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.

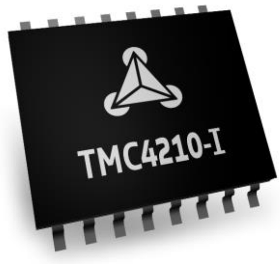
We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

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

TMC4210 DATASHEET

Low cost 1-Axis Stepper Motor Controller for TMC26x and TMC389 Stepper Driver
SPI Communication Interface for Microcontroller and Step/Direction interface to Driver



Applications

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

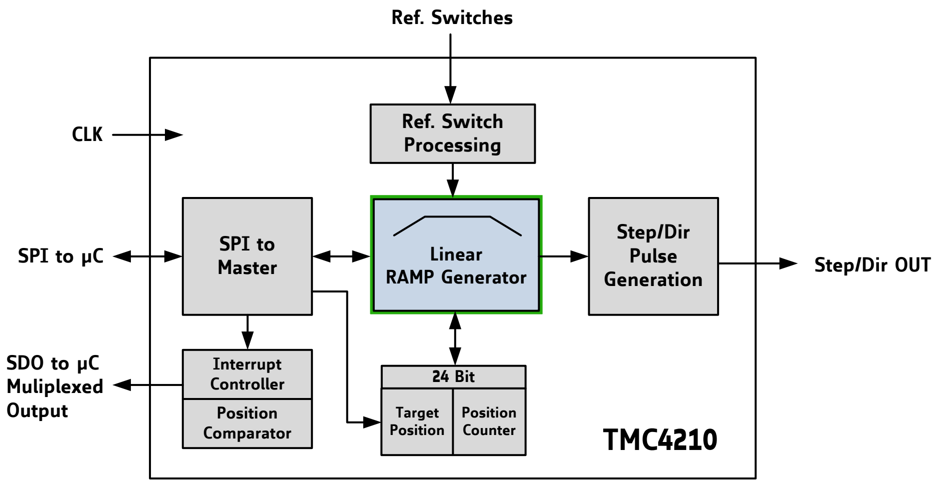
Features and Benefits

1-Axis stepper motor controller
3.3 V or 5 V operation with CMOS / TTL compatible IOs
Serial 4-wire interface for µC with easy-to-use protocol
Step/Direction interface to driver
Clock frequency: up to 32 MHz (can use CPU clock)
Internal position counters 24 bit wide
Microstep frequency up to 1 MHz
Read-out option for all motion parameters
Ramp generator for autonomous positioning / speed control
On-the-fly change of target motion parameters
Low power operation: 1.25 mA at 4 MHz (typ.)
Compact Size: ultra small 16 pin SSOP package
Directly controls TMC260, TMC261, TMC262, TMC2660, TMC389, TMC2100 and TMC2130

Description

The TMC4210 is a 1-axis miniaturized stepper motor controller with an industry leading feature set. It controls the motor via Step/Direction interface. Based on target positions and velocities - which can be altered on the fly - it performs all real time critical tasks autonomously. The TMC4210 offers high level control functions for robust and reliable operation. The 4 wire serial peripheral interface allows for communication with the microcontroller.
Together with a microcontroller the TMC4210 forms a complete motion control system. High integration and small form factor allow for miniaturized designs for cost-effective and highly competitive solutions.

Block Diagram



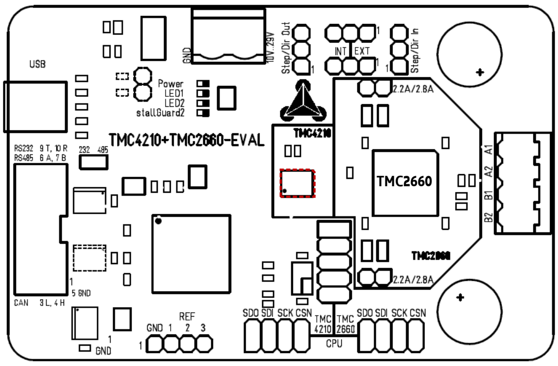
TRINAMIC Motion Control GmbH & Co. KG
Hamburg, Germany

TRINAMIC
MOTION CONTROL

APPLICATION EXAMPLE: RELIABLE CONTROL USING STEP/DIR

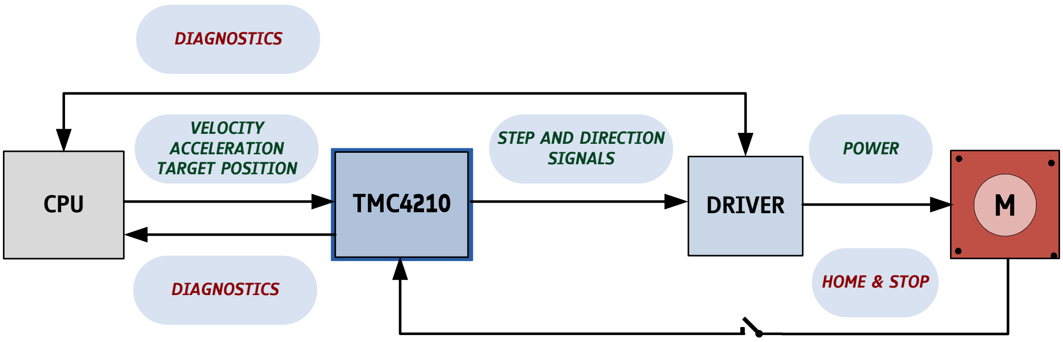
The TMC4210 scores with its autonomous handling of all real time critical tasks. By offloading the motion control function to the TMC4210, the stepper motor can be operated reliably with very little demand for service from the microcontroller. Software only needs to send target positions, and the TMC4210 generates precisely timed step pulses by hardware. Parameters for the motor can be changed on the fly while software retains full control. This way, high precision and reliable operation is achieved while costs are kept down.

TMC4210+TMC2660-EVAL Evaluation Board

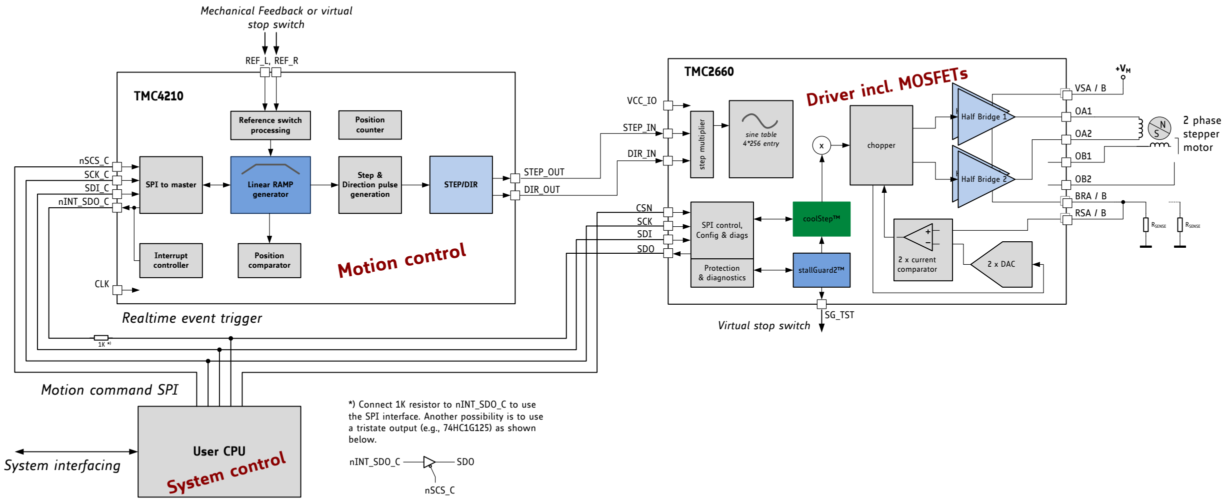


This evaluation board is a development platform for applications based on the TMC4210 and the TMC2660 stepper motor driver IC. The board features USB and CAN interfaces for communication with control software running on a PC. The power MOSFETs of the TMC2660 support drive currents up to 2.8A RMS at 29V.
The control software provides a user-friendly GUI for setting control parameters and visualizing the dynamic response of the motor.

Logic of the Controller/Driver Chain



System with TMC4210 and TMC2660



Order Codes

Order code	Description	Size
TMC4210-I	1-axis Step/Dir motion controller, SSOP16-package	6 x 5 mm^2
TMC4210+2660-EVAL	Evaluation board for TMC4210 and TMC2660 chipset	55 x 85 mm^2

Table of Contents

1 PRINCIPLES OF OPERATION 4

1.1 Key Concepts 4
1.2 Control Interfaces 5
1.3 Software Visibility 5
1.4 Step Frequencies 6
1.5 Moving the Motor 7

2 GENERAL DEFINITIONS, UNITS, AND NOTATIONS 8

2.1 Notations 8
2.2 Signal Polarities 8
2.3 Units of Motion Parameters 8
2.4 Representation of Signed Values by Two's Complement 8

3 PACKAGE 9

3.1 Package Outline 9
3.2 Signal Descriptions 9

4 SAMPLE CIRCUIT 11

5 CONTROL INTERFACE 12

5.1 Bus Signals 12
5.2 Serial Peripheral Interface for µC 12
5.3 Register Mapping 17

6 REGISTER DESCRIPTION 18

6.1 Axis Parameter Registers 18
6.2 Global Parameter Registers 34

7 REFERENCE SWITCH INPUTS 36

7.1 Reference Switch Configuration, *mot1r* 36
7.2 Triple Switch Configuration 36
7.3 Homing Procedure 37

8 STEP/DIR DRIVERS 38

8.1 Timing 38

9 RUNNING A MOTOR 39

9.1 Getting Started 39
9.2 Running a Motor with Start-Stop-Speed in *ramp_mode* 39

10 ON-CHIP VOLTAGE REGULATOR 40

11 POWER-ON RESET 41

12 ABSOLUTE MAXIMUM RATINGS 42

13 ELECTRICAL CHARACTERISTICS 42

13.1 Power Dissipation 42
13.2 DC Characteristics 43
13.3 Timing Characteristics 44

15 PACKAGE MACHANICAL DATA 45

15.1 Dimensional Drawings 45

16 MARKING 46

17 DISCLAIMER 47

18 ESD SENSITIVE DEVICE 47

19 TABLE OF FIGURES 48

20 REVISION HISTORY 48

21 REFERENCES 48

1 Principles of Operation

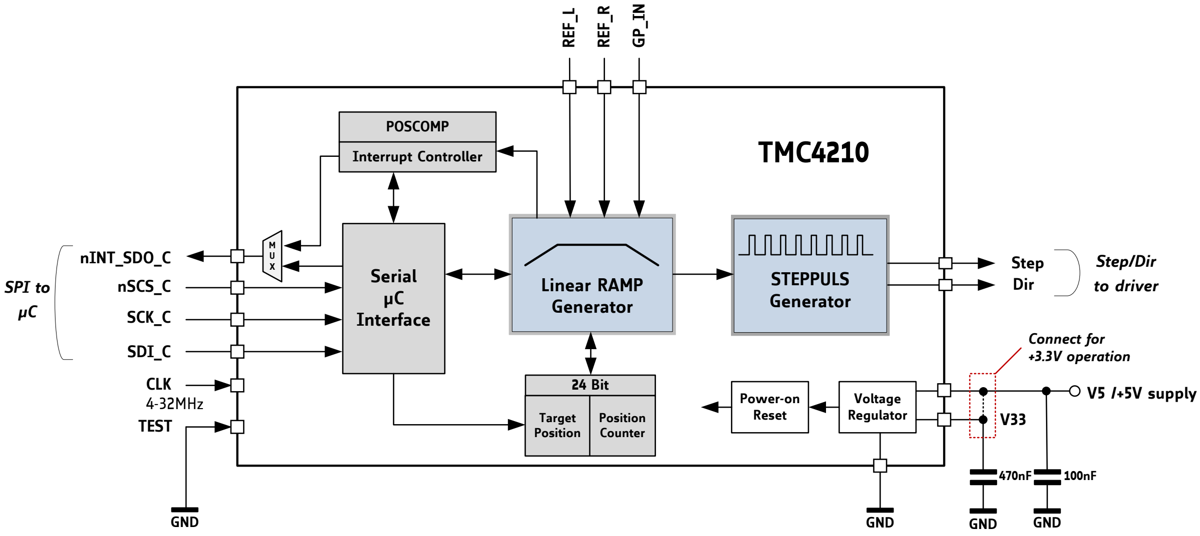


Figure 1.1 TMC4210 functional block diagram

The TMC4210 is a 1-axis miniaturized high performance stepper motor controller with an outstanding cost-performance ratio. It is designed for high volume automotive as well as for demanding industrial motion control applications. The TMC4210 receives target values for velocity, acceleration, and positioning from the microcontroller and calculates autonomously step and direction signals for the stepper motor driver IC. The motion controller is equipped with an SPI host interface with easy-to-use protocol and with a Step/Dir interface for addressing the stepper motor driver chip.

1.1 Key Concepts

The TMC4210 realizes real time critical tasks autonomously and guarantees for a robust and reliable drive. These following features contribute toward greater precision, greater efficiency, higher reliability, and smoother motion in many stepper motor applications.

Interfacing The TMC4210 provides an SPI interface for communication with the user CPU and a Step/Dir interface for driver interfacing.

Positioning The TMC4210 operates the motor based on user specified target positions and velocities. Modify all motion target parameters on-the-fly during motion.

Programming Every parameter can be changed at any time. The uniform access to any TMC4210 register simplifies application programming. A read-back option for all internal registers is available.

Microstepping Based on internal position counters the TMC4210 performs up to $\pm 2^{23}$ (micro)steps completely independent from the microcontroller. Via STEP/DIR signals any microstep resolution can be realized as supported by the driver.

1.2 Control Interfaces

1.2.1 Serial µC Interface

Using this interface, the TMC4210 receives target positions, target velocities, and target acceleration values for the microcontroller. Further, it is used for configuration.
From the software point of view, the TMC4210 provides a set of registers, accessed by the 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 TMC4210 it simultaneously receives a datagram from the TMC4210. This simplifies the communication with the TMC4210 and makes programming easy. Most microcontrollers have an SPI hardware interface, which directly connects to the serial four wire microcontroller interface of the TMC4210. For microcontrollers without SPI hardware software doing the serial communication is sufficient and can easily be implemented.

1.2.2 Step/Dir Driver Interface

The TMC4210-I controls the motor position by sending pulses on the STEP signal while indicating the direction on the DIR signal. A programmable step pulse length and step frequencies up to 1MHz 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 TMC4210-I perfectly fits to the TMC26x smart power Step/Dir driver family.

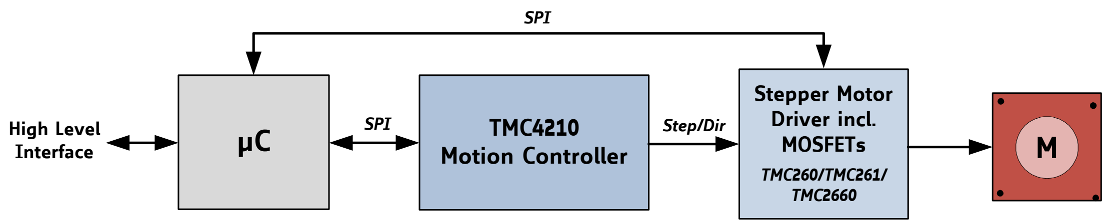


Figure 1.2 Application example using Step/Dir driver interface

1.3 Software Visibility

From the software point of view the TMC4210 provides a set of registers and on-chip RAM (see Figure 1.1), accessed via the serial µC interface in a uniform way. The serial interface uses a simple protocol with fixed datagram length for the read- and write-access. These registers are used for initializing the chip as required by the hardware configuration. Afterwards the motor can be moved by writing target positions or velocity and acceleration values.

1.4 Step Frequencies

INITIALIZE THE STEP/DIR INTERFACE!

The Step/Dir interface has to be initialized by writing 1 to *en_sd*. Refer to chapter 6.2.1.

The desired motor velocity is an important design parameter of an application. Therefore it is important to understand the limiting factors.

1.4.1 Step Frequencies using the Step/Dir Driver Interface

The step pulses can directly be fed to a Step/Dir driver. The maximum full step rate (fsf_{max}) depends on the microstep resolution of the external driver chip.

The TMC4210 microstep rate (µsf) is up to 1/32 of the clock frequency:

$$\mu sf_{max} = \frac{f_{CLK}}{32}$$

EXAMPLE FOR FULL STEP FREQUENCY CALCULATION

f_{CLK} = 16 MHz
μsf_{max} = 500 kHz
µstep resolution of external driver: 16

$$fsf_{max} = \frac{500\ kHz}{16} = 31.25\ kHz$$

With a standard motor with 1.8° per full step this results in up to 31.25kHz/200= 156 rotations per second, which is far above realistic motor velocities for this kind of motor and thus imposes no real limit on the application.

A 16 microsteps resolution can be extrapolated to 256 microsteps within the driver when using the TMC26x 2-phase stepper driver family or the TMC389 3-phase stepper motor driver.

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 TMC4210.
- To move a motor with a *new target 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.

1.5.2 Modes of Motion

ramp_mode	For positioning applications the *ramp_mode* is most suitable. The user sets the position and the TMC4210 calculates a trapezoidal velocity profile and drives autonomously to the target position. During motion, the position may be altered arbitrarily.
velocity_mode	For constant velocity applications the *velocity_mode* is most suitable. In *velocity_mode*, a target velocity is set by the user and the TMC4210 takes into account user defined limits of velocity and acceleration.
hold_mode	In *hold_mode*, the user sets target velocities, but the TMC4210 ignores any limits of velocity and acceleration, to realize arbitrary velocity profiles, controlled completely by the user.
soft_mode	The *soft_mode* is similar to the *ramp_mode*, but the decrease of the velocity during deceleration is done with a soft, exponentially shaped velocity profile.

1.5.3 Interrupts

The TMC4210 has capabilities to generate interrupts. Interrupts are based on ramp generator conditions which can be set using an interrupt mask. The interrupt controller (which continuously monitors reference switches and ramp generator conditions) generates an interrupt if required.

nINT_SDO_C is a low active interrupt signal while nSCS_C is high. If the microcontroller disables the interrupt during access to the TMC4210 and enables the interrupt otherwise, the multiplexed interrupt output of the TMC4210 behaves like a dedicated interrupt output. For polling, the TMC4210 sends the status of the interrupt signal to the microcontroller with each datagram.

1.5.4 Reference Switch Handling

The TMC4210 has a left (REF_L) and a right (REF_R) reference switch input. Further, the TMC4210 is equipped with a general purpose input (GP_IN).

INITIALIZE THE RIGHT REFERENCE SWITCH!

The right reference switch REF_R has to be initialized by writing 1 to *mot1r*.

1.5.5 Access to Status and Error Bits

The microcontroller directly controls and monitors the stepper driver. It also needs to take care for advanced current control, e.g. power down in stand still.

2 General Definitions, Units, and Notations

2.1 Notations

- *Decimal numbers* are used as usual without additional identification.
- *Binary numbers* are identified by a prefixed % character.
- *Hexadecimal numbers* are identified by a prefixed $ character.

EXAMPLE

Decimal: 42
Binary: %101010
Hexadecimal: $2A

TMC4210 DATAGRAMS ARE WRITTEN AS 32 BIT NUMBERS, E.G.:

$1234ABCD = %0001 0010 0011 0100 1010 1011 1100 1101

TWO TO THE POWER OF N

In addition to the basic arithmetic operators (+, -, *, /) the operator *two to the power of n* is required at different sections of this data sheet. For better readability instead of 2^n the notation 2^n is used.

2.2 Signal Polarities

External and internal signals are high active per default, but the polarity of some signals is programmable to be inverted. A pre-fixed lower case *n* indicates low active signals (e.g. *nSCS_C, nSCS_S*). See chapter 6.2, too.

2.3 Units of Motion Parameters

The motion parameters *position*, *velocity*, and *acceleration* are given as integer values within TMC4210 specific units. With a given stepper motor resolution one can calculate physical units for angle, angular velocity, angular acceleration. (See chapter 6.1.12)

2.4 Representation of Signed Values by Two's Complement

Motion parameters which have to cover negative and positive motion direction are processed as signed numbers represented by two's complement as usual. Limit motion parameters are represented as unsigned binary numbers.

SIGNED MOTION PARAMETERS ARE:

X_TARGET / X_ACTUAL / V_TARGET / V_ACTUAL / A_ACTUAL / A_THRESHOLD

UNSIGNED MOTION PARAMETERS ARE:

V_MIN / V_MAX / A_MAX

3 Package

The TMC4210 is qualified for the industrial temperature range. The package is RoHS compilant.

Order code	Package	Characteristics	JEDEC Drawing
TMC4210-I	SSOP16	150 mils, 16 pins, plastic package, industrial (-40… +85°C)	MO-137 (150 mils)

3.1 Package Outline

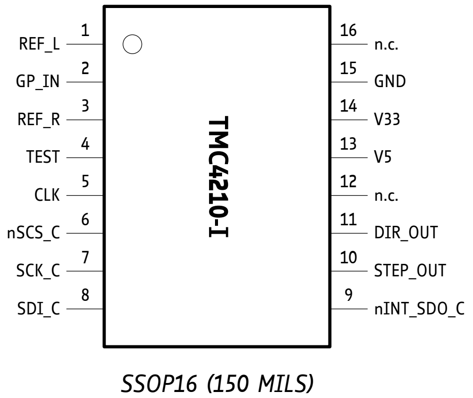


Figure 3.1 TMC4210 pin out

Please refer to the application note ***PCB_Guidelines_TRINAMIC_packages*** for a practical guideline for all available TRINAMIC IC packages and PCB footprints. The application note covers package dimensions, example footprints and general information on PCB footprints for these packages. It is available on www.trinamic.com.

3.2 Signal Descriptions

Pin	SSOP16	In/Out	Description
Reset	-	-	Internal power-on reset. *No external reset input pin is available.*
CLK	5	I	Clock input
nSCS_C	6	I	Low active SPI chip select input driven from µC
SCK_C	7	I	Serial data clock input driven from µC
SDI_C	8	I	Serial data input driven from µC
nINT_SDO_C	9	O	Serial data output to µC input / Multiplexed nINTERRUPT output if communication with µC is idle (resp. nSCS_C = 1) *SDO_C will never be high impedance*
n.c.	12, 16	-	Leave open
SCK_S	11	O	DIR output
SDO_S	10	O	STEP output
REF_L	1	I	Left reference/limit switch input. Pull to GND if not used. (no internal pull-up resistor)
GP_IN	2	I	General purpose input. Pull to GND if not used. (no internal pull-up resistor)
REF_R	3	I	Right reference/limit switch input. Pull to GND if not used. (no internal pull-up resistor)
V5	13		+5V supply / +3.3V supply
V33	14		470nF ceramic capacitor pin / +3.3V supply
GND	15		Ground
TEST	4	I	*Must be connected to GND as close as possible to the chip. No user function.*

Attention

- Preferably, long wires to the reference switch inputs and the general purpose input should be avoided. For long wires, a low pass filter for spike suppression should be provided (refer the TMC4210 evaluation board schematic as example).
- All inputs are Schmitt-Trigger. Unused inputs (REF_L, REF_R, and GP_IN) need to be connected to ground. Unused reference switch inputs have to be connected to ground, too.

4 Sample Circuit

This application example shows how to connect the TMC4210 motion controller with the processor and one out of TRINAMICs TMC260, TMC261, and TMC2660 stepper motor driver chips. These stepper motor driver chips have integrated MOSFETs. The TMC262 needs external power transistors.

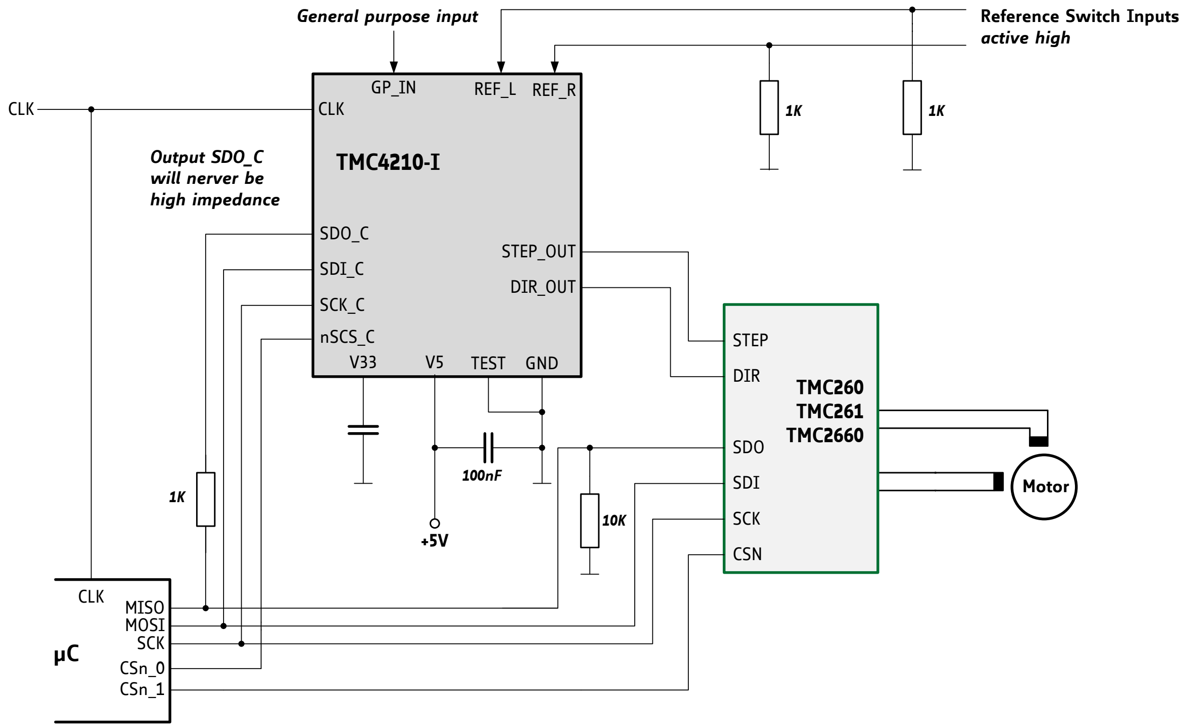


Figure 4.1 TMC4210 application environment with TMC260, TMC261 or TMC2660.

5 Control Interface

The communication takes place via a four wire serial interface and 32 bit datagrams of fixed length.

RESPONSIBILITIES ARE DEFINED AS FOLLOWS:

- The microcontroller is the master of the TMC4210. It initializes the motion controller and sets target values for velocity, acceleration, and positioning.
- The TMC4210 is the master of the stepper motor driver. The motion controller calculates, e.g., ramp profiles for positioning. It sends step and direction signals to the stepper motor driver.
- The microcontroller initializes the stepper motor driver. Further, the microcontroller can read out status and error flags and thus make the diagnostics.

AUTOMATIC POWER-ON RESET:

- The TMC4210 cannot be accessed before the power-on reset is completed and the clock is stable.
- All register bits are initialized with 0 during power-on reset, except the Step/Dir clock pre-devider STPDIV_4210 that is initialized with 15.

5.1 Bus Signals

Signal Description	TMC4210⟺Microcontroller
Bus clock input	SCK_C
Serial data input	SDI_C
Serial data output	SDO_C
Chip select input	nSCS_C

5.2 Serial Peripheral Interface for µC

The serial microcontroller interface of the TMC4210 acts as a 32 bit shift register.

COMMUNICATION BETWEEN µC AND THE TMC4210

1. The serial µC interface shifts serial data into SDI_C with each rising edge of the clock signal SCK_C.
2. Then, it copies the content of the 32 bit shift register into a buffer register with the rising edge of the selection signal nSCS_C.
3. The serial interface of the TMC4210 immediately sends back data read from registers or read from internal RAM via the signal SDO_C.
4. The signal SDO_C can be sampled with the rising edge of SCK_C. SDO_C becomes valid at least four CLK clock cycles after SCK_C becomes low as outlined in the timing diagram.

5.2.1 Timing

A complete serial datagram frame has a fixed length of 32 bit. Because of on-the-fly processing of the input data stream, the serial µC interface of the TMC4210 requires the serial data clock signal SCK_C to have a minimum low / high time of three clock cycles. The SPI signals from the µC interface may be asynchronous to the clock signal CLK of the TMC4210.

If the microcontroller and the TMC4210 work on different clock domains that run asynchronously by the timing of the SPI interface of the microcontroller should be made conservative in the way that the length of one SPI clock cycle equals 8 or more clock cycles of the TMC4210 clock CLK.

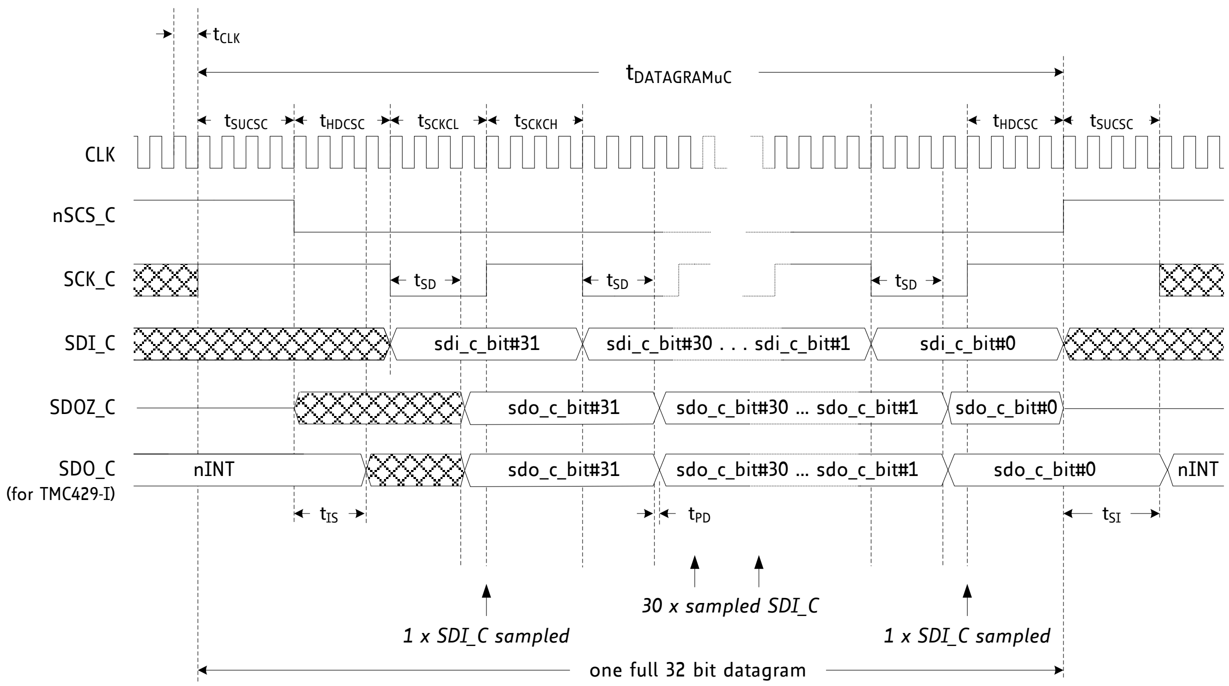


Figure 5.1 Timing diagram of the serial µC interface

EXPLANATORY NOTES

- While the data transmission from the microcontroller to the TMC4210 is idle, the low active serial chip select input nSCS_C and also the serial data clock signal SCK_C are set to high.
- While the signal nSCS_C is high, the TMC4210 assigns the status of the internal low active interrupt signal nINT to the serial data output SDO_C.
- The data signal SDI_C driven by the microcontroller has to be valid at the rising edge of the serial data clock input SCK_C. The maximum duration of the serial data clock period is unlimited.
- While the µC interface of the TMC4210 is idle, the SDO_C signal is the (active low) interrupt status nINT of the integrated interrupt controller of the TMC4210. The timing of the multiplexed interrupt status signal nINT is characterized by the parameters t_{IS} and t_{SI} (see chapter 13.3).

> The following SPI clock frequencies are recommended in order to avoid possible issues concerning the SPI frequency between microcontroller and TMC4210:
> - For fCLK = 16MHz an upper SPI clock frequency of 1MHz is recommended.
> - For fCLK = 32MHz an upper SPI clock frequency of 2MHz is recommended.

PROCEDURE OF DATA TRANSMISSION

1. The signal nSCS_C has to be high for at least three clock cycles before starting a datagram transmission. To initiate a transmission, the signal nSCS_C has to be set to low.
2. Three clock cycles later the serial data clock may go low.
3. The most significant bit (MSB) of a 32 bit wide datagram comes first and the least significant bit (LSB) is transmitted as the last one.
4. A data transmission is finished by setting nSCS_C high three or more CLK cycles after the last rising SCK_C slope.
5. So, nSCS_C and SCK_C change in opposite order from low to high at the end of a data transmission as these signals change from high to low at the beginning.

> In contrast to most other SPI compatible devices, the serial data output SDO_C of the TMC4210-I is always driven. It will never be high impedance Z. If high impedance is required for the SDO_C connected to the microcontroller, it can be realized using a single gate 74HCT1G125.

TIMING CHARACTERISTICS OF THE SERIAL MICROCONTROLLER INTERFACE					
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Unit**
tSUCSC	Setup Clocks for nSCS_C	3		∞	CLK periods
tHDCSC	Hold Clocks for nSCS_C	3		∞	CLK periods
tSCKCL	Serial Clock Low	3		∞	CLK periods
tSCKCH	Serial Clock High	3		∞	CLK periods
tSD	SDO_C valid after SCK_C low	2.5		3.5	CLK periods
tIS	nINTERRUPT status valid after nSCS_C low	2.5			CLK periods
tSI	SDO_C valid after nSCS_C high			4.5	CLK periods
tDAMAGRAMuC	Datagram Length	3+3+32*6= 198		∞	CLK periods
tDAMAGRAMuC	Datagram Length	12.375		∞	µs
fCLK	Clock Frequency	0		32	MHz
tCLK	Clock Period tCLK = 1 / fCLK	31.25		∞	ns
tPD	CLK-rising-edge-to-Output Propagation Delay		5		ns

5.2.2 Datagram Structure

The µC communicates with the TMC4210 via the four wire serial interface. Each datagram sent to the TMC4210 via the pin SDI_C and each datagram received from the TMC4210 via the pin SDO_C is 32 bits long.

The first bit sent is the *most significant bit (MSB)* sdi_c_bit#31. The last bit sent is the *least significant bit (LSB)* sdi_c_bit#0 (see Figure 5.1). During the reception of a datagram, the TMC4210 immediately sends back a datagram of the same length to the microcontroller. This return datagram consists of requested read data in the lower 24 datagram bits and status bits in the higher 8 datagram bits. A read request is distinguished from a write request by the read/not write datagram bit (RW).

5.2.2.1 Datagrams Sent to the TMC4210

The datagrams sent to the TMC4210 are assorted in four groups of bits:

RRS	The *register RAM select (RRS) bit* selects either registers or the on-chip RAM.
ADDRESS	*Address bits* address memory within the register set or within the RAM area.
RW	The *read / not write (RW) bit* distinguishes between read access and write access: read: RW = 1 / write RW = 0.
DATA	*Data bits* are only for write access. For read access these bits are not used (*don't care*) and should be set to 0.

MSB	**32 BIT DATAGRAM SENT FROM µC TO THE TMC4210 VIA PIN SDI_C**																														*LSB*
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
RRS	ADDRESS						*RW*	DATA																							

NOTE

- Different internal registers of the TMC4210 have different lengths. For some registers only a subset of 24 data bits is used.
- Unused data bits should be set to 0.
- Some addresses select a couple of registers mapped together into the 24 data bit space.

5.2.2.2 Datagrams Received by µC from the TMC4210

The datagrams received by the µC from the TMC4210 contain two groups of bits:

STATUS BITS The status bits, sent back with each datagram, comprehend the most important internal status bits of the TMC4210 and the settings of the reference switches

DATA BITS Data bits are only for write access.

The most significant bit *MSB* is received first; the least significant bit *LSB* is received last. The TMC4210 only sends datagrams on demand.

MSB	**32 BIT DATAGRAM SENT BACK FROM THE TMC4210 TO µC VIA PIN SDO_C**																														*LSB*
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
STATUS BITS								DATA BITS																							
INT	-	R_R	-	GP_IN	-	R_L	xEQt1																								

STATUS INFORMATION BITS

INT
The status bit *INT* is the *internal high active interrupt controller status output signal*. Handling of interrupt conditions without using interrupt techniques is possible by polling this status bit.
The interrupt signal is available multiplexed with the SPI read back data at the nINT_SDO_C pin of the TMC4210. The pin nINT_SDO_C may additionally be connected to an interrupt input of the microcontroller. Do not set SDO_INT=1 because this setting disables the SPI output.
Since the SDO_C / nINT output on TMC4210-I is multiplexed, the microcontroller has to disable its interrupt input while it sends a datagram to the TMC4210. The SDO_C signal driven by the TMC4210 alternates during datagram transmission.

R_L
The status bit R_L represents the state of the left reference switch input (*r_l*).

R_R
r_r is visible here only, while mot1R has not yet been set.

GP_IN
The GP_IN status bit represents the setting of the general purpose input. Refer to chapter 6.1.10.2, too.

xEQt1
The status bit xEQt1 indicates for the stepper motor, if it has reached its target position.

The status bits *r_r*, *r_l*, and *gp_in* and the bit xEQt1 can trigger an interrupt or enable simple polling techniques. See chapter 5.3, register 01 1110 for accessing the input bits.

5.2.3 Simple Datagram Examples

The % prefix – normally indicating binary representation in this data sheet – is omitted for the following datagram examples. Assuming, one would like to write (RW=0) to a register (RRS=0) at the address %001101 the following data word %0000 0000 0000 0001 0010 0011, one would have to send the following 32 bit datagram

0001101**0**000000000000000100100011

to the TMC4210. With inactive interrupt (**INT**=0), no cover datagram waiting (**CDGW**=0), all reference switches inactive (RS3=0, RS2=0, RS1=0), and all stepper motors at target position (xEQt3=1, xEQt2=1, xEQt1=1) the status bits would be %10010101 the TMC4210 would send back the 32 bit datagram:

10010101000000000000000000000000

To read (RW=**1**) back the register written before, one would have to send the 32 bit datagram

0001101**1**000000000000000000000000

to the TMC4210 and the TMC4210 would reply with the datagram

10010101000000000000000100100011.

Write (RW=0) access to on-chip RAM (RRS=1) to an address %111111 occurs similar to register access, but with RRS=1. To write two 6 bit data words %100001 and %100011 to successive pair-wise RAM addresses %1111110 and %1111111 (%100001 to %1111110 and %100011 to %1111111) which are commonly addressed by one datagram, one would have to send the datagram

1111111**0**000000000010001100100001.

To read (rw=**1**) from that on-chip memory address, one would have to send the datagram

1111111**1**000000000000000000000000.

5.3 Register Mapping

All register bits are initialized with 0 during power on reset, except the step pulse length setting that is initialized with 15. During power-up, the on-chip RAM of the TMC4210 is initialized internally and the chip does not send any datagrams to the stepper motor driver.

Changing Target Position or Target Velocity

The stepper motor is controlled directly by writing motion parameters into associated registers. Only one register write access is necessary for changing a target motion parameter. Thus the microcontroller has to send one 32 bit datagram to the TMC4210 for altering the target position or the target velocity of the stepper motor.

Read and Write

Read and write access is selected by the RW bit (sdi_c_bit#24) of the datagram sent from the µC to the TMC4210. The on-chip configuration RAM and the registers are writeable with read-back option. Some addresses are read-only. Write access (RW=0) to some of those read-only registers triggers additional functions, explained in detail later.

TMC4210 REGISTER MAPPING

32 BIT DATAGRAM SENT FROM µC TO THE TMC4210 VIA PIN SDI_C

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
	ADDRESS						RW	DATA																							
	SMDA		IDX					STEPPER MOTOR REGISTER SET (SMDA=00)																							
0	0	0	0	0	0	0	RW=0 : WRITE access / RW=1 : READ access	X_TARGET																							
			0	0	0	1		X_ACTUAL																							
			0	0	1	0															V_MIN										
			0	0	1	1															V_MAX										
			0	1	0	0														V_TARGET											
			0	1	0	1														V_ACTUAL											
			0	1	1	0															A_MAX										
			0	1	1	1														A_ACTUAL											
			1	0	0	1										1	PMUL											PDIV			
			1	0	1	0									lp					REF_CONF										R_M	
			1	0	1	1										INTERRUPT_MASK								INTERRUPF_FLAGS							
			1	1	0	0										PULSE_DIV				RAMP_DIV										0	
			1	1	0	1														DX_REF_TOLERANCE											
			1	1	1	0		X_LATCHED																							
			1	1	1	1																		USTEP_COUNT_4210							
			JDX					GLOBAL PARAMETER REGISTERS (SMDA=11)																							
	1	1	0	1	0	0																	IF_CONFIGURATION_4210								
			0	1	0	1		POS_COMP_4210																							
			0	1	1	0		POS_COMP_INT_4210															M								I
			1	0	0	0		POWER-DOWN																							
			1	0	0	1		TYPE_VERSION (= $429101 for TMC4210, read-only)																							
			1	1	1	0																				-	-	gp_in	-	r_l	r_r
			1	1	1	1				mot1r	0				0	0	0	0	0	STPDIV_4210				0	0	0	0	0	0	00	

SMDA	Stepper motor driver address	*r_l, r_r, gp_in*	Left switch / right switch / general purpose input (read out)
R_ M	*RAMP_MODE* mask	*I*	Interrupt
(shaded)	*Unused bits*		

6 Register Description

The TMC4210 provides axis parameter registers and global parameter registers.

6.1 Axis Parameter Registers

The registers hold binary coded numbers. Some are unsigned (positive) numbers, some are signed numbers in two's complement, and some are control bits or single flags. The functionality of different registers depends on the *RAMP_MODE* (refer to chapter 6.1.10).

OVERVIEW AXIS PARAMETER REGISTER MAPPING

REGISTER	R / W	TYPE	DESCRIPTION
X_TARGET	R/W	24 bit unsigned	This register holds the current target position in units of microsteps.
X_ACTUAL	R/W*[2]	24 bit unsigned	The current position of each stepper motor is available by read out of this register.
V_MIN	R/W	11 bit unsigned	This register holds the absolute velocity value at or below which the stepper motor can be stopped abruptly.
V_MAX	R/W	11 bit unsigned	This parameter sets the maximum motor velocity.
V_TARGET	R/W	12 bit signed	The *V_TARGET* register holds the current target velocity. The use of *V_TARGET* depends on the chosen mode of operation.
V_ACTUAL	R*[1]	12 bit signed	This read-only register holds the current velocity of the stepper motor.
A_MAX	R/W	11 bit unsigned	This register defines the absolute value of the desired acceleration for *velocity_mode* and *ramp_mode* (resp. *soft_mode*) with a value range from 0 to 2047.
A_ACTUAL	R	11 bit unsigned	The actual acceleration can be read out by the microcontroller from the *A_ACTUAL* read-only register.
PMUL *PDIV*	R/W R/W	1+7 bit 4 bit unsigned	These values form a floating point number with *PMUL* as mantissa and *PDIV* as exponent. *PMUL* and *PDIV* are used for calculating the deceleration ramp.
RAMP_MODE *REF_CONF* *lp*	R/W R/W R	2 bit 4 bit 1 bit	The two bits *RAMP_MODE* (*R_M*) select one of the four possible modes of operation. The configuration bits *REF_CONF* select the behavior of the reference switches. The bit called *lp* (latched position) is a read only status bit.
INTERRUPT_MASK *INTERRUPT_FLAGS*	R/W R/W	8 bit 8 bit	The TMC4210 provides one interrupt register of eight flags for the stepper motor.
RAMP_DIV *PULSE_DIV*	R/W R/W R/W	4 bit 4 bit 2 bit	The parameter *RAMP_DIV* scales the acceleration parameter *A_MAX.* The pulse generator clock – defining the maximum step pulse rate – is determined by the parameter *PULSE_DIV*. The parameter *PULSE_DIV* scales the velocity parameters.
DX_REF_TOLERANCE	R/W	12 bit	*DX_REF_TOLERANCE* excludes a motion range to allow motion near the reference position.
X_LATCHED	R	24 bit unsigned	This read-only register stores the actual position *X_ACTUAL* upon a change of the reference switch-state.
USTEP_COUNT_4210	R/W	8 bit	The read-write register *USTEP_COUNT_4210* holds the actual microstep pointer of the internal sequencer.

*[1] in *hold_mode* only, this register is a read-write register.

*[2] before overwriting *X_ACTUAL* choose *velocity_mode* or *hold_mode*. Refer to chapter 6.1.2.

6.1.1 *X_TARGET* (IDX=%0000)

This register holds the current target position in units of microsteps.

UNIT OF TARGET POSITION

The unit of the target position depends on the setting of the associated microstep resolution register *usrs*.

POSITIONING

- If the difference *X_TARGET* to *X_ACTUAL* is not zero and *R_M* = *ramp_mode* or *soft_mode*, the TMC4210 moves the stepper motor in the direction of *X_TARGET* in order to position *X_ACTUAL* to *X_TARGET*. Usually *X_TARGET* is modified to start a positioning.
- The condition | *X_TARGET* – *X_ACTUAL* | < 2^{23} must be satisfied for motion into correct direction.
- Target position *X_TARGET* and current position *X_ACTUAL* may be altered on the fly.
- To move from one position to another, the ramp generator of the TMC4210 automatically generates ramp profiles in consideration of the velocity limits *V_MIN* and *V_MAX* and acceleration limit *A_MAX*.

The registers *X_TARGET*, *X_ACTUAL*, *V_MIN*, *V_MAX*, and *A_MAX* are initialized with zero after power up.

6.1.2 *X_ACTUAL* (IDX=%0001)

The current position of the stepper motor is available by read out of the registers called *X_ACTUAL*. The actual position can be overwritten by the microcontroller. This feature is important for the reference switch position calibration controlled by the microcontroller.

UNIT OF CURRENT POSITION

The unit of the target position depends on the setting of the associated microstep resolution register *usrs*.

Attention
Before overwriting *X_ACTUAL* choose *velocity_mode* or *hold_mode*.
If *X_ACTUAL* is overwritten in *ramp_mode* or *soft_mode* the motor directly drives to *X_TARGET*.

6.1.3 *V_MIN* (IDX=%0010)

This register holds the absolute velocity value at or below which the stepper motor can be stopped abruptly.

UNIT OF VELOCITY

The unit of velocity parameters is *steps per time unit*. The scale of velocity parameters (*V_MIN*, *V_MAX*, *V_TARGET*, *V_ACTUAL*) is defined by the parameter *PULSE_DIV* (see page 6.1.12 for details) and depends on the clock frequency of the TMC4210.

DECELERATION

- The parameter *V_MIN* is relevant for deceleration while reaching a target position. *V_MIN* should be set greater than zero.
- This control value allows reaching the target position faster because the stepper motor is not slowed down below *V_MIN* before the target is reached.
- Due to the finite numerical representation of integral relations the target position cannot be reached exactly, if the calculated velocity is less than one, before the target is reached. Setting *V_MIN* to at least one assures reaching each target position exactly.

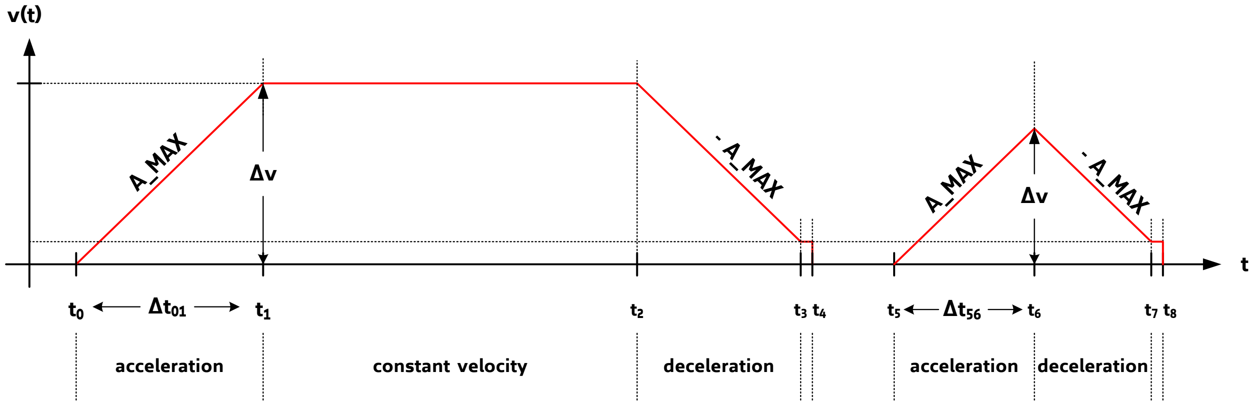


Figure 6.1 Velocity ramp parameters and velocity profiles

6.1.4 *V_MAX* (IDX=%0011)

This parameter sets the maximum motor velocity. The absolute value of the velocity will not exceed this limit, except if the limit *V_MAX* is changed during motion to a value below the current velocity.

UNIT OF VELOCITY

The unit of velocity parameters is *steps per time unit*. The scale of velocity parameters (*V_MIN*, *V_MAX*, *V_TARGET*, *V_ACTUAL*) is defined by the parameter *PULSE_DIV* (see page 6.1.12 for details) and depends on the clock frequency of the TMC4210.

HOMING PROCEDURE

To set target position *X_TARGET* and current position *X_ACTUAL* to an equivalent value (e.g. to set both to zero at a reference point) the stepper motor should be stopped first and the parameter *V_MAX* should be set to zero to hold the stepper motor at rest before writing into the register *X_TARGET* and *X_ACTUAL*.

Attention Before overwriting *X_ACTUAL* choose *velocity_mode* or *hold_mode*. If *X_ACTUAL* is overwritten in *ramp_mode* or *soft_mode* the motor directly drives to *X_TARGET*.

6.1.5 *V_TARGET* (IDX=%0100)

The use of *V_TARGET* depends on the chosen mode of operation:

Mode of operation	Functionality of *V_TARGET*
ramp_mode	The *V_TARGET* register holds the current target velocity calculated internally by the ramp generator.
velocity_mode	A target velocity can be written into the *V_TARGET* register. The stepper motor accelerates until it reaches the specified target velocity. The velocity is changed according to the motion parameter limits if the register *V_TARGET* is changed.
hold_mode	The register *V_TARGET* is ignored.
soft_mode	The *V_TARGET* register holds the current target velocity calculated internally by the ramp generator.

UNIT OF VELOCITY

The unit of velocity parameters is *steps per time unit*. The scale of velocity parameters (*V_MIN*, *V_MAX*, *V_TARGET*, *V_ACTUAL*) is defined by the parameter *PULSE_DIV* (see chapter 6.1.12 for details) and depends on the clock frequency of the TMC4210.

6.1.6 *V_ACTUAL* (IDX=%0101)

This read-only register holds the current velocity of the associated stepper motor. Internally, the ramp generator of the TMC4210 processes with 20 bits while only 12 bits (the most significant bits) can be read out as *V_ACTUAL*.
In *hold_mode* only, this register is a read-write register. Writing zero to the register *V_ACTUAL* immediately stops the associated stepper motor, because hidden bits are set to zero with each write access to the register *V_ACTUAL*. In *hold_mode* motion parameters are ignored and the microcontroller has the full control to generate a ramp. The TMC4210 only handles the microstepping and datagram generation for the associated stepper motor.

UNIT

The unit of velocity parameters is *steps per time unit*. The scale of velocity parameters (*V_MIN*, *V_MAX*, *V_TARGET*, and *V_ACTUAL*) is defined by the parameter *PULSE_DIV* (see chapter 6.1.12 for details) and depends on the clock frequency of the TMC4210.

An actual velocity of zero read out by the microcontroller means that the *current velocity is in an interval between zero and one*. Therefore the actual velocity should not be used to detect a stop of the stepper motor. It is advised to detect the *target_reached* flag instead.

6.1.7 *A_MAX* (IDX=%0110)

This register defines the absolute value of the desired acceleration for *velocity_mode* and *ramp_mode* (resp. *soft_mode*) with a value range from 0 to 2047.

Note
The motion controller cannot stop the stepper motor if *A_MAX* is set to zero on the fly because afterwards the velocity cannot be changed automatically any more.

UNIT

The unit of the acceleration is *change of step frequency per time unit divided by 256*. The scale of acceleration parameters (*A_MAX*, *A_ACTUAL*, and *A_THRESHOLD*) is defined by the parameter *RAMP_DIV* (see section 6.1.12) and depends on the clock frequency of the TMC4210.

6.1.7.1 *A_MAX* in *ramp_mode*

As long as *RAMP_DIV* ≥ *PULSE_DIV* – 1 is valid, any value of *A_MAX* within its range (0… 2047) is allowed and there exists a valid pair {*PMUL*, *PDIV*} for each *A_MAX*. The reason is that the acceleration scaling determined by *RAMP_DIV* is compatible with the step velocity scaling determined by *PULSE_DIV*. A large *RAMP_DIV* stands for low acceleration and a large *PULSE_DIV* stands for low velocity. Low acceleration is compatible with low speed and high speed as well, but high acceleration is more compatible with high speed.

Changing one parameter out of the triple {*A_MAX*, *RAMP_DIV*, *PULSE_DIV*} requires re-calculation of the parameter pair {*PMUL*, *PDIV*} to update the associated register. For description of the parameters *PMUL* and *PDIV* see section 6.1.9.

6.1.7.1.1 Deceleration in *ramp_mode* and *soft_mode*

If *RAMP_DIV* and *PULSE_DIV* differ more than one while deceleration in *ramp_mode* or *soft_mode* the parameter *A_MAX* needs to have a lower limit (>1) and an upper limit (<2047). The reason is that the deceleration ramp is internally limited to 2^{19} steps (respectively microsteps).

THE LOWER LIMIT OF *A_MAX* IS GIVEN BY

$$A\_MAX_{LOWER\_LIMIT} = 2^{(RAMP\_DIV - PULSE\_DIV - 1)}$$

- With *V_MAX* set to $\frac{2048}{\sqrt{2}}$ (≈ 1448) or lower the $A\_MAX_{LOWER\_LIMIT}$ is half of this value.
- If *RAMP_DIV* – *PULSE_DIV* – 1 ≤ 0 the limit $A\_MAX_{LOWER\_LIMIT}$ is 1 and the parameter *A_MAX* may be set to 1.

THE UPPER LIMIT OF *A_MAX* IS GIVEN BY

$$A\_MAX_{UPPER\_LIMIT} = 2^{(RAMP\_DIV - PULSE\_DIV + 12)} - 1$$

- If *RAMP_DIV* – *PULSE_DIV* + 1 ≥ 0 the *A_MAX_UPPER_LIMIT* is > 2048 and the parameter *A_MAX* might be set to any value up to 2047.

CONDITIONS

The parameter *A_MAX* must not be set below $A\_MAX_{LOWER\_LIMIT}$ except *A_MAX* is set to 0. The condition $A\_MAX \geq A\_MAX_{LOWER\_LIMIT}$ as well as $A\_MAX \leq A\_MAX_{UPPER\_LIMIT}$ must be satisfied to reach any target position without oscillations. If that condition is not satisfied, oscillations around a target position may occur.

6.1.8 *A_ACTUAL* (IDX=%0111)

The actual acceleration can be read out by the microcontroller from the *A_ACTUAL* read-only register. The actual acceleration is used to select scale factors for the coil currents. It is updated with each clock. The returned value *A_ACTUAL* is smoothed to avoid oscillations of the readout value. Thus, returned *A_ACTUAL* values should not be used directly for precise calculations.

UNIT

The unit of the acceleration is *change of step frequency per time unit divided by 256*. The scale of acceleration parameters (*A_MAX*, *A_ACTUAL*, and *A_THRESHOLD*) is defined by the parameter *RAMP_DIV* (see section 6.1.12) and depends on the clock frequency of the TMC4210.

6.1.9 *PMUL & PDIV* (IDX=%1001)

In ramp mode, the TMC4210 uses an internal algorithm to calculate the deceleration ramp on the fly. This algorithm requires an additional proportionality factor *P* which allows the TMC4210 to calculate the velocity required for stopping in time to exactly reach the target position without overshooting. This calculation is done for each ramp step. The result of this calculation can be read in the register *V_TARGET*. Whenever *V_TARGET* falls below the actual velocity, the TMC4210 decelerates. As there is a large range of acceleration and velocity values, *p* is stored in a floating point representation, using the registers *PMUL* (mantissa) and *PDIV* (exponent).
Using the *proportionality factor P* target positions are quickly reached without overshooting. The proportionality factor primarily depends on the acceleration limit *A_MAX* and on the two clock divider parameters *PULSE_DIV* and *RAMP_DIV*. These two separate clock divider parameters (set to the same value for most applications) provide an extremely wide dynamic range for acceleration and velocity. *PULSE_DIV* and *RAMP_DIV* allow reaching very high velocities with very low acceleration.

Changing one parameter out of the triple {*A_MAX*, *RAMP_DIV*, *PULSE_DIV*} requires re-calculation of the parameter pair {*PMUL*, *PDIV*} to update the associated register.

6.1.9.1 Calculation of the Proportionality Factor *p*

The representation of the proportionality factor *p* by the two parameters *PMUL* and *PDIV* is a floating point representation.

NOTATIONS

Registers are *PMUL* and *PDIV*.
Operating values are P_{MUL} and P_{DIV}.

CALCULATE *P* AS FOLLOWS:

$$p = \frac{P_{MUL}}{P_{DIV}}$$

with

P_{MUL} = 128… 255 representing a factor of 1.000 to 1.992 (=1+127/128)
P_{DIV} = $\{2^3, 2^4, 2^5… 2^{14}, 2^{15}, 2^{16}\}$

P_{MUL} ranges from 128 to 255. P_{DIV} is a power of two with a range from 8 to 65536. Values of *p* less than 128 can be achieved by increasing P_{DIV}.

The TMC4210 does not directly store the P_{DIV} parameter. The motion controller stores *PDIV* with

$$P_{DIV} = 2^{3+PDIV}$$

NOTE

- Setting the factor *p* too small will result in a slow approach to the target position.
- Setting the factor *p* too large will cause overshooting and even oscillations around the target position.
- The parameters *PMUL* and *PDIV* share the address IDX=%1001. The MSB of *PMUL* is fixed set to 1 and cannot be changed. This way, *PMUL* represents a mantissa in the range 1.000 (%1000 0000) to 1.992 (%1111 1111).

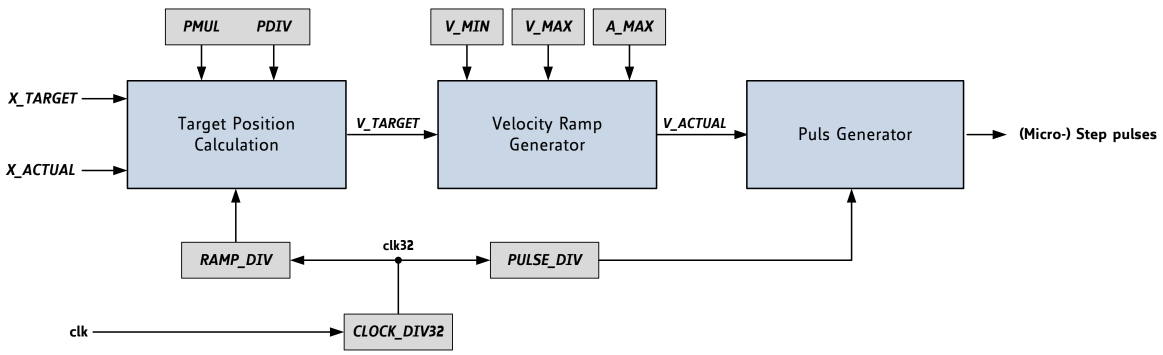


Figure 6.2 Target position calculation, ramp generator, and pulse generator

6.1.9.1.1 Calculation of *p* for a Given Acceleration

p and the fitting *PMUL* and *PDIV* values can be calculated by the microcontroller. Optionally a pair of matching values of *A_MAX*, *PMUL* and *PDIV* can be stored into the microcontroller memory. The acceleration limit is a stepper motor parameter which is fixed in most applications. If the acceleration limit has to be changed nevertheless, the microcontroller can calculate a pair of *PMUL* and *PDIV* on demand for each new acceleration limit *A_MAX* with *RAMP_DIV* and *PULSE_DIV*. Also, pre-calculated pairs of *PMUL* and *PDIV* read from a table can be sufficient.

6.1.9.2 Calculation of *PMUL* and *PDIV*

A pair of *PMUL* and *PDIV* has to be calculated for each provided acceleration limit *A_MAX*. Note, that there may be more than one valid pair of *PMUL* and *PDIV* for a given *A_MAX* acceleration limit.

CONSIDERATIONS FOR THE CALCULATION OF *PMUL* AND *PDIV*

- To accelerate, the ramp generator accumulates the acceleration value to the actual velocity with each time step.
- The absolute value *V_MAX* of the velocity internally is represented by 11+8=19 bits, while only the most significant 11 bits and the sign are used as input for the step pulse generator. So, there are 2^{11}=2048 values possible for specifying a velocity within a range of 0 to 2047.
- The ramp generator accumulates 1/256**A_MAX* with each time step to the actual velocity value *V_ACTUAL* during acceleration phases. This accumulation uses 8 bits for decimals. So, the acceleration from a velocity *V_ACTUAL*=0 to the maximum possible velocity *V_MAX*=2047 spans over 2048*256 / *A_MAX* pulse generator clock pulses.
- Within the acceleration phase the pulse generator generates S = ½ * 2048* 256 / *A_MAX* * T steps for the (micro) step unit.
- The parameter T is the clock divider ratio: T = $2^{RAMP\_DIV} / 2^{PULSE\_DIV} = 2^{RAMP\_DIV-PULSE\_DIV}$

During the acceleration, the velocity has to be increased until the velocity limit *V_MAX* is reached or deceleration is required in order to exactly reach the target position. The TMC4210 automatically determines the deceleration position in *ramp_mode* and decelerates. This calculation uses the difference between current position and target position and the proportionality parameter *p*, which has to be *p* = 2048 / S.
The following formula results:

$$p = \frac{2048}{\left(\frac{1}{2} * 2048 * \frac{256}{A_{MAX}}\right) * 2^{RAMP\_DIV-PULSE\_DIV}}$$

This can be simplified to

$$p = \frac{A\_MAX}{128 * 2^{RAMP\_DIV-PULSE\_DIV}}$$

HINTS

- To avoid overshooting, the parameter *PMUL* should be made approximately 1% smaller than calculated. Alternatively set *p* reduced by an amount of 1%.
- If the proportionality parameter *p* is too small, the target position will be reached slower, because the slow down ramp starts earlier. The target position is approached with minimal velocity *V_MIN*, whenever the internally calculated target velocity becomes less than *V_MIN*.
- With a good parameter *p* the minimal velocity *V_MIN* is reached a couple of steps before the target position.
- With parameter *p* set a little bit too large and a small *V_MIN* overshooting of one step (respectively one microstep) may occur. A decrement of the parameter *PMUL* avoids this one-step overshooting.

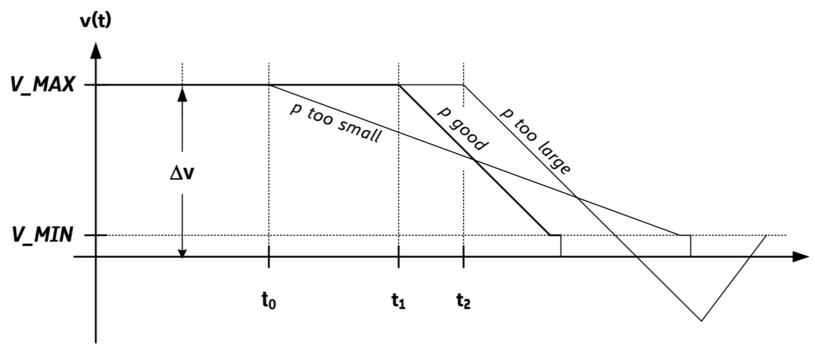


Figure 6.3 Proportionality parameter *p* and outline of velocity profile(s)

6.1.9.2.1 Choosing a Pair of *PMUL* and *PDIV*

The calculation is based on the formula

$$p = \frac{P_{MUL}}{P_{DIV}} = \frac{PMUL}{2^{3+PDIV}}$$

CALCULATIONS

1. To represent the parameter *p* choose a pair of *PMUL* and *PDIV* which approximates *p*.
2. Value range for *PMUL*: 128… 255
3. Value range for *PDIV*: one out of {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13} (representing P_{DIV} one out of {8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32786, 65536})
4. Try all 128 * 14 = 1792 possible pairs of *PMUL* and *PDIV* with a program and choose a matching pair.
5. To find a pair, calculate for each pair of *PMUL* and *PDIV*

   $p = \frac{A\_MAX}{128 * 2^{RAMP\_DIV-PULSE\_DIV}}$ and

   $p' = \frac{P_{MUL}}{P_{DIV}} = \frac{PMUL}{2^{3+PDIV}}$ and

   $q = \frac{p'}{p}$
6. Select one of the pairs satisfying the condition 0.95 < q < 1.0. The value *q* interpreted as a function *q*(a_max, ramp_div, pulse_div, pmul, pdiv) gives the *quality criterion* required.

Although q = 1.0 indicates that the chosen *P_MUL* and *P_DIV* perfectly represent the desired *p* factor for a given *A_MAX*, overshooting could result because of finite numerical precision. On the other hand in case of high resolution microstepping, overshooting of one microstep is negligible in most applications.
To avoid overshooting, use *P_MUL-1* instead of the selected *P_MUL* or select a pair (*P_MUL*, *P_DIV*) with q = 0.99.

6.1.9.2.2 Optimized Calculation of *PMUL* and *PDIV*

The calculation of the parameters *PMUL* and *PDIV* can be simplified using the expression

$$PMUL = p * 2^3 * 2^{PDIV} \quad \text{with} \quad p = \frac{A\_MAX}{128 * 2^{RAMP\_DIV-PULSE\_DIV}}$$

To avoid overshooting, use

$$p_{reduced} = p * (1 - p_{reduction}[\%])$$ with *p_reduction* approximately 1%

This results in:

$$PMUL = p_{reduced} * 2^3 * 2^{PDIV} = 0.99 * p * 2^3 * 2^{PDIV}$$

PMUL becomes a function of the parameter *PDIV*. To find a valid pair {*PMUL*, *PDIV*} choose one out of 14 pairs for *PDIV* = {0, 1, 2, 3, …, 13} with *PMUL* within the valid range 128 ≤ *PMUL* ≤ 255.

The C language example *pmulpdiv.c* can be found on www.trinamic.com. The source code can directly be copied from the PDF datasheet file.