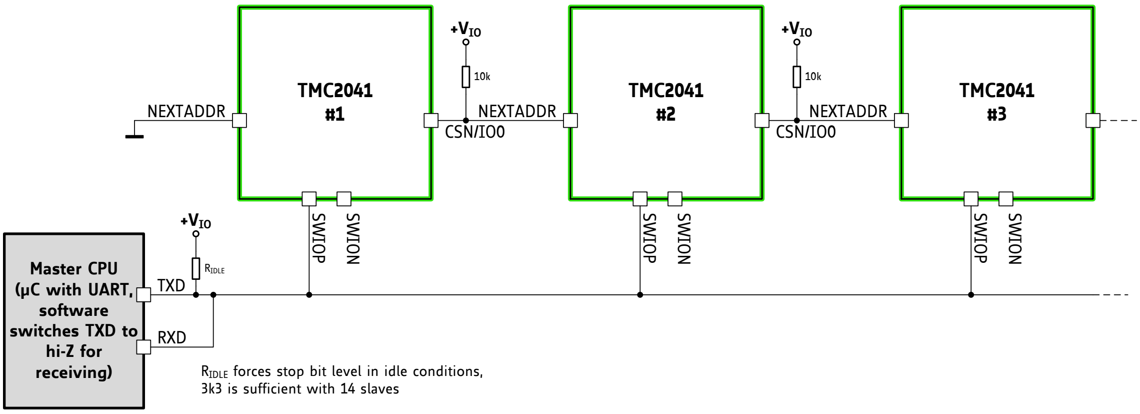
5.4 Addressing Multiple Slaves

Addressing one or two Slaves

If only one or two TMC2041 are addressed by a master using a single UART interface, a hardware address selection can be done by *setting the NEXTADDR pins to different levels*.

Addressing up to 255 Slaves

A different approach can address any number of devices by *using the input NEXTADDR as a selection pin*. Addressing up to 255 units is possible.



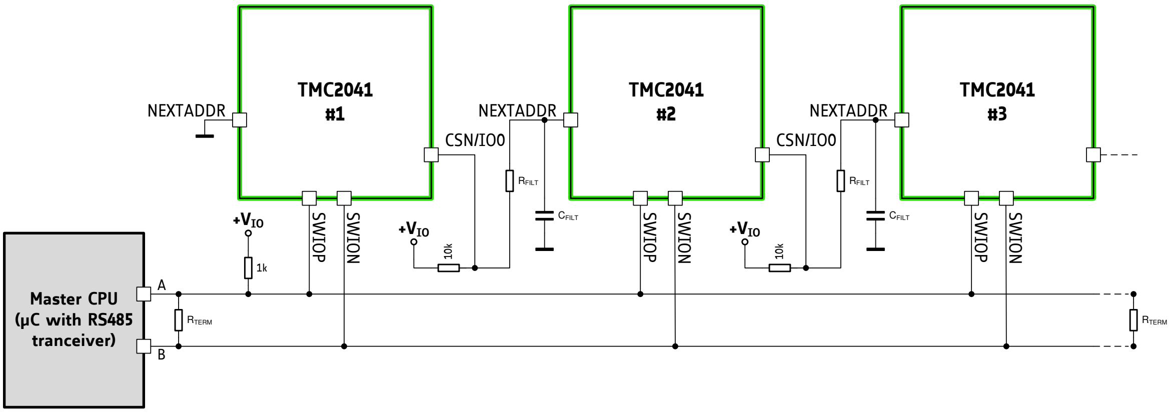
Example for addressing up to 255 TMC2041

Addressing phase 1:	address 0, IO0 is high-Z	address 1	address 1
Addressing phase 2:	program to address 254 & set IO0 low	address 0, IO0 is high-Z	address 1
Addressing phase 3:	address 254	program to address 253 & set IO0 low	address 0
Addressing phase 4:	address 254	address 253	program to address 252 & set IO0 low
Addressing phase *X*:	*continue procedure*		

Figure 5.1 Addressing multiple TMC2041 via single wire interface using chaining

Proceed as follows:

- Tie the NEXTADDR pin of your first TMC2041 to GND.
- Interconnect one of the general purpose IO-pins of the first TMC2041 to the next drivers NEXTADDR pin using an additional pull-up resistor. Connect further drivers in the same fashion.
- Now, the first driver responds to address 0. Following drivers are set to address 1.
- Program the first driver to its dedicated slave address. Note: once a driver is initialized with its slave address, its general purpose output, which is tied to the next drivers NEXTADDR has to be programmed as output and set to 0.
- Now, the second driver is accessible and can get its slave address. Further units can be programmed to their slave addresses sequentially.



EXAMPLE FOR ADDRESSING UP TO 255 TMC2041

Addressing phase 1:	address 0, IO0 high	*address 1*	*address 1*
Addressing phase 2:	program to address 254 & set IO0 low	address 0, IO0 high	*address 1*
Addressing phase 3:	address 254	program to address 253 & set NAO low	address 0, IO0 high
Addressing phase 4:	address 254	address 253	program to address 252 & set IO0 low
Addressing phase *X*:	*continue procedure*		

Figure 5.2 Addressing multiple TMC2041 via differential interface, additional filtering for NEXTADDR

A different scheme (not shown) uses bus switches (like 74HC4066) to connect the bus to the next unit in the chain without using the NAI input. The bus switch can be controlled in the same fashion, using the NAO output to enable it (low level shall enable the bus switch). Once the bus switch is enabled it allows addressing the next bus segment. As bus switches add a certain resistance, the maximum number of nodes will be reduced.

It is possible to mix different styles of addressing in a system. For example a system using two boards with each two TMC2041 can have both devices on a board with a different level on NEXTADDR, while the next board is chained using analog switches separating the bus until the drivers on the first board have been programmed.

6 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 - global configuration - global status flags - slave address configuration - and I/O configuration
Current Setting	This register set offers registers for - driver current control and hold current
Motor Driver Register Set	This register set offers registers for - chopper and driver configuration - coolStep and stallGuard2 configuration

6.1 General Configuration Registers

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description / *bit names***	
				Bit	***GCONF* – Global configuration flags**
RW	0x00	10	*GCONF*	0	*single_driver* 0: Two motors can be operated. 1: Single motor, double current operation - driver 2 outputs are identical to driver 1, all driver 2 related controls are unused in this mode. *Attention: Set correctly before driver enable!*
				1	*stepdir1_enable* 0: Driver 1 does not accept steps. 1: Step/Dir interface enabled via STEP1 and DIR1
				2	*stepdir2_enable* 0: Driver 2 does not accept steps. 1: Step/Dir interface enabled via STEP2 and DIR2
				3	*reserved* set to 0
				4	*reserved* set to 0
				5	*reserved* set to 0
				6	*reserved* set to 0
				7	*test_mode* 0: Normal operation 1: Enable analog test output on pin REFR2 *SLAVEADDR* selects the function of REFR2: 0…4: T120, DAC1, VDDH1, DAC2, VDDH2 *Attention: Not for user, set to 0 for normal operation!*
				8	*shaft1* 1: Inverse motor 1 direction
				9	*shaft2* 1: Inverse motor 2 direction
				10	*lock_gconf* 1: GCONF is locked against further write access.

GENERAL CONFIGURATION REGISTERS (0X00…0X0F)					
R/W	**Addr**	**n**	***Register***	**Description / *bit names***	
R+C	0x01	4	*GSTAT*	**Bit**	***GSTAT* – Global status flags**
				0	*reset* 1: Indicates that the IC has been reset since the last read access to *GSTAT.* All registers have been cleared to reset values.
				1	*drv_err1* 1: Indicates, that driver 1 has been shut down due to overtemperature or short circuit detection since the last read access. Read *DRV_STATUS1* for details. The flag can only be reset when all error conditions are cleared.
				2	*drv_err2* *1:* Indicates, that driver 2 has been shut down due to overtemperature or short circuit detection since the last read access. Read *DRV_STATUS2* for details. The flag can only be reset when all error conditions are cleared.
				3	*uv_cp* 1: Indicates an undervoltage on the charge pump. The driver is disabled in this case.
R	0x02	8	*IFCNT*	Interface transmission counter. This register becomes incremented with each successful UART interface write access. It can be read out to check the serial transmission for lost data. Read accesses do not change the content. Disabled in SPI operation. The counter wraps around from 255 to 0.	
W	0x03	8 + 4	*SLAVECONF*	**Bit**	***SLAVECONF***
				7..0	*SLAVEADDR*: Sets the address of unit for the UART interface. The address becomes incremented by one when the external address pin NEXTADDR is active. Range: 0-253 (254), *default=0* In ring mode, 0 disables forwarding.
				11..8	*SENDDELAY*: 0, 1: 8 bit times (not allowed with multiple slaves) 2, 3: 3*8 bit times 4, 5: 5*8 bit times 6, 7: 7*8 bit times 8, 9: 9*8 bit times 10, 11: 11*8 bit times 12, 13: 13*8 bit times 14, 15: 15*8 bit times
R	0x04	9 + 8	*INPUT*	**Bit**	***INPUT***
					Reads the digital state of all input pins available plus the state of IO pins set to output.
				0	*io0_in*: IO0 polarity
				1	*io1_in*: IO1 polarity
				2	*io2_in*: IO2 polarity
				3	*io3_in*: IO3 polarity
				4	*iop_in:* IOP pin polarity (always input in SPI mode)
				5	*ion_in*: ION pin polarity (always input in SPI mode)
				6	*nextaddr_in*: NEXTADDR pin polarity
				7	*drv_enn_in*: DRV_ENN pin polarity
				8	*sw_comp_in*: UART input comparator (1: IOP voltage is above ION voltage). The accuracy is about 20mV.