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

TMC4361-LA DATASHEET

Cost-effective S-ramp motion controller with servo option for stepper motors. Optimized for high velocities. SPI and Step/Dir interfaces to motor driver and encoder interface for closed loop operation.

APPLICATIONS

Textile, Sewing Machines
Factory Automation
Lab Automation
Medical
Office Automation
Printer and Scanner
CCTV, Security
ATM, Cash recycler
POS
Pumps and Valves
Heliostat Controller
CNC Machines

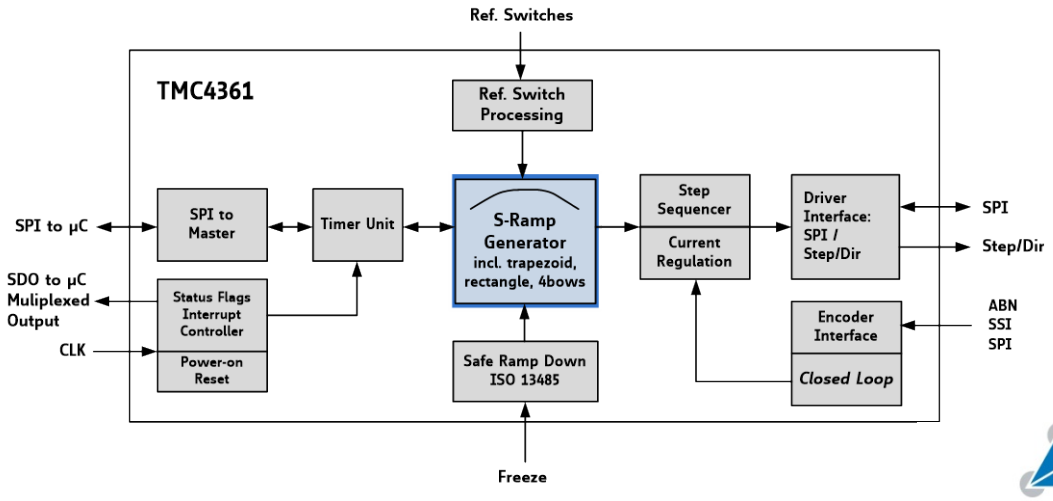
FEATURES AND BENEFITS

3.3V or 5V operation
SPI interface for µC with easy-to-use protocol
SPIinterface for SPI motor drivers
Step/Dirinterface forStep/Dir motor drivers
Clock frequency 4.2 MHz up to 32 MHz
Encoder interface: incremental ABN and serial SSI/SPI
2x ref.-switch input
Servo drive option
S-shaped or linear velocity ramps, optimally calculated
On-the-fly change of target motion parameters
Low power operation using clock gating technology
Different current levels related to the motion profile status
Programmable microstep table
Read-out option for all important motion parameters
Compact Size 6x6 mm² QFN40 package
Directly controls TMC23x, TMC24x, and TMC26x motor driver

DESCRIPTION

The TMC4361 is intended for applications where a fast and jerk-limited motion profile is desired. This motion controller adds to any microcontroller with SPI interface. It supports S-shaped, trapezoid, and rectangle ramps. With encoder, the TMC4361 allows for an extremely quick and precise positioning. Its servo features provide step-loss protection, energy efficiency, and target positioning with stepper typical stability. Standard SPI and STEP/DIR interfaces to the motor driver simplify communication.
High end features,no software effort and the small form factor of the TMC4361 enable miniaturized designs with low external component count for cost-effective and highly competitive solutions.

BLOCK DIAGRAM



TRINAMIC
MOTION CONTROL

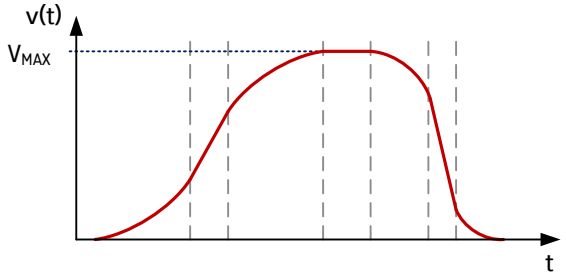
HIGH-END SOLUTION: VELOCITY MEETS PRECISION

The TMC4361 is a miniaturized high performance stepper motor controller with an outstanding cost-performance ratio. It is designed for high volume as well as for demanding industrial motion control applications. The TMC4361 is equipped with an SPI™ host interface (SPI is trademark of Motorola) with easy-to-use protocol and three driver interfaces (SPI, Step/Dir, and PWM) for addressing various stepper motor driver types.The TMC4361 scores with its unique servo drive features, high integration and a versatility that covers a wide spectrum of applications, motor sizes, and encoder types.

For a comfortable handling, the chip works with real world units. Extensive support at the chip, board, and software levels enables rapid design cycles and fast time-to-market with competitive products. High energy efficiency delivers further cost savings.

S-SHAPED VELOCITY PROFILE

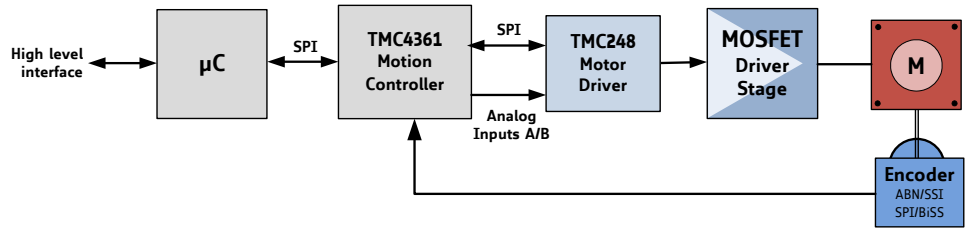
This outstanding ramp profile minimizes jerk. Seven segments of the ramp allow for an optimum adaption of the velocity profile to the customer specific application requirements. High torque with high velocities can be reached by calibrating the bows of the ramp in a way that the acceleration value near V_{MAX} is reduced in parallel to the available motor torque.



COMPACT DESIGN FOR RELIABLE CLOSED LOOP OPERATION

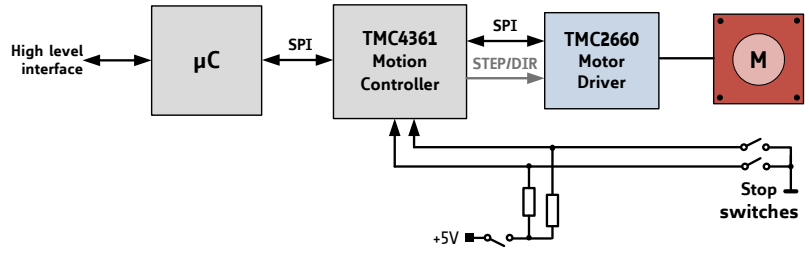
BENEFIT FROM HIGH VELOCITIES COMBINED WITH EXTREMELY HIGH PRECISION!

Closed loop operation is an optimum choice in case a dynamic and reliable drive without step-loss or motor stall is desired. The controller IC monitors the encoder values nonstop and uses them for a sophisticated motor field control.



COMPACT DESIGN FOR RELIABLE OPERATION USING STOP SWITCHES

The TMC4361 offers a left and a right stop switch in hardware as well as a home switch. Further, it provides two virtual stop switches which can trigger stop slopes in case the related virtual stop switch microstep position is reached.



ORDER CODES

Order code	Description	Size
TMC4361-LA	Motion controller with servo and dcStep features, QFN40	6 x 6 mm²

Table of Contents

1 Principles of Operation

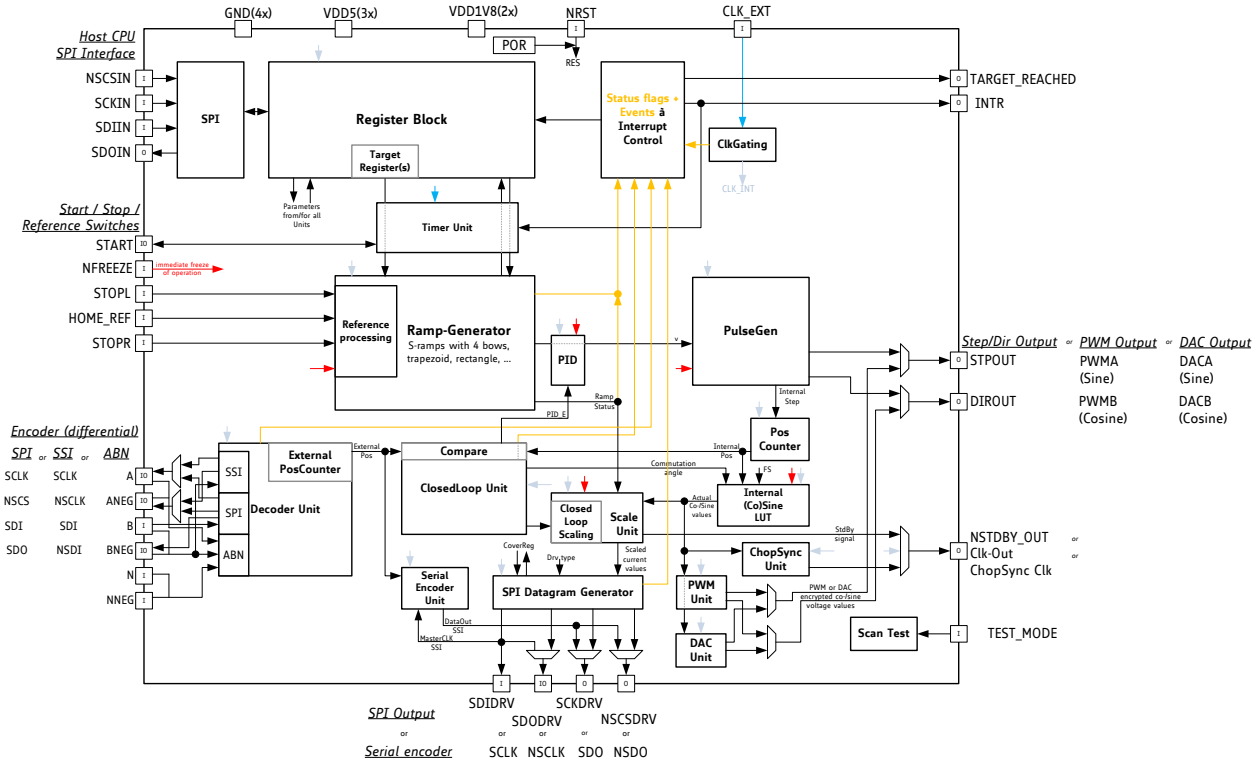


Figure 1.1 Basic application block diagram

1.1 Drive Concepts and Control Modes

The TMC4361 motion controller provides four different drive concepts respectivelycontrol modes. Choose the specific control mode related to the requirements of your application.

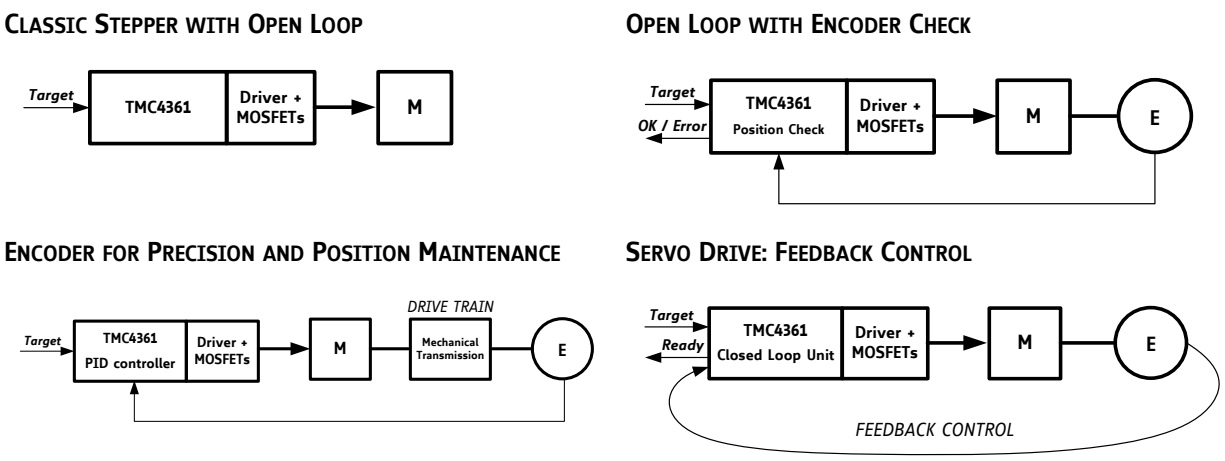


Figure 1.2Drive Concepts. M=Motor, E=Encoder

1.2 Key Concepts

The TMC4361 realizes real time critical tasks autonomously and guarantees for a robust and reliable drive. The following features contribute toward greater precision, greater efficiency, higher reliability, higher velocity, and smoother motion in many stepper motor applications.

Interfacing The TMC4361 offers application specific interfacing via SPI,Step/Dir, and PWM interface.

Initialization Adapt the TMC4361 to the driver type and configuration and send initial configuration data to SPI drivers. Configure microstep resolution and waveform.

Positioning The TMC4361 operates motor based on user specified target positions and velocities. Modify all motion target parameters on-the-fly during motion.

Microstepping Based on internal position counters the TMC4361 performs up to $\pm 2^{31}$ (micro)steps completely independent from the microcontroller. Microstep resolutions are individually programmable. The range goes from full stepping (1 microstep = 1 full step) and half stepping (2 microsteps per full step) up to 8 bit micro stepping (256 microsteps per full step) for precise positioning and noiseless stepper motor rotation. With Step/Dir drivers any microstep resolution is possible as supported by the driver.The internal microstep table can be adapted to specific motor characteristics to further reduce torque ripple.

Servo Drive The TMC4361 provides closed loop operation for Step/Dir and SPI drivers. Using a differential or serial encoder, the closed loop unit of the TMC4361 compares the external position counter values with the internal ones and sends signals for correction.

chopSync™ The TMC4361 has an integratedchopSync chopper for very smooth motor movementwith TMC23x/24x.

Programming Every parameter can be changed at any time. The uniform access to any TMC4361 register simplifies application programming. A read-back option for nearly all internal registers is available.

Synchronization The TMC4361 provides synchronizing several TMC4361 motion controller chips if it is desired to drive motors simultaneously. In this case one TMC4361 is the master and the connected TMC4361 are slaves.

1.3 Overview Interfaces

1.3.1 SPI to CPU

From the software point of view, the TMC4361 provides a set of registers, accessed by a microcontroller via a serial interface in a uniform way. Each datagram contains address bits, a read-write selection bit, and data bits to access the registers and the on-chip memory. Each time the microcontroller sends a datagram to the TMC4361 it simultaneously receives a datagram from the TMC4361. This simplifies the communication with the TMC4361 and makes programming easy. Most microcontrollers have an SPI hardware interface, which directly connects to the serial four wire microcontroller interface of the TMC4361. For microcontrollers without SPIhardware, software doing the serial communication is sufficient and can easily be implemented. (For further information refer to chapter 5.)

1.3.2 SPI to Driver

The TMC4361 automatically generates the required data-stream for SPI drivers and provides user configurable microstep waves and motor ramps. The serial interface to the motor driver is configurable for all TRINAMIC drivers as well as for SPI DACs. Pre-settings for TRINAMIC driver chips are provided. (For further information refer to chapter 11.)

Third party driver chips can be configured via SPI interface using cover datagrams. During motor movement the SPI interface remains switched off and the Step/Dir interface isused for driving the motor.

1.3.3 Step/Dir to Driver

The TMC4361 provides a configurable Step/Dir interface to the driver. The motion controller controls the motor position by sending pulses on the STEP signal while indicating the direction on the DIR signal. Programmable step pulse length and step frequencies allow operation at high speed and high microstep resolution. The driver chip converts these signals into the coil currents which control the position of the motor. The TMC4361 perfectly fits to the TMC26x smart power Step/Dir driver family. (For further information refer to chapters 11.4 and 11.5.)

1.3.4 PWM Interface to Driver

The TMC4361 allows for using PWM output values instead of Step/Dir outputs due to disabling the Step/Dir output and forwarding PWM signals via STPOUT_PWMA and DIROUT_PWMB. The PWM frequency is calculated with $f_{PWM} = f_{CLK} / PWM\_FREQ$.This mode supports noisefree and smooth microstepping with TMC23x and TMC24x stepper motor drivers. (For further information refer to chapter 13.)

1.3.5 Encoder Interface

The TMC4361 is equipped with a six pin encoder input interface for incremental ABN encoders (differential or single ended) or absolute encoders like SSI or SPI encoders. Motor feedback can be analyzed and closed loop behavior can be reached. All encoder input signals are filtered using anadaptable digital filter. (For further information refer to chapter 14.)

1.3.6 Reference Switches and Special IOs

The TMC4361 offers a left and a right stop switch in hardware as well as a home switch. Further, it provides two virtual stop switches which can trigger stop slopes in case the related virtual stop switch microstep position is reached. (Refer to chapter 9)

The START pin can be used as input or as output:a ramp start can be initialized viaa start input signal. The other way round, the START pin can be used as output. In this case, multiple drivers can be synchronized using an internal start signal of a TMC4361 master and forwarding it as start trigger to further TMC4361 which act as slaves then. (Refer to chapter 10.1.)

The TMC4361 provides a clock output (STDBY_CLK). This output can be used to provide a step-synchronous chopper or application specific. (Refer to chapter 11.)

1.3.7 Safety Stop

The low-active safety pin NFREEZE can be used to end current operations without any delay. This way, an emergency-stop can be realized in case of dysfunctions on board level. (Refer to chapter 12.)

1.4 Step Frequencies

All parameter units are real physical units. Therefore, it is necessary to set the *CLK_FREQ* register to the appropriate value in [Hz] which is given by the external clock. As operation frequency any value between 4.2 MHz and 32 MHz can be chosen.The maximum motion velocity is restricted by the clock frequency. Values higher than ½ pulse * f_{CLK} are prohibited because the STPOUT output remains active for one clock cycle and inactive for one clock cycle afterwards for a Step/Dir driver. The microstep resolution can be chosen in the range from full steps up to 256 microsteps per full step when using the internal sequencer. (Refer to chapter 8.2.3.)

1.5 Moving the Motor

Moving the motor is simple:
To move a motor to a *new target position*, write the target position into the associated register by sending a datagram to the TMC4361.
To move a motor with a *newtarget velocity*, write the velocity into the register assigned to the stepper motor.

1.5.1 Motion Controller Functionality

The ramp generator monitors the motion parameters stored in its registers and calculates velocity profiles. Based on the actual ramp generator velocity a pulse generator supplies step pulses to the motor driver and to the internal sequencer.

1.5.2 Ramp Modes and Types

Two general ramp modes can be chosen (see chapter 8.2):

Velocity mode The target velocity *VMAX* will be reached using the selected ramp type.

Positioning mode The maximum velocity value is used within the given ramp type as long as the target position is not exceeded. The stepping direction depends on *XACTUAL*, *XTARGET*, and the current ramp status.

Three ramp types can be selected: rectangle shaped ramps, trapezoidal ramps, and S-shaped ramps. S-shaped ramps in positioning mode finish exactly at the target position by keeping the actual velocity at maximum value as long as possible while staying within the motion limits. The slopes to and from maximum velocity are as fast as possible without exceeding limits.

1.6 Status Flags, Events, and Interrupts

The microcontroller connected to the TMC4361 normally requires status information. Therefore, the TMC4361 provides 32 status flags and 32 status events. Status events can be configured customer specific and led through to the interrupt output of the TMC4361. (Refer to chapter 7.)

2 Pin Assignments

2.1 Package Outline

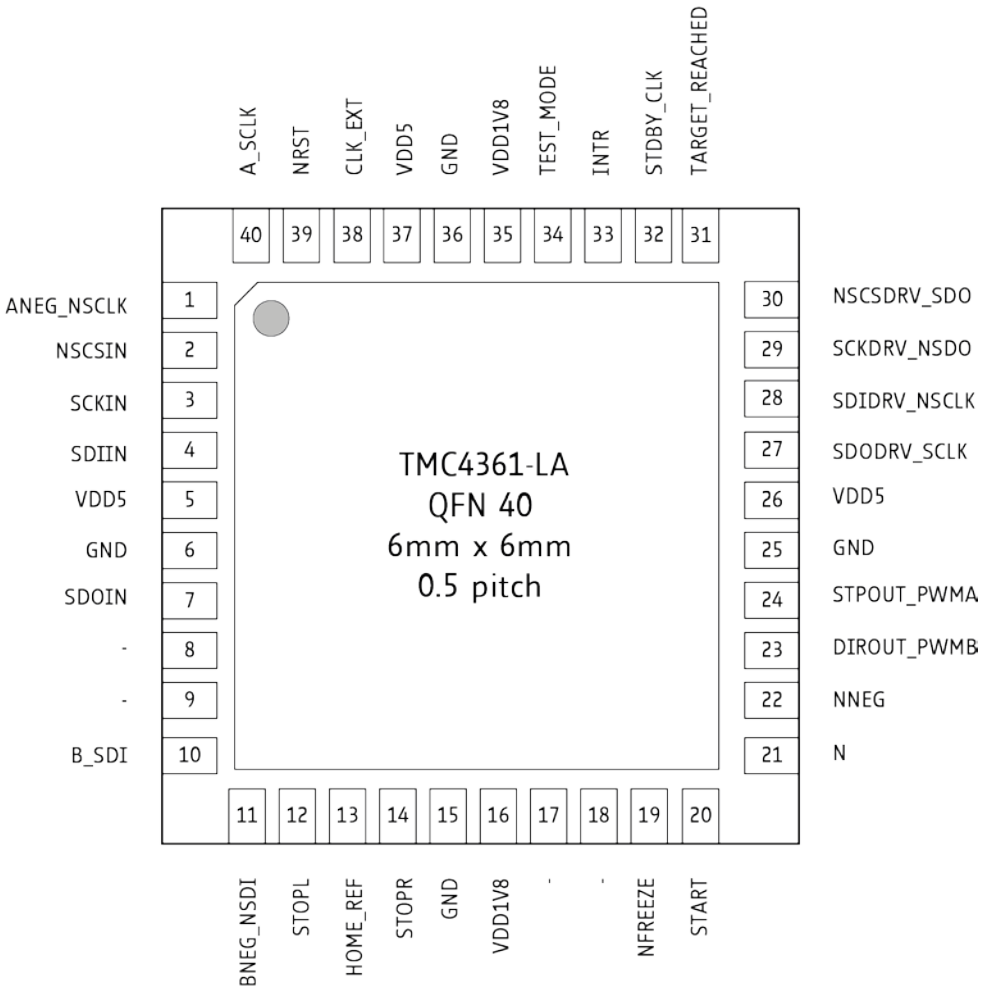


Figure 2.1Pinning (top view)

Attention: Do not connect pins without assignment!

2.2 Signal Description

Pin	Number	Type	Function
GND	6,15, 25,36	GND	Digital ground pin for IOs and digital circuitry
VDD5	5,26,37	VDD	Digital power supply for IOs and digital circuitry (3.3V… 5V)
VDD1V8	16,35	VDD	Connection of *internal generated* core voltage of 1.8V
NSCSIN	2	I	Low active chip select input of the SPI interface to the µC
SCKIN	3	I	Serial clock for the SPI interface to the µC
SDIIN	4	I	Serial data input of the SPI interface to the µC
SDOIN	7	O	Serial data output of the SPI interface to the µC (*Z* if NSCSIN=*1*)
CLK_EXT	38	I	Clock input to provide aclock with the frequency f_{CLK} for all internal operations.
NRST	39	I (PU)	Low active reset. If not connected, Power-on-Reset and internal pull-up resistor will be active.
TEST_MODE	34	I	Test mode input. Tie to low for normal operation.

Pin	Number	Type	Function
STOPL	12	I (PD)	Left stop switch. External signal to stop a ramp. If not connected, an internal pull-down resistor will be active.
HOME_REF	13	I (PD)	Home reference signal input. External signal for reference search.If not connected,an internal pull-down resistor will be active.
STOPR	14	I (PD)	Right stop switch. External signal to stop a ramp. If not connected, an internal pull-down resistor will be active.
INTR	33	O	Interrupt output
TARGET_REACHED	31	O	Target reached output
START	20	IO	Start signal input/output
NFREEZE	19	I (PU)	Low active safety pin to immediately freeze output operations.If not connected, an internal pull-up resistor will be active.
STDBY_CLK	32	O	StandBy signal or internal CLK output or ChopSync output
N	21	I (PD)	N signal input of incremental encoder input interface If not connected, an internal pull-down resistor will be active.
NNEG	22	I (PD)	Negated N signal input of incremental encoder input interface If not connected, an internal pull-down resistor will be active.
B SDI	10	I (PD)	B signal input of incremental encoder input interface. Serial data input signal of serial encoder input interface (SSI/SPI). If not connected, an internal pull-down resistor will be active.
BNEG NSDI SDO_ENC	11	IO	Negated B signal input of incremental encoder input interface. Negated serial data input signal of SSI encoder input interface Serial data output of SPI encoder input interface.
A SCLK	40	IO	A signal input of incremental encoder interface. Serial clock output signal of serial encoder interface (SSI/SPI).
ANEG NSCLK NSCS_ENC	1	IO	Negated A signal input of incremental encoder interface. Negated serial clock output signal of serial encoder interface. Low active chip select output of SPI encoder input interface.
STPOUT PWMA DACA	24	O	Step output. First PWM signal (Sine). First DAC output signal (Sine).
DIROUT PWMB DACB	23	O	Direction output. Second PWM signal (Cosine). Second DAC output signal (Cosine).
NSCSDRV SDO	30	O	Low active chip select output of SPI interface to motor driver. Serial data output of serial encoder output interface.
SCKDRV NSDO	29	O	Serial clock output of SPI interface to motor driver. Negated serial data output of serial encoder output interface.
SDODRV SCLK	27	IO	Serial data output of SPI interface to motor driver. Clock input of serial encoder output interface.
SDIDRV NSCLK	28	I (PD)	Serial data input of SPI interface to motor driver. Negated clock input of serial encoder output interface If not connected, an internal pull-down resistor will be active.
n.c.	8,9,17,18	-	Do not connect

PD: if n.c. → pull-down
PU: if n.c. → pull-up

3 Sample Circuits

The sample circuits show the connection of external components.

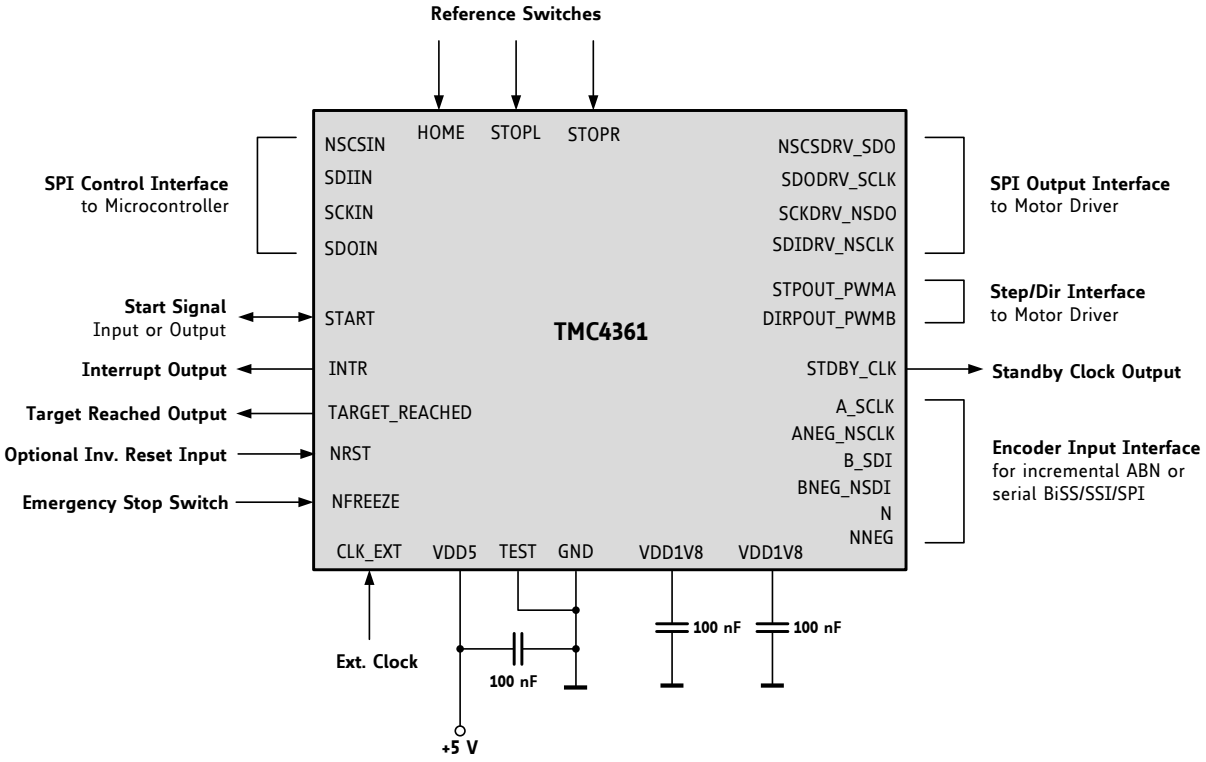


Figure 3.1 How to connect the TMC4361

CHECK YOUR CONNECTIONS!

1. Check, if the STDBY_CLK output provides the pulse which is applied at the CLK_EXT input pin. If both values fit, POR (power on reset), power supply and clk frequency are ready to be used.
2. The SPI communication to TMC4361 is established if the step frequency at STPOUT_PWMA matches to the selected value of VMAX (maximum velocity value). This relationship is valid with a clock frequency of 16MHz. When using another frequency, it is necessary to convert the values appropriately.

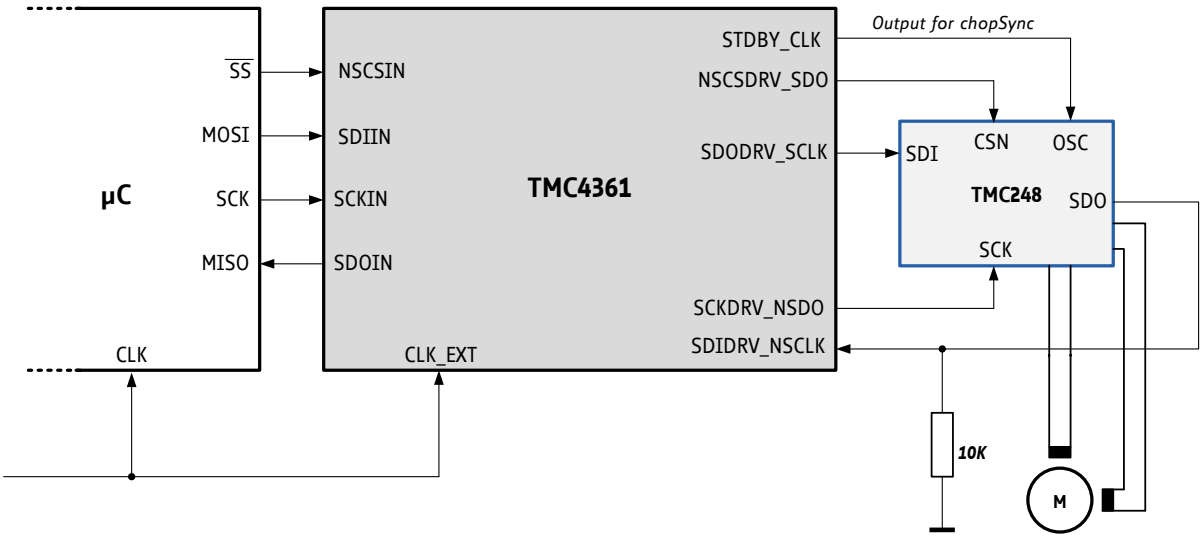


Figure 3.2 TMC4361 with TMC248 stepper driver in SPI mode

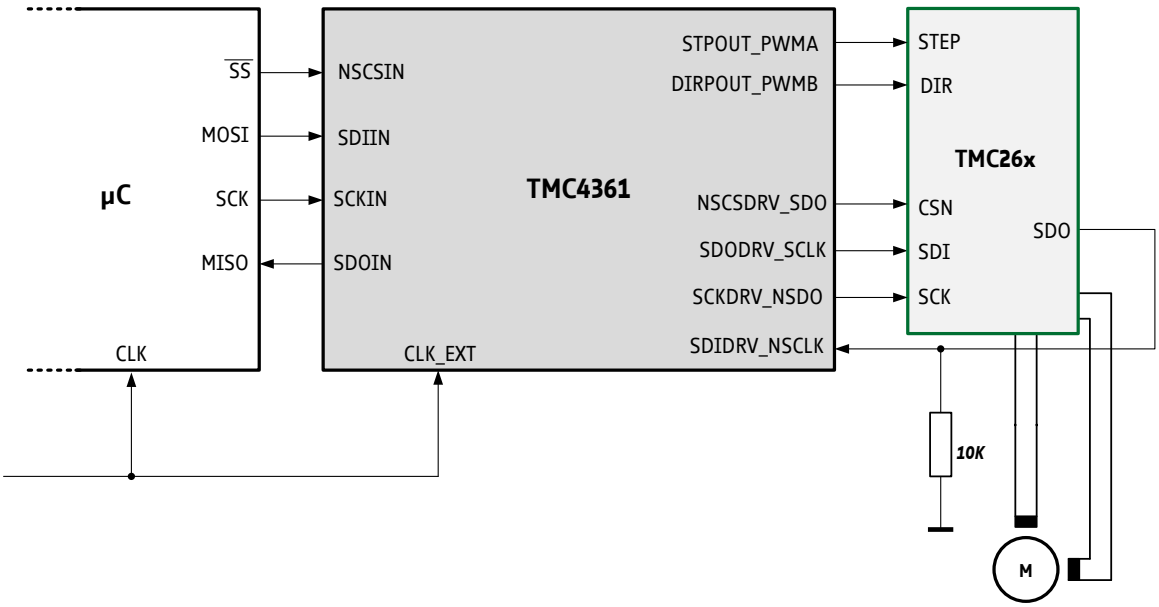


Figure 3.3 TMC4361 with TMC26x stepper driver in Step/Dir mode. The SPI interface is used for configuration.

4 Notes

REGISTER names are italicized with VALUE REGISTER in capital letters and switches with small letters.

PIN names are written with capital letters.

5 SPI Control Interface

The TMC4361 uses 40 bit SPI™ datagrams for communication with a microcontroller. The bit-serial interface is 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 TMC4361 slave always consists of sending one 40-bit command word and receiving one 40-bit status word. The SPI command rate typically comprises a few commands per complete motor motion.

SPI CONTROL INTERFACE

Pin Name	Type	Remarks
NSCSIN	Input	Chip Select of the SPI-µC interface (low active)
SCKIN	Input	Clock of the SPI-µC interface
SDIIN	Input	Data input of the SPI-µC interface
SDOIN	Output	Data output of the SPI-µC interface

5.1 SPI Datagram Structure

Microcontrollers which are equipped with hardware SPI are typically able to communicate using integer multiples of 8 bit. The NSCSINline of the TMC4361has to be handled in a way, that it stays active (low) for the complete duration of the datagram transmission.

Each datagram sent to the TMC4361 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 TMC4361. Each register is accessed via 32 data bits even if it uses less than 32 data bits.

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.

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

TMC4361 SPI DATAGRAM STRUCTURE

MSB (transmitted first) 40 bit LSB (transmitted last)				
39 … … 0				
→ 8 bit address ← 8 bit SPI status	←→ 32 bit data			
39 … 32	31 … 0			
→ to TMC4361: RW + 7 bit address ← from TMC4361: 8 bit SPI status	8 bit data	8 bit data	8 bit data	8 bit data
39 / 38 … 32	31 … 24	23 … 16	15 … 8	7 … 0
W 38…32	31…28 27…24	23…20 19…16	15…12 11…8	7…4 3…0
39 38 37 36 35 34 33 32	31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0

5.1.1 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. Thus, 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.Further, the following read or write access delivers back 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. Data which will be delivered are latched immediately after the prior data transfer.

Whenever data is read from or written to the TMC4361, the MSBs delivered back contain the SPI status*SPI_STATUS*, which is a number of eight status bits.The selection of these bits will be explained in chapter 7.2.

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, e.g., 0.

action	data sent to TMC…	data received from TMC…
read *XACTUAL*	→0x2100000000	←0xSS & unused data
read *XACTUAL*	→0x2100000000	←0xSS & X_ACTUAL
write *VACTUAL*:= 0x00ABCDEF	→0xA200ABCDEF	←0xSS & X_ACTUAL
write *VACTUAL*:= 0x00123456	→0xA200123456	←0xSS00ABCDEF

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

5.1.2 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.

5.2 SPI Signals

The SPI bus on the TMC4361 has four signals:

- SCKIN – bus clock input
- SDIIN – serial data input
- SDOIN – serial data output
- NSCSIN – chip select input (active low)

The slave is enabled for an SPI transaction by a transition to low level on the chip select input NSCSIN. Bit transfer is synchronous to the bus clock SCKIN, with the slave latching the data from SDIIN on the rising edge of SCKIN and driving data to SDOIN following the falling edge. The most significant bit is sent first. A minimum of 40 SCKIN clock cycles is required for a bus transaction with the TMC4361. If less than 40 clock cycles are transmitted, the transfer will not be valid, even for a read access. However, sending only eight clock cycles can be useful to obtain the SPI status because it sends the status information back first.

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

NSCSIN must be low during the whole bus transaction. When NSCSIN 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 NSCSIN are recognized as the command.

5.3 Timing

The SPI interface is synchronized to the internal system clock, which limits the SPI bus clock SCKIN to half of the system clock frequency. The signal processing of the SPI inputs are supported with internal Schmitt Trigger, but not with RC elements. To avoid glitches at the inputs of the SPI interface between µC and TMC4361, external RC elements have to be provided. Figure 5.1 shows the timing parameters of an SPI bus transaction and the table below specifies the parameter values.

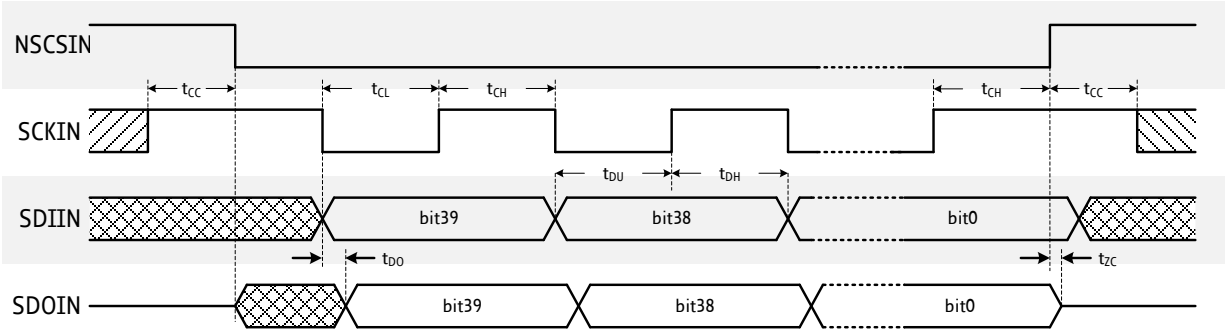


Figure 5.1 SPI timing

SPI interface timing	**AC-Characteristics** clock period: t_{CLK}					
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
SCKIN valid before or after change of NSCSIN	t_{CC}		10			ns
NSCSIN high time	t_{CSH}	*) Min time is for synchronous CLK with SCKIN high one t_{CH} before SCSIN high only	t_{CLK}*)	>2t_{CLK}+10		ns
SCKIN low time	t_{CL}	*) Min time is for synchronous CLK only	t_{CLK}*)	>t_{CLK}+10		ns
SCKIN high time	t_{CH}	*) Min time is for synchronous CLK only	t_{CLK}*)	>t_{CLK}+10		ns
SCKIN frequency using external clock (Example: f_{CLK} = 16 MHz)	f_{SCK}	assumes synchronous CLK			f_{CLK} / 2 (8)	MHz
SDIIN setup time before rising edge of SCKIN	t_{DU}		10			ns
SDIIN hold time after rising edge of SCKIN	t_{DH}		10			ns
Data out valid time after falling SCKIN clock edge	t_{DO}	no capacitive load on SDOIN			t_{FILT}+5	ns

t_{CLK} = 1 / f_{CLK}

6 Input Filtering

Input signals can be noisy due to long cables and circuit paths. To prevent jamming, every input pin provides a Schmitt Trigger. Additionally, several signalsare passed through a digital filter. Particular inputpins areseparated into four filtering groups. Each groupcan be programmedindividuallyaccording to its filter characteristics.

PINS AND REGISTERS: INPUT FILTERING GROUPS

Pin names	Type		Remarks
A_SCLK B_SDI N ANEG_NSCLK BNEG_NSDI NNEG	Inputs		Encoder interface input pins
STOPL HOME_REF STOPR	Inputs		Reference input pins
START	Input		START input pin
SDODRV_SCLK SDIDRV_NSCLK	Inputs		Master clock input interface pins for serial encoder
Pin name	Register address		Remarks
INPUT_FILT_CONF	0x03	RW	Filter configuration for all four input groups

6.1 Input Filter Configuration

Every filtering groupcan be configured separately with regard toinput sample rate and digital filter length.

6.1.1 Input Sample Rate (SR)

$f_{CLK} \cdot 1 / 2^{SR}$

where SR (extended with the particular name extension) isin [0… 7].

This means that every $(2^{SR})^{th}$ input bit will be considered for internal processing.

Sample rate configuration	
SR value	Sample rate
0	f_{CLK}
1	f_{CLK} / 2
2	f_{CLK} / 4
3	f_{CLK} / 8
4	f_{CLK} / 16
5	f_{CLK} / 32
6	f_{CLK} / 64
7	f_{CLK} / 128

6.1.2 Digital Filter Length (FILT_L)

One bit is sampled within each $(2^{SR})^{th}$input clock cycle. The filter length FILT_L can be set within the range [0… 7]. The filter lengthFILT_L specifies the number of sampled bits thatmust have the same voltage level toseta new input bit voltage level.

Configuration of digital filter length	
FILT_L value	Filter length
0	No filtering
1	2 equal bits
2	3 equal bits
3	4 equal bits
4	5 equal bits
5	6 equal bits
6	7 equal bits
7	8 equal bits

6.1.3 Examples

The following three examples depict the input pin filtering of three different input filtering groups. The voltage levels after passing the Schmitt Trigger are compared to the internal signals which are processed by the motion controller.

The sample points are depicted as green dashed lines.

Reference Input Pins

Here, every second clock cycle is sampled.Two sampled input bits must be equal to be a valid input voltage.

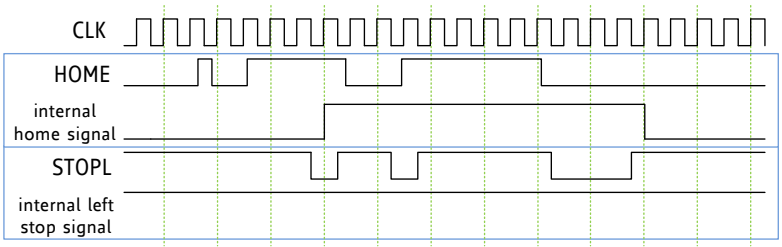


Figure 6.1 Reference input pins: SR_REF = 1, FILT_L_REF = 1

START Input Pin

Every fourth clock cycle is sampled and the sampled input bit is valid.

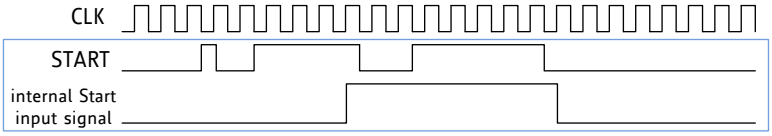


Figure 6.2START input pin: SR_S = 2, FILT_L_S = 0

Encoder Interface Input Pins

Every clock cycle bit is sampled.Eight sampled input bits must be equal to be a valid input voltage.

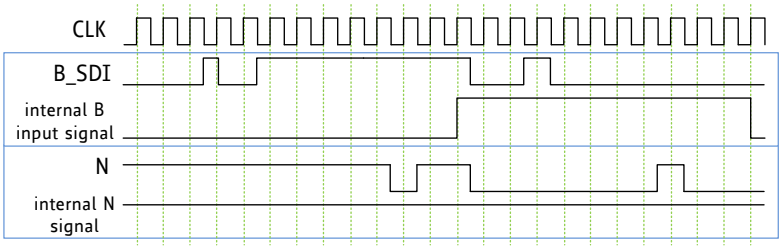


Figure 6.3 Encoder interface input pins: SR_ENC_IN = 0, FILT_L_ENC_IN = 7

7 Status Flags & Events

The TMC4361 offers several possibilities for velocity ramps.It combinestarget positioning and velocity ramps without interventions in between. However, the microcontroller connected to the TMC4361 normally requiresstatus information. Therefore, TMC4361 provides 32 status flags and 32 status events. Status events can be configured customer specific and lead through using the interrupt output of the TMC4361. Further, the eight SPI status bits sent witheach SPI datagram can be read out.

PINS AND REGISTERS: STATUS FLAGS AND EVENTS

Pin names	Type		Remarks
INTR	Output		Interrupt output to indicate status events
Register name	Register address		Remarks
STATUS_FLAGS	0x0F	R	32 status flags of the TMC4361 and the connected TMC motor driver chip
EVENTS	0x0E	R+C	32 events triggered by altered TMC4361 status bits
SPI_STATUS_SELECTION	0x0B	RW	Selection of 8 out of 32 events for SPI status bits
EVENT_CLEAR_CONF	0x0C	RW	Exceptions for cleared event bits
INTR_CONF	0x0D	RW	Selection of 32 events for INTR output

7.1 Status Flags

Status bits of the *STATUS_FLAGS* register are specified in the register chapter (see 17).

7.2 Status Events &SPI Status &Interrupts

STATUS FLAGS - STATUS EVENTS

Status eventsare triggered during the transition process of status bits from inactive to active level.Status bits and status events are associated in different ways:

- Several status eventsare associated with one status bit.
- Some status events show the status transition of one or more status bits out of a status bit group.The motor driver flags, e.g.,triggeronly one motor driver event *MOTOR_EV*in case one of the selected motor driver status flags becomes active.
- In case a flag consists of more than one bit,the number of associated events that can be triggered corresponds to the valid combinations. The *VEL_STATE* flag, e.g., has two bit but three associated velocity state events (00/01/10).Such an eventistriggeredif the associated combination switches from inactive to active.
- Furthermore, some events have no equivalence in the *STATUS_FLAGS* register(e.g.,*COVER_DONE* which indicates new data from the motor driver chip).

The *EVENTS* register is automatically cleared after reading the register subsequent to an SPI datagram request.

To prevent events from being cleared, the *EVENT_CLEAR_CONF* register can be assigned properly. Just set the related *EVENT_CLEAR_CONF* register bit position to 1.

HOW TO AVOID A LACK OF INFORMATION

The recognition of a status event can fail in case it is triggered right before or during the *EVENTS* register becomes cleared. By setting the *EVENT_CLEAR_CONF* register appropriately, this can be avoided. Up to eight events can be selected for permanent SPI status report.Therefore, select up to eight events by writing 1 to the specific bit positions of the *SPI_STATUS_SELECTION*register.The bit positionsaresorted according to the event bit positions in the *EVENTS* register. In case more than eight events are chosen, the first eight bits (starting from index 0) are forwarded as *SPI_STATUS*.

INTERRUPTS

Similar to the *EVENT_CLEAR_CONF* register and the *SPI_STATUS_SELECTION*register, events can be selected using the *INTR_CONF* register to be forwarded to the INTR output.The active polarity of the INTR output can be set with*intr_pol*. The selected events will be ORed to one signal. The INTR output becomes active as soon as one of the selected events triggers.

Due to the importance of events for interrupt generation and SPI status monitoring, it is recommended to clear the *EVENTS* register before starting regular operation.

8 Ramp Generator

Stepgeneration is one of themain tasks of a steppermotor motion controller. The internal ramp generator of the TMC4361 provides several ways of step generation in order to form different ramp types to fit for various applications.

PINS AND REGISTERS: RAMP GENERATOR

Pin names	Type		Remarks
STPOUT_PWMA	Output		Step output signal
DIROUT_PWMB	Output		Direction output signal
Register name	Register address		Remarks
GENERAL_CONF	0x00	RW	Ramp generator affecting bits 1 : 5
STP_LENGTH_ADD DIR_SETUP_TIME	0x10	RW	Additional step length in clock cycles; 16 bits Additional time in clock cycles when no steps will occur after a direction change; 16 bits
RAMPMODE	0x20	RW	Requested ramp type and mode; 3 bits
XACTUAL	0x21	RW	Current internal microstep position;signed; 32 bits
VACTUAL	0x22	R	Current step velocity; 24 bits; signed; no decimals
AACTUAL	0x23	R	Current step acceleration; 24 bits; signed; no decimals
VMAX	0x24	RW	Maximum permitted or target velocity; signed; 32 bits= 24+8 (24 bits integer part, 8 bits decimal places)
VSTART	0x25	RW	Velocity at ramp start; unsigned; 31 bits=23+8
VSTOP	0x26	RW	Velocity at ramp end; unsigned; 31 bits=23+8
VBREAK	0x27	RW	At this velocity value, the ac-/deceleration will change during trapezoidal ramps; unsigned; 31 bits=23+8
AMAX	0x28	RW	Maximum permitted or target acceleration; unsigned; 24 bits=22+2 (22 bits integer part, 2 bits decimal places)
DMAX	0x29	RW	Maximum permitted or target deceleration; unsigned; 24 bits=22+2
ASTART	0x2A	RW	Acceleration at ramp start or below VBREAK; unsigned; 24 bits=22+2
DFINAL	0x2B	RW	Deceleration at ramp end or below VBREAK; unsigned; 24 bits=22+2
BOW1	0x2D	RW	First bow value of a complete velocity ramp; unsigned; 24 bits=24+0 (24 bits integer part, no decimal places)
BOW2	0x2E	RW	Second bow value of a complete velocity ramp; unsigned; 24 bits=24+0
BOW3	0x2F	RW	Third bow value of a complete velocity ramp; unsigned; 24 bits=24+0
BOW4	0x30	RW	Fourth bow value of a complete velocity ramp; unsigned; 24 bits=24+0
CLK_FREQ	0x31	RW	External clock frequency f_{CLK}; unsigned; 25 bits
XTARGET	0x37	RW	Target position;signed; 32 bits

8.1 Step/Dir Output Configuration

Step/Dir output signals can be configured for the driver circuit:

- For step signals that have to be longer than one clock cycle set*STP_LENGTH_ADD* appropriately. Then, the resulting step length is equal to *STP_LENGTH_ADD*+1 clock cycles. Thus, the step length can be chosen within the range 1…2^{16} clock cycles.
- DIROUT does not change the level during the active step pulse signal and for *STP_LENGTH_ADD*+1 clock cycles after the step signal returns to theinactive level.
- With the register *DIR_SETUP_TIME* the delay [clock cycles] between DIROUT and STPOUT voltage level changes can be set. Using this register, no steps are sent via STPOUT for *DIR_SETUP_TIME* clock cycles after a level change at DIROUT.

Note:
- Per default, the step output is high active because a rising edge at STPOUT indicates a step.
- For changing the polarity, set*step_inactive_pol*=1. Now,each falling edge indicates a step.
- A step can be generated by toggeling the step output. Therefore, set *toggle_step*=1.
- *pol_dir_out* sets the output level for the negative velocity direction.
- *pol_dir_out*, *step_inactive_pol*, and *toggle_step* are part of the general configuration register.

8.2 Ramp Modes and Types

With proper configuration, the internal ramp generator of the TMC4361 is able to generate various ramps andthe relatedstep outputsfor STPOUT. Note, that there are many possibilities to combine a *general ramp mode* (velocity mode, positioning mode) with *basicramp types* (ramp in hold mode, trapezoidal ramp, S-shaped ramp). Therefore, select the general ramp mode first and proceed with the ramp type and further specifications, e.g., setting start and stop velocities or choosing different acceleration/deceleration values for each ramp phase.

General Ramp Modes

Two general ramp modes can be chosen with the*RAMPMODE*register. Therefore, bit 2 of the *Ramp Generator Register Set*(see chapter 17.16) is used:

RAMPMODE(2)=0 *Velocity mode*. The target velocity *VMAX* will be reached using the selected ramp type.

RAMPMODE(2)=1 *Positioning mode*. *VMAX* is the maximum velocity value which will be used within the given ramp type and as long as the target position *XTARGET* will not be exceeded. Furthermore, the sign of *VMAX* is not relevant during positioning. The direction of the steps depends on *XACTUAL*, *XTARGET*, and the current ramp status.

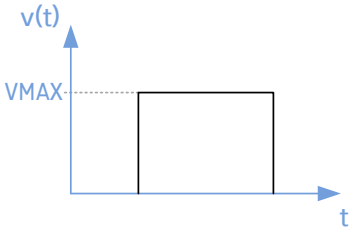
Ramp Types

Three basic ramp types are provided. These types differ in the velocity value development during the drive. For setting the basic ramp type, use the *Ramp Generator Register Set* bits 1 and 0.

TMC4361 Ramp Types		
***RAMPMODE*(1 : 0)**	**Ramp type**	**Function**
b'00	*Ramp in hold mode*	Follow *VMAX*only (rectangle velocity shape).
b'01	*Trapezoidal ramp*	Consideration of acceleration and deceleration values without adaption of these values.
b'10	*S-shaped ramp*	Use all ramp values (including bow values).

RAMPMODE(1 : 0)=00

Rectangle shaped ramp type in hold mode. *VACTUAL* is set immediately to *VMAX*.

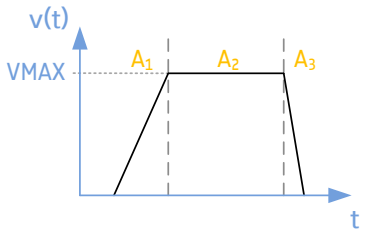


In positioning mode (*RAMPMODE*(2)=1), *VACTUAL* is set instantly to 0 if the target position is reached.
For exact positioning, it is recommended to set VMAX ≤ fCLK · ¼ pulses

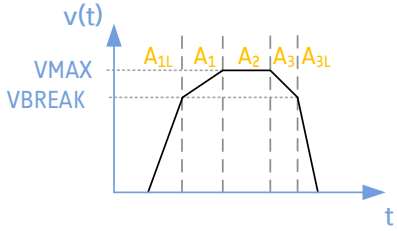
Figure 8.1 Rectangle shaped ramp type

RAMPMODE(1 : 0)=01

Trapezoidal shaped ramp type



Acceleration slope and deceleration slope haveonly one acceleration/deceleration value each. For this types, set *VBREAK* = 0.



The acceleration/deceleration factor alters at *VBREAK*.
In positioning mode, the ramp finishes exactly at the target position *XTARGET* by keeping *VACTUAL* = *VMAX* as long as possible.

Figure 8.2 Trapezoidal shaped ramp type

This trapezoidal ramp type reaches*VMAX*using linear ramps whereas the actual acceleration/deceleration factor *AACTUAL* depends on the current ramp phase and the velocity which should be reached. The corresponding sign assignment for different ramp phases is depicted in the following table:

Ramp phase:	A_{1L}	A_1	A_2	A_3	A_{3L}
v>0: AACTUAL=	ASTART	AMAX	0	-DMAX	-DFINAL
v<0: AACTUAL=	-ASTART	-AMAX	0	DMAX	DFINAL

RAMPMODE(1 : 0)=10

S-shaped ramp types

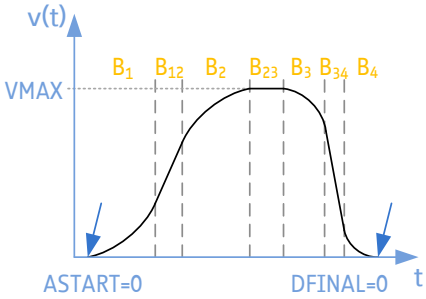
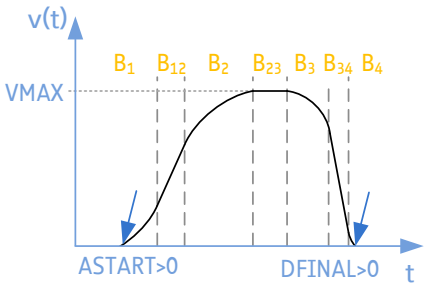


Figure 8.3 S-shaped ramp without initial and final acceleration/deceleration values



The start phase and the end phase of an S-shaped rampcan be accelerated/decelerated by *ASTART and DFINAL*.Using these parameters, the ramp starts with *ASTART* and it is ended with *DFINAL*. *DFINAL* becomes valid as soon as *AACTUAL* reaches the chosen *DFINAL* value. *ASTART* and *DFINAL*can be set separately.

Figure 8.4 S-shaped ramp type with initial acceleration and final deceleration value for B1 and B4

This ramp type reaches*VMAX* by means of S-shaped ramps whereas the acceleration/deceleration factor depends on the current ramp phase and alters every 64 clock cycles during the bow phases B_1, B_2, B_3, and B_4.

Ramp phase:	B_1	B_{12}	B_2	B_{23}	B_3	B_{34}	B_4
v>0: AACTUAL=	ASTART→AMAX	AMAX	AMAX→0	0	0→-DMAX	-DMAX	-DMAX→-DFINAL
BOW_{ACTUAL}=	BOW1	0	-BOW2	0	-BOW3	0	BOW4
v<0: AACTUAL=	-ASTART→-AMAX	-AMAX	-AMAX→0	0	0→DMAX	DMAX	DMAX→DFINAL
BOW_{ACTUAL}=	-BOW1	0	BOW2	0	BOW3	0	-BOW4

S-SHAPED RAMPS IN POSITIONING MODE

The ramp finishesexactly at the target position by keeping abs(*VACTUAL)* = *VMAX* as long as possible. Furthermore, the slopes to and from *VMAX*are as fast as possible without exceeding given values. It is even possible that the phases B_{12}, B_{23}, and $B_{34}$are left out due to given values. Nevertheless, the S-shaped ramp style is always performed in positioning mode, if *RAMP_MODE(1 : 0)* = b'10 is set.The parameter *DFINAL* is not considered during positioning mode.

8.2.1 Velocity Start *VSTART*and Velocity Stop *VSTOP*

S-shaped and trapezoidal velocity ramps can be started with an initial velocity value by setting *VSTART* higher than zero (see Figure 8.5).Such an S-shaped ramp with *VSTART* > 0 is a ramp without the first ramp bow B_1. The ramp starts with *AACTUAL* = *AMAX* and *VACTUAL* = *VSTART*.Logically, the parameter *ASTART*is not considered.

It is also possible to set *VSTOP* (a final velocity value) which finishes the ramp if*VACTUAL* reaches the *VSTOP*value (see Figure 8.6). This leads to an S-shaped velocity ramp without the bow B_4. Hence, *DFINAL*is not considered.

TRAPEZOIDAL AND S-SHAPED RAMPS USING PARAMETER *VSTART*

VSTART> 0 and *VSTOP* = 0

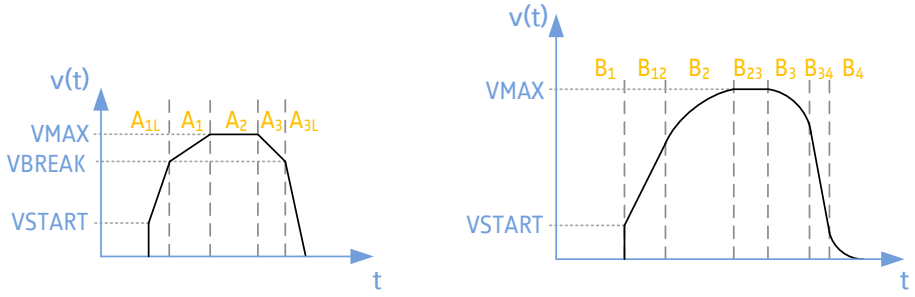


Figure 8.5 Trapezoidal and S-shaped ramps using *VSTART*

TRAPEZOIDAL AND S-SHAPED RAMPS USING PARAMETER *VSTOP*

VSTART = 0 and *VSTOP*> 0

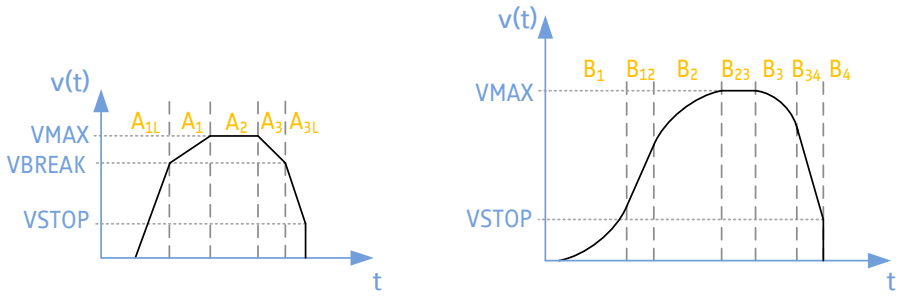


Figure 8.6 Trapezoidal and S-shaped ramps using *VSTOP*

TRAPEZOIDAL AND S-SHAPED RAMPS USING PARAMETERS *VSTART* AND *VSTOP*

VSTART> 0 and *VSTOP*> 0

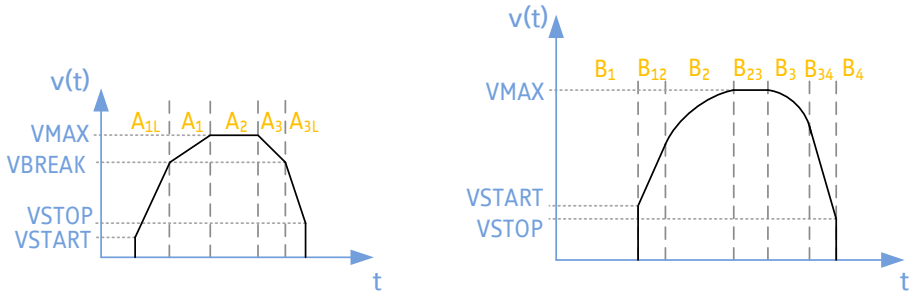


Figure 8.7 Trapezoidal and S-shaped ramps using *VSTART* and *VSTOP*

SUGGESTIONS

- *VSTART*and *VSTOP* are used whenstarting or ending a velocity ramp. If the velocity direction alters due to register assignments while a velocity ramp is in progress, the velocity values develop according to the current velocity ramp type without using *VSTART*or *VSTOP*.
- *VSTOP* is used in positioning modewhen the target position is reached. In velocity mode, *VSTOP*is only used when*VACTUAL* ≠ 0 and the target velocity *VMAX* is assigned to 0.
- The unsigned values *VSTART* and *VSTOP* are valid for both velocity directions.
- Every register value change is assigned immediately.

8.2.2 Limitations

ATTENTION

- Ramp parameter value changes in positioning mode during the ramp progress (except*VMAX and XTARGET*) are allowed but can result in a temporarily overshooting of *XTARGET*.

- To stop an S-shaped ramp during positioning do not set only *VMAX* = 0!There are two possibilities for further settings:
  Switch to velocity mode soon after setting *VMAX* = 0 and when reaching *VACTUAL* = 0 (*VEL_REACHED* event triggers) switch back to positioning mode.
  The other possibility is to set *VMAX* = 1. As soon as the*VEL_REACHED* event triggers, set *VMAX* to 0.

- *Only valid for trapezoidal ramps*: If a register value during a deceleration ramp (e.g. target position)is altered in a way that an immediate acceleration in the same direction must follow, the deceleration ramp becomes finished to *VACTUAL* = 0 first. Afterwards, the acceleration slope begins regularly. To avoid the unintentional finishing process of the deceleration ramp, set *VMAX* < abs(*VACTUAL*).If *VMAX* is reached now, set *VMAX* to the requested first value.
  The same procedure has to be usedin velocity mode if *VMAX*becomes decreased and increased again during the deceleration slope.
  Very slow deceleration slopes (*DMAX* ≤ (*VMAX* / 20s)) of trapezoidal ramps can result in an overdrive of the target position with an immediate subsequent ramp to overhaul target mismatch. To avoid this, please use a reasonable value for *VSTOP*.

- A *VACTUAL* value which exceeds *VMAX* can be result of register changes during an S-shaped ramp. This is, because the bows B1, B2, B3, and B4 are maintained during the ramp progress.

- If the requested conditions for the acceleration slope of an S-shaped ramp (*VSTART* or *ASTART*, BOW1 and BOW2) do not fit to*VMAX*, the starting acceleration value *ASTART*becomes altered. In case of misconfiguration at ramp start *AMAX*or *VSTART*have to be decreased in order to reach *XTARGET*.

FASTEST POSSIBLE SLOPE IN POSITIONING MODE

The fastest possible slopes are always performed if the phases B_{12} and/or B_{34} are not reached during a rising and/or falling S-shaped slope. Thus, the ramp maintains the maximum velocity *VMAX* as long as possible in positioning mode until the falling slope finishes the ramp to reach *XTARGET* exactly. The result is the fastest possible positioning ramp in matters of time.

8.2.3 Internal Ramp Generator Units

All parameter units are real arithmetical units. Therefore, it is necessary to set the *CLK_FREQ* register to the appropriate value in [Hz] which is given by the external clock. Any value between 4.2 MHz and 32 MHz can be chosen.

VELOCITY VALUES

VACTUAL is given as a 32 bit signed value with no decimal places. The unsigned velocity values *VSTART*, *VSTOP*, and *VBREAK* consist of 23 digits and 8 decimal places. *VMAX*is a signed value with 24 digits and 8 decimal places. Velocity values are given in pulses per second [pps].

The maximum velocity *VMAX* is restricted by the clock frequency. Values higher than ½ puls * f_{CLK} are prohibited because of an incorrect STPOUT output if *VACTUAL* exceeds this limit.

ACCELERATION VALUES

The unsigned values *AMAX*, *DMAX*, *ASTART*, and *DFINAL* consist of 22 digits and 2 decimal places. *AACTUAL* shows a 24 bit non decimal signed value. Acceleration and deceleration units are given in pulses per second² [pps²].

BOW PARAMETER VALUES

Bow values are unsigned 24 bit values without decimal places. They are given in pulses per second³ [pps³].

The following absolute minimum and maximum values are valid:

Value Classes	Velocity	Acceleration	Bow	Clock
Registers	VMAX, VSTART, VSTOP, VBREAK	AMAX, DMAX, ASTART, DFINAL	BOW1, BOW2, BOW3, BOW4	CLK_FREQ (f_{CLK})
Minimum	3.906250 mpps	0.250000 mpps²	1 mpps³	4.194304 MHz
Maximum	8.388607 mpps ½ puls * f_{CLK}	4.194303 mpps²	16.777 mpps³	32MHz

SHORT AND STEEP RAMPS

For short and steep ramps higher acceleration/deceleration and bow values than usual are available by activating *direct_acc_val_en* and*direct_bow_val_en* (see*generator configuration register*, chapter 17.1). Set these parameters to 1 to change the units:

direct_acc_val_en=1 The values for*AMAX*, *DMAX*,*ASTART*, *DFINAL*, and *DSTOP* (see chapter9) are given as velocity value change per clock cycle with 24 bit unsigned decimal places(MSB =2^{-14}).

direct_bow_val_en=1 Bow values aregiven as acceleration value change per clock cycle. The values *BOW1*, *BOW2*, *BOW3*, and *BOW4* are 24 bit unsigned decimal places with the MSB defined as 2^{-29}.

EXAMPLE

With a clock frequency f_{CLK}=16 MHz the following maximum values are valid:

Value Classes	Acceleration (*direct_acc_val_en*=1)	Bow (*direct_bow_val_en*=1)
Registers	AMAX, DMAX, ASTART, DFINAL, DSTOP	BOW1, BOW2, BOW3, BOW4
Calculation	a[pps²] = (Δv/clk_cycle) / 2^{37} · f_{CLK}^2	bow[pps³] = (Δa/clk_cycle) / 2^{53} · f_{CLK}^3
Minimum	-1.86 kpps²	-454.75 kpps³
Maximum	-31.25 Gpps²	-7.63 Tpps³

9 Reference Switches

The reference input signals of the TMC4361 can be considered as a safety feature.The TMC4361 provides differentpossibilitiesfor reference switches and allowsfor appropriate settings for variousapplications.The TMC4361 offers two switches in hardware (STOPL, STOPR) and two additional virtual stop switches (VIRT_STOP_LEFT, VIRT_STOP_RIGHT). Additionally, a home reference switch is available.

PINS AND REGISTERS: REFERENCE SWITCHES

Pin names	Type		Remarks
STOPL	Input		Left reference switch
STOPR	Input		Right reference switch
HOME_REF	Input		Home switch
TARGET_REACHED	Output		Reference switch to indicate XACTUAL=XTARGET
Register name	Register address		Remarks
REFERENCE_CONF	0x01	RW	Configuration of interaction with reference pins
HOME_SAFETY_MARGIN	0x1E	RW	Region of uncertainty around X_HOME
DSTOP	0x2C	RW	Deceleration value if stop switches STOPL/STOPR or virtual stops are used with soft stop ramps.The deceleration value allows for an automatic linear stop ramp.
POS_COMP	0x32	RW	Free configurable compare position; signed; 32 bits
VIRT_STOP_LEFT	0x33	RW	Virtual left stop that triggers a stop event at XACTUAL ≤ VIRT_STOP_LEFT; signed; 32 bits
VIRT_STOP_RIGHT	0x34	RW	Virtual left stop that triggers a stop event at XACTUAL ≥ VIRT_STOP_RIGHT; signed; 32 bits
X_HOME	0x35	RW	Home reference position; signed; 32 bits
X_LATCH	0x36	RW	Stores XACTUAL at different conditions; signed; 32 bits

9.1 STOPL and STOPR

A left and a right stop switch are provided in hardware in order to stop the drive immediately, if one of them is triggered. Therefore, pin 12 and pin 14 of the motion controller have to be used. Both switches have to be enabled first:

- To use STOPL set *stop_left_en*=1. Now, the current velocity ramp stopsin case STOPL is equal to the chosen active polarity *pol_stop_left* and *VACTUAL* < 0.
- To use STOPR set *stop_right_en*=1. Now, STOPR stops the ramp in case the STOPR voltage level matches *pol_stop_right* and *VACTUAL* > 0.

The deceleration slope for stopping the ramp is influenced by *soft_stop_en*:

- Set *soft_stop_en*=0 for a hard and quick stop.
- Set soft_stop_en=1 to stop the ramp with a linear falling slope. In this case the deceleration factor is determined by *DSTOP.VSTOP*is not considered during the stop deceleration slope.

At the same time when a stop switch becomes active, the related status flag will be setand the particular event will be released. The flag remains set as long as the stop switch remains active. After reaching *VACTUAL*=0 due to the slope, further movement in the particular direction is not possible.

Driving on in the direction of areference switch is possible if the following conditions are met:

- The related status event is set back. The reference switch is not active anymore or alternatively, the related enabling switch (*stop_left_en*, *stop_right_en*) is reset to 0 (switched off) to go on driving in the - prior to that - closed direction.
- Stop eventsare cleared by reading out the *EVENTS* register. This is done automatically by the motion controller subsequent to an SPI datagram read request to this register. (There is only one exception to this if an event is selected for the *EVENT_CLEAR_CONF* register in order to inhibit the regular clearing.)