TMC4330A DATASHEET

TMC4330A Document Revision 1.00 • 2016-NOV-25

The S-ramp and sixPoint™ ramp motion controller for stepper motors is optimized for high velocities, allowing on-the-fly changes. TMC4330A offers Step/Dir interfaces, as well as an encoder interface for closed-loop operation.

Figure 1: Sample Image
TMC4330A Closed-Loop Drive

**Marking details are explained on page 159.*

Features

- SPI Interfaces for µC with easy-to-use protocol.
- Encoder interface for incremental or serial encoders.
- Closed-loop operation for Step drivers.
- Internal ramp generator generating S-shaped ramps or sixPoint™ ramps supporting on-the-fly changes.
- Controlled PWM output.
- Reference switch handling.
- Hardware and virtual stop switches.

Applications

- Textile, sewing machines
- CCTV, security
- Printers, scanners
- ATM, cash recycler
- Office automation
- POS
- Factory automation
- Lab automation
- Pumps and valves
- Heliostat controllers
- CNC machines
- Robotics

Block Diagram: TMC4330A Interfaces & Features

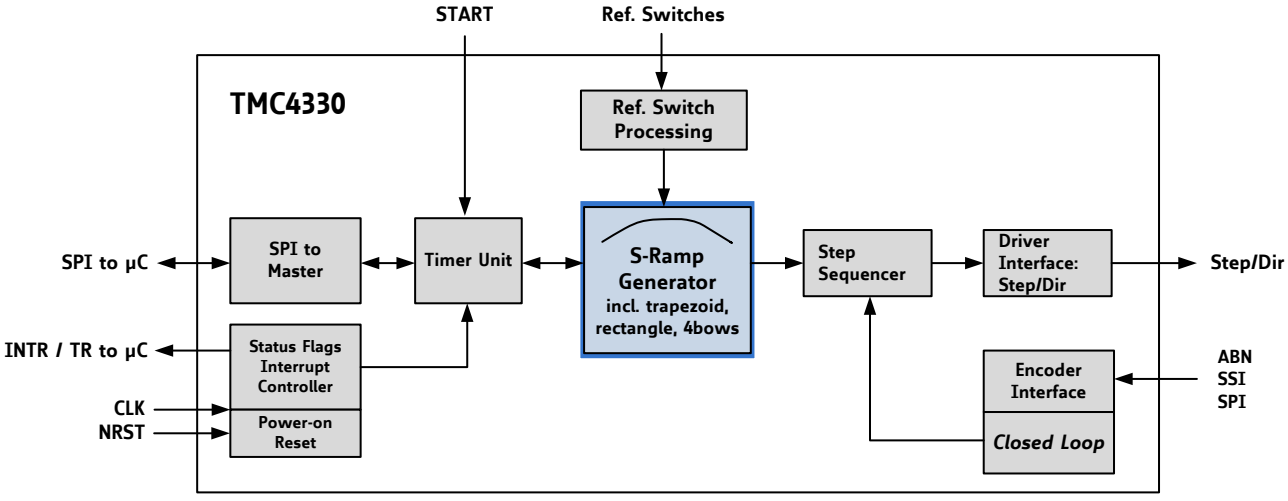


Figure 2: Block Diagram

Read entire documentation; especially the Supplemental Directives in chapter 17 (page 160).

TRINAMIC MOTION CONTROL

Functional Scope of TMC4330A

TMC4330A is a miniaturized high-performance motion controller for stepper motor drivers, particularly designed for fast and jerk-limited motion profile applications with a wide range of ramp profiles. The S-shaped or sixPoint™ velocity profile, closed-loop and open-loop features offer many configuration options to suit the user's specifications, as presented below:

S-Shaped Velocity Profile

S-shaped ramp profiles are jerk-free. Seven ramp segments form the S-shaped ramp that can be optimally adapted to suit the user's requirements. High torque with high velocities can be reached by calibrating the bows of the ramp, as explained in this user manual.

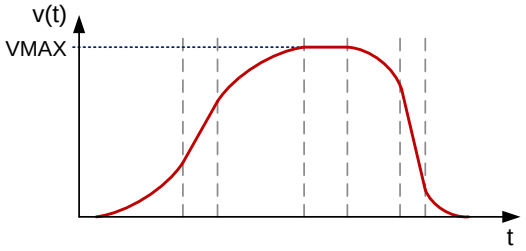


Figure 3: S-shaped Velocity Profile

i More information on ramp configurations and other velocity profiles, e.g. sixPoint™ ramps, are provided in chapter 6 (Page 24).

Closed-loop Operation Feature

A typical hardware setup for closed-loop operation with a TMC220x/222x stepper motor driver is shown in the figure below.

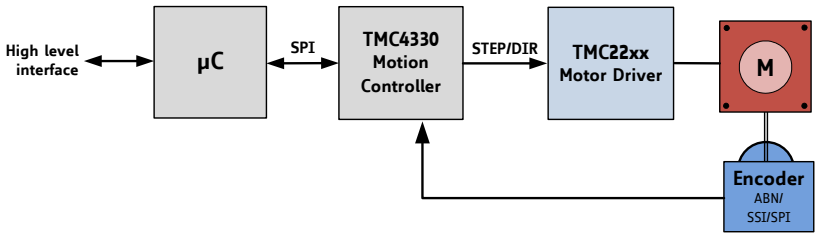


Figure 4: Hardware Set-up for Closed-loop Operation with TMC220x/222x

Reference Switch Support

A typical hardware setup for open-loop operation with enhanced modifications, by use of external stop switches with the TMC2100 motor stepper driver is shown below. Home switches with different configurations are also supported.

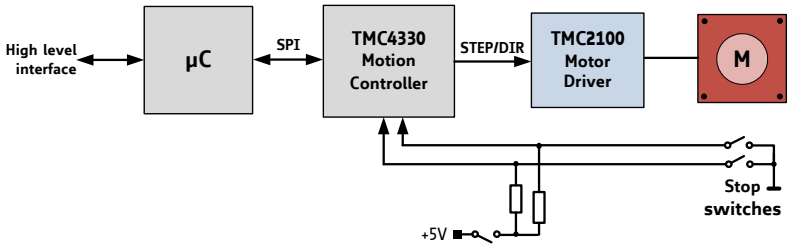


Figure 5: Hardware Set-up for Open-loop Operation with TMC2100 supporting External Stop Switches

Order Codes

Order code	Description	Size
TMC4330A-LA	*Motion controller with closed-loop features, QFN32*	*4 x 4 mm²*

Table 1: TMC4330A Order Codes

TABLE OF CONTENTS

Read entire documentation; especially the "Supplemental Directives" on page 160.

Read entire documentation; especially the "Supplemental Directives" on page 160.

Read entire documentation; especially the "Supplemental Directives" on page 160.

Read entire documentation; especially the "Supplemental Directives" on page 160.

Read entire documentation; especially the "Supplemental Directives" on page 160.

MAIN MANUAL

1. Pinning and Design-In Process Information

In this chapter you are provided with a list of all pin names and a functional description of each.

1.1. Pin Assignment: Top View

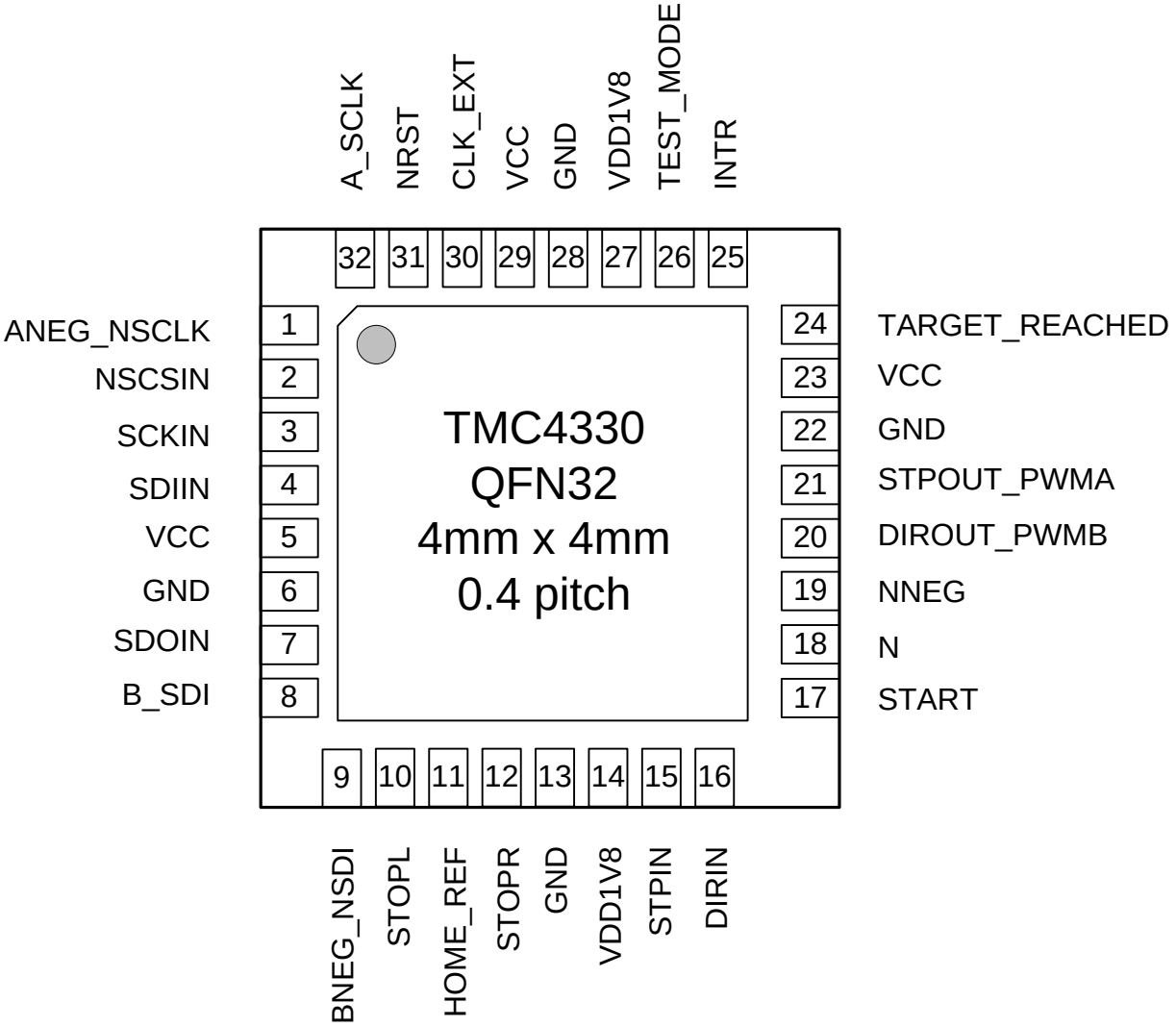


Figure 6: Package Outline: Pin Assignments Top View

Read entire documentation; especially the "Supplemental Directives" on page 160.

1.2. Pin Description

Pin Names and Descriptions			
Pin	**Number**	**Type**	**Function**
Supply Pins			
GND	6, 13, 22, 28	GND	Digital ground pin for IOs and digital circuitry.
VCC	5, 23, 29	VCC	Digital power supply for IOs and digital circuitry (3.3V… 5V).
VDD1V8	14, 27	VDD	Connection of internal generated core voltage of 1.8V.
CLK_EXT	30	I	Clock input to provide a clock with the frequency fCLK for all internal operations.
NRST	31	I (PU)	Low active reset. If not connected, Power-on-Reset and internal pull-up resistor is active.
TEST_MODE	26	I	Test mode input. Tie to low for normal operation.
Interface Pins for µC			
NSCSIN	2	I	Low active chip selects input of SPI interface to µC.
SCKIN	3	I	Serial clock for SPI interface to µC.
SDIIN	4	I	Serial data input of SPI interface to µC.
SDOIN	7	O	Serial data output of SPI interface to µC (Z if NSCSIN=1).
INTR	25	O	Interrupt output, programmable PD/PU for wired-and/or.
TARGET_REACHED	24	O	Target reached output, programmable PD/PU for wired-and/or.
Reference Pins			
STOPL	10	I (PD)	Left stop switch. External signal to stop a ramp. If not connected, internal pull-down resistor is active.
HOME_REF	11	I (PD)	Home reference signal input. External signal for reference search. If not connected, internal pull-down resistor is active.
STOPR	12	I (PD)	Right stop switch. External signal to stop a ramp. If not connected, internal pull-down resistor is active.
STPIN	15	I (PD)	Step input for external step control. If not connected, internal pull-down resistor is active.
DIRIN	16	I (PD)	Direction input for external step control. If not connected, internal pull-down resistor is active.
START	17	IO	Start signal input/output.
S/D Output Pins			
STPOUT PWMA	21	O	Step output. First PWM signal (Sine).
DIROUT PWMB	20	O	Direction output. Second PWM signal (Cosine).

•→ *Continued on next page!*

Pin Names and Descriptions			
Pin	**Number**	**Type**	**Function**
Encoder Interface Pins			
N	18	I (PD)	N signal input of incremental encoder input interface. If not connected, internal pull-down resistor will be active.
NNEG	19	I (PD)	Negated N signal input of incremental encoder input interface. If not connected, internal pull-down resistor will be active.
B SDI	8	I (PD)	B signal input of incremental encoder input interface. Serial data input signal of serial encoder interface (SSI/SPI). If not connected, internal pull-down resistor is active.
BNEG NSDI SDO_ENC	9	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	32	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.

Table 2: Pin Names and Descriptions

Read entire documentation; especially the "Supplemental Directives" on page 160.

1.3. System Overview

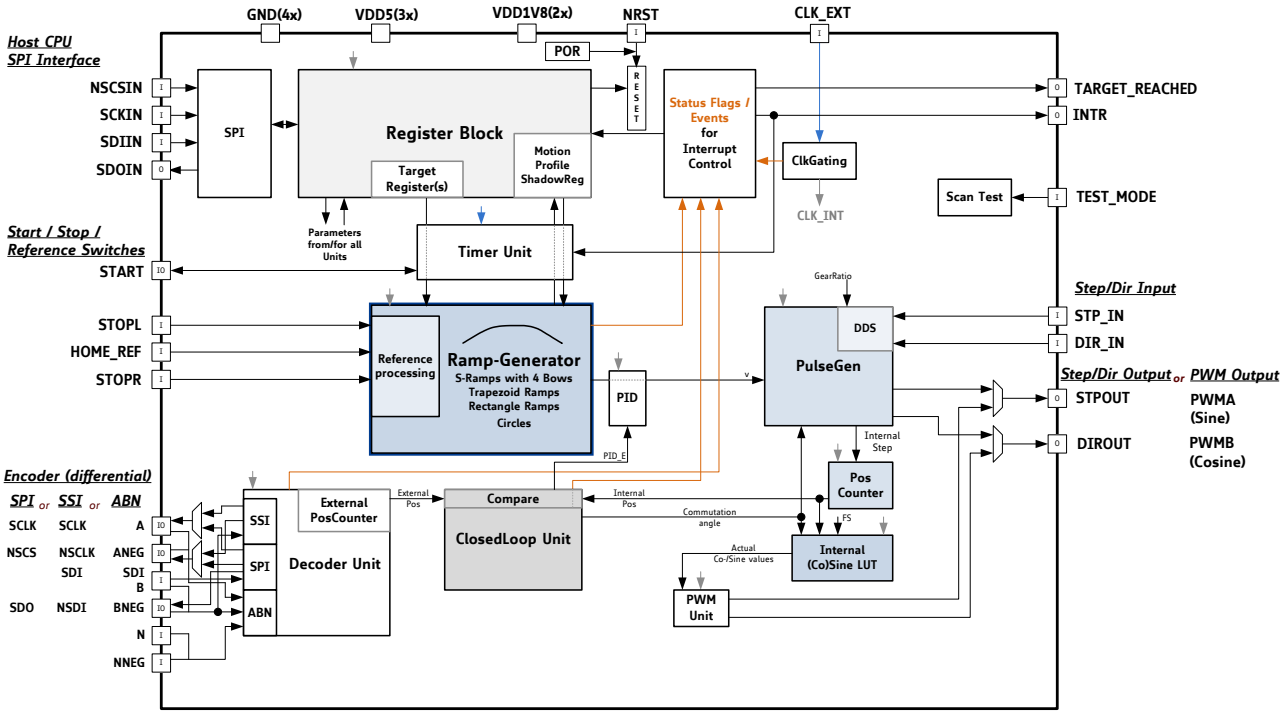


Figure 7: System Overview

Read entire documentation; especially the "Supplemental Directives" on page 160.

2. Application Circuits

In this chapter application circuit examples are provided that show how external components can be connected.

2.1. TMC4330A Standard Connection: VCC=3.3V

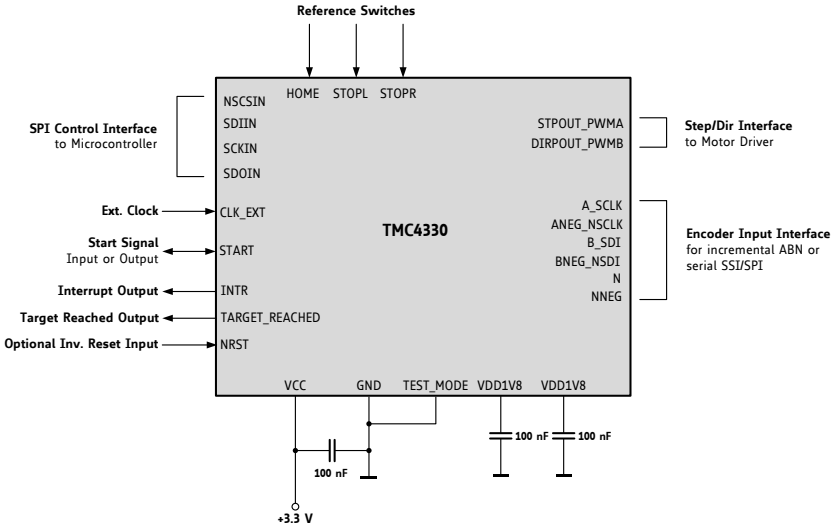


Figure 8: TMC4330A Connection: VCC=3.3V

2.2. TMC4330A with TMC2100 Stepper Connection and Encoder feedback

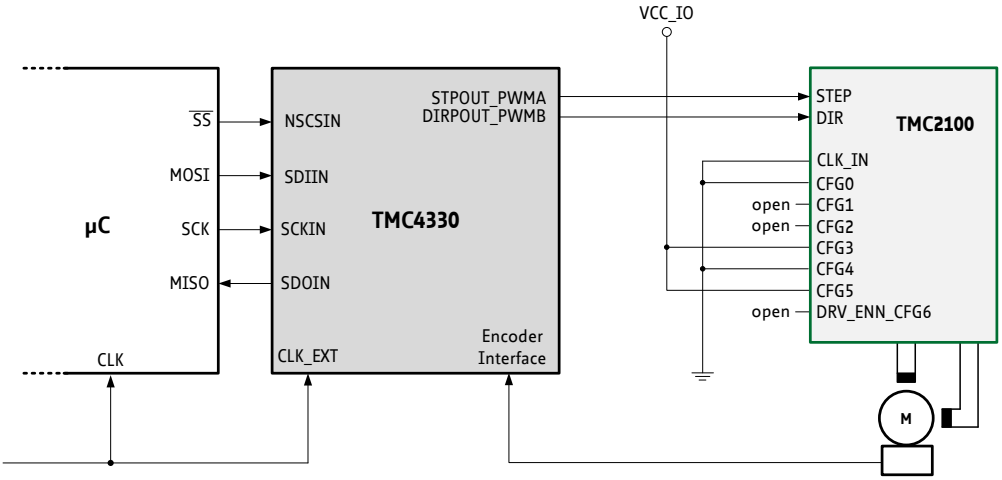


Figure 9: TMC4330A with TMC2100 Stepper Driver in stealthChop Mode

2.3. TMC4330A with TMC22xx Stepper Connection

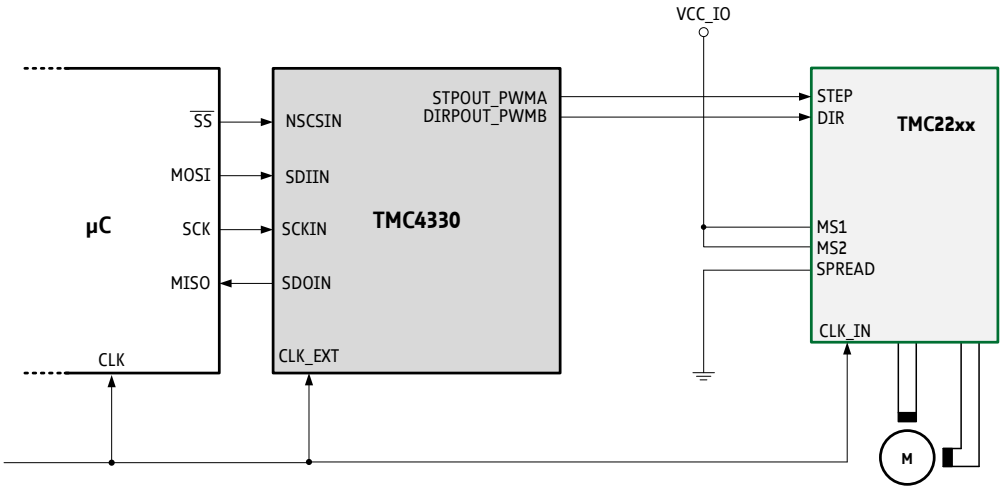


Figure 10: TMC4330A with TMC22xx Stepper Driver (32 microsteps settings)

3. SPI Interfacing

TMC4330A 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. In the following chapter information is provided about the SPI control interface, SPI datagram structure and SPI transaction process.

SPI Input Control Interface Pins		
Pin Name	**Type**	**Remarks**
NSCSIN	Input	Chip Select of SPI-µC interface (low active)
SCKIN	Input	Serial clock of SPI-µC interface
SDIIN	Input	Serial data input of SPI-µC interface
SDOIN	Output	Serial data output of SPI-µC interface

Table 3: SPI Input Control Interface Pins

3.1. SPI Datagram Structure

- Microcontrollers that are equipped with hardware SPI are typically able to communicate using integer multiples of 8 bit.
- The NSCSIN line of the TMC4330A has to stay active (low) for the complete duration of the datagram transmission.
- Each datagram that is sent to TMC4330A is composed of an address byte followed by four data bytes. This allows direct 32-bit data word communication with the register set of TMC4330A. Each register is accessed via 32 data bits; even if it uses less than 32 data bits.

`i` Each register is specified by a one-byte address:
- *For read access the most significant bit of the address byte is 0.*
- *For write access the most significant bit of the address byte is 1.*

NOTE:

→ *Some registers are write only registers. Most registers can be read also; and there are also some read only registers.*

TMC4330A SPI Datagram Structure				
MSB (transmitted first)		**40 bits**		**LSB (transmitted last)**
39		**...**		**0**
→ 8-bit address ← 8-bit SPI status	← → 32-bit data			
39 ... 32	31 ... 0			
→ to TMC4330A: RW + 7-bit address ← from TMC4330A: 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

Figure 11: TMC4330A SPI Datagram Structure

Read entire documentation; especially the "Supplemental Directives" on page 160.

Read/Write Selection Principles and Process

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. Consequently, the bit named W is a WRITE_notREAD control bit.

The active high write bit is the MSB of the address byte.
Consequently, 0x80 must be added to the address for a write access.

The SPI interface always delivers data back to the master, independent of the Write bit W.

Difference between Read and Write Access	
If ...	**Then ...**
The previous access was a read access.	The data transferred back is the data read from the address which was transmitted with the previous datagram.
The previous access was a write access	The data read back mirrors the previously received write data.

Figure 12: Difference between Read and Write Access

Conclusion:
Consequently, 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.

NOTE:

→ *Please note that the following read delivers back data read from the address transmitted in the preceding read cycle. The data is latched immediately after the read request.*

AREAS OF SPECIAL CONCERN

A read access request datagram uses dummy write data.
Read data is transferred back to the master with the subsequent read or write access.

`i` Reading multiple registers can be done in a pipelined fashion. Data that is delivered is latched immediately after the initiated data transfer.

Use of Dummy Write Data

Read and Write Access Examples

For read access to register *XACTUAL* with the address 0x21, the address byte must be set to 0x21 in the access preceding the read access.
For write access to register *VACTUAL*, the address byte must be set to 0x80 + 0x22 = 0xA2. For read access, the data bit can have any value, e.g., 0.

Read and Write Access Examples		
Action	**Data sent to TMC**	**Data received from TMC**
read *XACTUAL*	→ 0x2100000000	← 0xSS[1)] & unused data
read *XACTUAL*	→ 0x2100000000	← 0xSS & *XACTUAL*
write *VACTUAL*:= 0x00ABCDEF	→ 0xA200ABCDEF	← 0xSS & *XACTUAL*
write *VACTUAL*:= 0x00123456	→ 0xA200123456	← 0xSS00ABCDEF

Table 4: Read and Write Access Examples

1) SS is a placeholder for the status bits SPI_STATUS.

Read entire documentation; especially the "Supplemental Directives" on page 160.

Data Alignment

All data is right-aligned. Some registers represent unsigned (positive) values; others represent integer values (signed) as two's complement numbers.
Some registers consist of switches that are represented as bits or bit vectors.

SPI Transaction Process

The SPI transaction process is as follows:

- The slave is enabled for 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.

i A minimum of 40 SCKIN clock cycles is required for a bus transaction with TMC4330A.

AREAS OF SPECIAL CONCERN

System Behavior Specifics

Take the following aspects into consideration:

- **Whenever data is read from or written to the TMC4330A,** the first eight bits that are delivered back contain the SPI status *SPI_STATUS* that consists of eight user-selected event bits. The selection of these bits are explained in chapter 5.2. (Page 22).
- **If less than 40 clock cycles are transmitted,** the transfer is not valid; even for 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 cycles are transmitted,** 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.

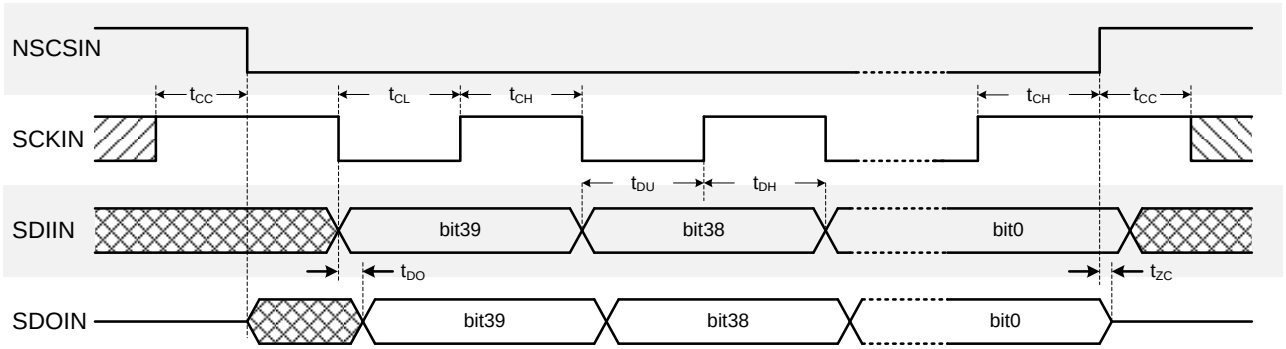


Figure 13: SPI Timing Datagram

3.1.1. SPI Timing Description

The SPI interface is synchronized to the internal system clock, which limits SPI bus clock SCKIN to a quarter of the system clock frequency. The signal processing of SPI inputs is supported with internal Schmitt Trigger, but not with RC elements.

NOTE:

→ *In order to avoid glitches at the inputs of the SPI interface between µC and TMC4330A, external RC elements have to be provided.*

Figure 14 shows the timing parameters of an SPI bus transaction, and the table below specifies the parameter values.

SPI Interface Timing						
SPI Interface Timing	**AC Characteristics:**		**External clock period: t_{CLK}**			
Parameter	**Symbol**	**Conditions**	**Min**	**Type**	**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}	>2·t_{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} / 4 (4)	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

Table 5: SPI Interface Timing

i $t_{CLK} = 1 / f_{CLK}$

4. 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 signals are passed through a digital filter. Particular input pins are separated into four filtering groups. Each group can be programmed individually according to its filter characteristics. In this chapter informed on the digital filtering feature of TMC4330A is provided; and how to separately set up the digital filter for input pins.

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.
STPIN DIRIN	Inputs	Step/Dir interface inputs.

Table 6: Input Filtering Groups (Assigned Pins)

Register Names			
Register Names	**Register Address**		**Remarks**
INPUT_FILT_CONF	0x03	RW	Filter configuration for all four input groups.

Table 7: Input Filtering (Assigned Register)

Input Filter Assignment

Every filtering group can be configured separately with regard to input sample rate and digital filter length.

The following groups exist:

- Encoder interface input pins.
- Reference input pins.
- Start input pin.
- Step/Dir input pins.

NOTE:

→ *For the correct set-up of the INPUT_FILT_CONF register 0x03, please check section 14.4. , page 128.*

Input Sample Rate (SR)

Input sample rate = f_{CLK} $1/2^{SR}$ where:

^{SR} (extended with a particular name extension) is in [0… 7].

i This means that the next input value is considered after 2^{SR} clock cycles.

Sample Rate Configuration

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	$f_{CLK}/128$

Table 8: Sample Rate Configuration

Digital Filter Length (*FILT_L*)

i The filter length *FILT_L* can be set within the range [0… 7].

i The filter length *FILT_L* specifies the number of sampled bits that must have the same voltage level to set a new input bit voltage level.

Digital Filter Length Configuration Table

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.

Table 9: Configuration of Digital Filter Length

4.1. Input Filtering Examples

The following three examples depict input pin filtering of three different input filtering groups.

- i After passing Schmitt trigger, voltage levels are compared to internal signals, which are processed by the motion controller.
- i The sample points are depicted as green dashed lines.

Example 1: Reference Input Pins

In this example every second clock cycle is sampled. Two sampled input bits must be equal to receive a valid input voltage.

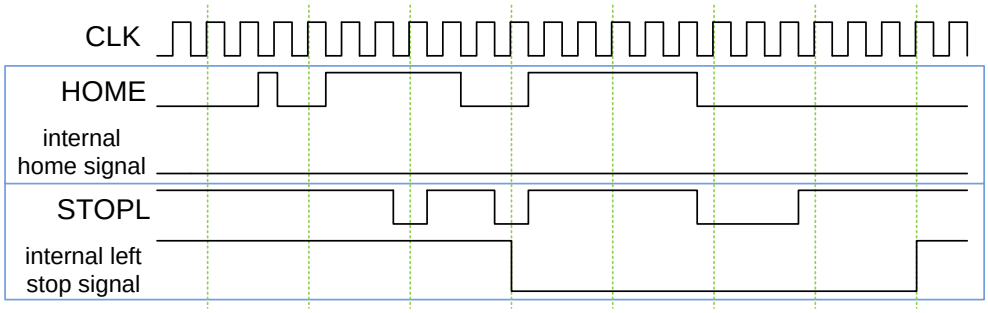


Figure 14: Reference Input Pins: SR_REF = 1, FILT_L_REF = 1

Example 2: START Input Pin

This example shows the START input pattern at every fourth clock cycle:

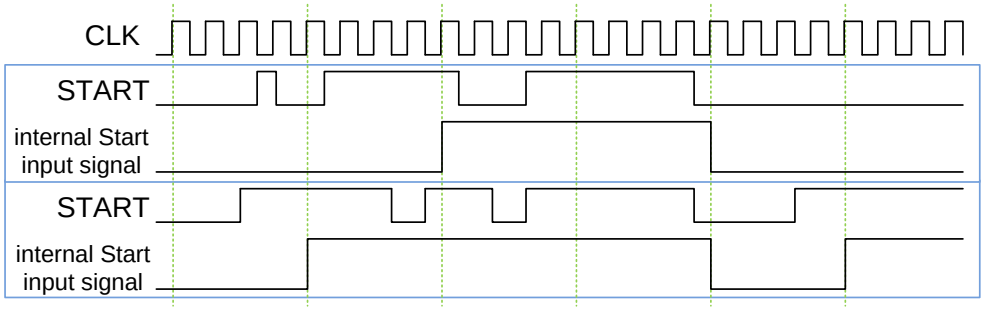


Figure 15: START Input Pin: SR_S = 2, FILT_L_S = 0

Example 3: Encoder Interface Input Pins

This example shows every clock cycle bit. Eight sampled input bits must be equal to receive a valid input voltage.

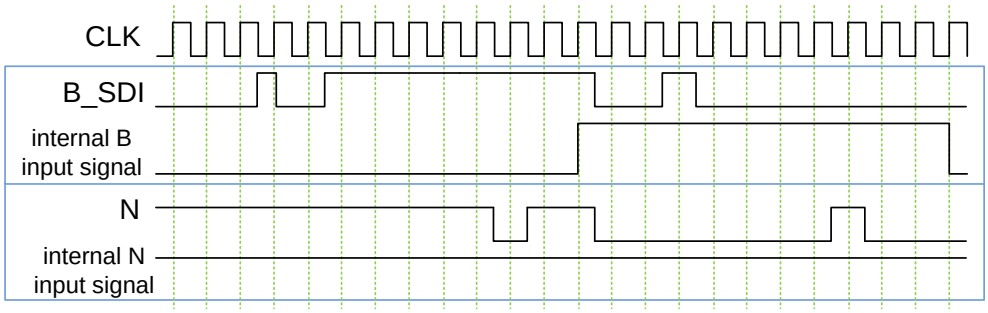


Figure 16: Encoder Interface Input Pins: SR_ENC_IN = 0, FILT_L_ENC_IN = 7

5. Status Flags and Events

TMC4330A provides several status flags and status events to obtain short information on the internal status or motor driver status. These flags and events can be read out from dedicated registers. In the following chapter, you are informed about the generation of interrupts based on status events. Status events can also be assigned to the first eight SPI status bits, which are sent within each SPI datagram.

Pin Names: Status Events		
Pin Names	**Type**	**Remarks**
INTR	Output	Interrupt output to indicate status events.

Table 10: Pins Names: Status Events

Register Names: Status Flags and Events			
Register Name	**Register Address**		**Remarks**
GENERAL_CONF	0X00	RW	Bits: 15, 29, 30.
STATUS_FLAGS	0X0F	R	32 status flags of TMC4330A and the connected TMC motor driver chip.
EVENTS	0X0E	R+C W	32 events triggered by altered TMC4330A 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.

Table 11:Register Names: Status Flags and Events

Read entire documentation; especially the "Supplemental Directives" on page 160.

5.1. Status Event Description

Status events are based on status bits. If the status bits change, related events are triggered from inactive to active level. Resetting events back to inactive must be carried out manually.

Association of Status Bits

Status bits and status events are associated in different ways:

- Status flags reflect the as-is-condition, whereas status events indicate that the dedicated information has changed since the last read request of the *EVENTS* register. Several status events are associated with one status bit.
- Some status events show the status transition of one or more status bits out of a status bit group.
- 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 (b'00/b'01/b'10). Such an event is triggered if the associated combination switches from inactive to active.

NOTE:

→ *Some events have no equivalence in the* STATUS_FLAGS *register 0x0F.*

Automatic Clearance of EVENTS

The *EVENTS* register 0x0E is automatically cleared after reading the register; subsequent to an SPI datagram request. Events are important for interrupt generation and SPI status monitoring.

NOTE:

→ *It is recommended to clear EVENTS register 0x0E by read request before regular operation.*

AREAS OF SPECIAL CONCERN

Recognition of a status event can fail; in case it is triggered right before or during *EVENTS* register 0x0E becomes cleared.

How to Avoid Lack of Information

In order to prevent events from being cleared, assign *EVENT_CLEAR_CONF* register 0x0C according to the particular event in the EVENTS register:

Action:

➢ Set related *EVENT_CLEAR_CONF* register bit position to 1.

Result:

The related event is not cleared when *EVENTS* register is read out.

In order to clear these events, do the following, if necessary:

Action:

➢ Set related *EVENTS* register 0x0E bit position to 1.

Result:

The related event is cleared by writing to the *EVENTS* register.

5.2. SPI Status Bit Transfer

Up to eight events can be selected for permanent SPI status report. Consequently, these events are always transferred at the most significant transfer bits within each TMC4330A SPI response.

Assign an Event to a Status Bit

In order to select an event for the SPI status bits, assign the *SPI_STATUS_SELECTION* register 0x0B according to the particular event in the EVENTS register:

Action:
- Set the related *SPI_STATUS_SELECTION* register bit position to 1.

Result:
The related event is transferred with every SPI datagram response as *SPI_STATUS*.

NOTE:

→ *The bit positions are sorted according to the event bit positions in the EVENTS register 0x0E. In case more than eight events are selected, the first eight bits (starting from index 0 = LSB) are forwarded as SPI_STATUS.*

5.3. Generation of Interrupts

Similar to *EVENT_CLEAR_CONF* register and *SPI_STATUS_SELECTION* register, events can be selected for forwarding via INTR output. The selected events are ORed to one signal which means that INTR output switches active as soon as one of the selected events triggers.

Generate Interrupts

In order to select an event for the INTR output pin, assign the *INTR_CONF* register 0x0D according to the particular event in the EVENTS register:

Action:
- Set the related *INTR_CONF* register bit position to 1.

Result:
The related event is forwarded at the INTR output. If more than one event is requested, INTR becomes active as soon as one of the selected events is active.

INTR Output Polarity

Per default, the INTR output is low active.

In order to change the INTR polarity to high active, do the following:

Action:
- Set *intr_pol* =1 (*GENERAL_CONF* register 0x00).

Result:
INTR is high active.

5.4. Connection of Multiple INTR Pins

INTR pin can be configured for a shared interrupt signal line of several TMC4330A interrupt signals to the microcontroller.

Connecting several Interrupt Pins

In order to make use of a Wired-Or or Wired-And behavior, the below described actions must be taken:

Action:

➢ **Step 1:** Set *intr_tr_pu_pd_en* = 1 (*GENERAL_CONF* register 0x00).

OPTION 1: WIRED-OR

Action:

➢ **Step 2:** Set *intr_as_wired_and* = 0 (*GENERAL_CONF* register 0x00).

Result:
The INTR pin works efficiently as Wired-Or (default configuration).

i In case INTR pin is inactive, the pin drive has a weak inactive polarity output. If one of the connected pins is activated, the whole line is set to active polarity.

OPTION 2: WIRED-AND

Action:

➢ **Step 2:** Set *intr_as_wired_and* = 1 of the *GENERAL_CONF* register 0x00.

Result:
In case no interrupt is active, the INTR pin has a strong inactive polarity output. During the active state, the pin drive has a weak active polarity output. Consequently, the whole signal line is activated in case all pins are forwarding the active polarity.

6. Ramp Configurations for different Motion Profiles

Step generation is one of the main tasks of a stepper motor motion controller. The internal ramp generator of TMC4330A provides several step generation configurations with different motion profiles. They can be configured in combination with the velocity or positioning mode.

Pin Names: Ramp Generator		
Pin Names	**Type**	**Remarks**
STPOUT_PWMA	Output	Step output signal.
DIROUT_PWMB	Output	Direction output signal.

Table 12: Pin Names: Ramp Generator

Register Names: Ramp Generator			
Register Name	**Register Address**		**Remarks**
GENERAL_CONF	0x00	RW	Ramp generator affecting bits 5:0.
STP_LENGTH_ADD	0x10	RW	Additional step length in clock cycles; 16 bits.
DIR_SETUP_TIME			Additional time in clock cycles when no steps will occur after a direction change; 16 bits.
RAMPMODE	0x20	RW	Requested motion profile and operation 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 aceleration/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; 24bits=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.

Table 13: Register Names: Ramp Generator

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.1. Step/Dir Output Configuration

This section focuses on the description of the Step/Dir output configuration.

6.1.1. Step/Dir Output Configuration Steps

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

If step signals must be longer than one clock cycle, do as follows:

Action:
➢ Set proper *STP_LENGTH_ADD* register 0x10 (bit 15:0).

Result:
The resulting step length is equal to *STP_LENGTH_ADD*+1 clock cycles. This is how the step length is assigned within a range of up to 1-up-to-2^{16} clock cycles.

Action:
➢ Set proper *DIR_SETUP_TIME* register 0x10 (bit 31:16).

Result:
The delay period between DIROUT and STPOUT voltage level transitions last *DIR_SETUP_TIME* clock cycles. No steps are sent via STPOUT for *DIR_SETUP_TIME* clock cycles after a level change at DIROUT.

PRINCIPLE:

DIROUT does not change the level:

- During active step pulse signal
- For (*STP_LENGTH_ADD*+1) clock cycles after the step signal returns to inactive level

6.1.2. STPOUT: Changing Polarity

STPOUT characteristics can be set differently, as follows:

Per default, the step output is high active because a rising edge at STPOUT indicates a step.

In order to change the polarity, do as follows:

Action:
➢ Set *step_inactive_pol* =1 (bit3 of *GENERAL_CONF* register 0x00).

Result:
Each falling edge indicates a step.

How to prompt Level Change with every Step

In order to prompt a step at every level change, do as follows:

Action:
➢ Set *toggle_step* =1 (bit4 of *GENERAL_CONF* register 0x00).

Result:
Every level change indicates a step.

DIROUT: Changing the Polarity

Per default, voltage level 1 at DIROUT indicates a negative step direction. DIROUT characteristics can be set differently, as shown below.

In order to change polarity, do as follows:

Action:
➢ Set *pol_dir_out* =0 (bit5 of *GENERAL_CONF* register 0x00).

Result:
A high voltage level at DIROUT indicates a positive step direction.

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.2. Configuration Details for Operation Modes and Motion Profiles

This section provides information on the two available operation modes (velocity mode and positioning mode), and on the four possible motion profiles (no ramp, trapezoidal ramp including sixPoint™ ramp, and S-shaped ramp). Different combinations are possible. Each one of them has specific advantages. The choice of configuration depends on the user's design specification to best suit his design needs.

Description of Internal Ramp Generator

With proper configuration, the internal ramp generator of the TMC4330A is able to generate various ramps with the related step outputs for STPOUT.

In order to configure the internal ramp generator successfully – i.e. to make it fit as best as possible with your specific use case – information about the scope of each possible combination is provided in the table below and on the following pages.

Ramp Generator Configuration Options			
Operation Mode	**Motion Profile**	**RAMPMODE(2:0)**	**Description**
Velocity Mode	No ramp	b'000	Follows *VMAX* request only.
	Trapezoidal ramp	b'001	Follows *VMAX* request and considers acceleration and deceleration values.
	sixPoint ramp	b'001	Follows *VMAX* request and considers acceleration / deceleration values and start and stop velocity values.
	S-shaped ramp	b'010	Follows *VMAX* request and considers maximum acceleration / deceleration values and adapts these values with 4 different bow values.
Positioning Mode	No Ramp	b'100	Follows *XTARGET* and *VMAX* requests only.
	Trapezoidal ramp	b'101	Follows *XTARGET* request and a maximum velocity *VMAX* request and considers acceleration and deceleration values.
	sixPoint ramp	b'101	Follows *XTARGET* request and a maximum velocity *VMAX* request and considers acceleration / deceleration values and start and stop velocity values.
	S-shaped ramp	b'110	Follows *XTARGET* request and a maximum velocity *VMAX* request and considers maximum acceleration / deceleration values and adapts these values with 4 different bow values.

Table 14: Overview of General and Basic Ramp Configuration Options

6.2.1. Starting Point: Choose Operation Mode

Two operation modes are available: velocity mode and positioning mode.

BEFORE YOU BEGIN

Before setting any parameters:

First select:

- **Operation mode and**
- **Motion profile**

It is not advisable to change operation mode nor motion profile during motion.

Operation Mode: Velocity Mode

The *RAMPMODE* register provides a choice of two operation modes. Either velocity mode or positioning mode can be chosen.

In order to use the velocity mode, do as follows:

Action:

➢ Set *RAMPMODE*(2) =0 (*RAMPMODE* register 0x20).

Result:

Velocity mode is selected. The target velocity *VMAX* is reached with the selected motion profile.

Operation Mode: Positioning Mode

In order to make use of the positioning mode, do as follows:

Action:

➢ Set *RAMPMODE*(2)=1 (*RAMPMODE* register 0x20).

Result:

Positioning mode is selected. *VMAX* is the maximum velocity value of this motion profile that is based on the condition that the ramp stops at target position *XTARGET*.

NOTE:

→ *The sign of* VMAX *is not relevant during positioning. The direction of the steps depends on* XACTUAL*,* XTARGET*, and the current ramp motion profile status.*

NOTE:

→ *Do NOT exceed* VMAX ≤ f_{CLK} ¼ *pulses for positioning mode.*

6.2.2. Stop during Motion

In order to stop the motion during positioning, do as follows:

Action:

➢ Set *VMAX* = 0 (register 0x24).

Result:

The velocity ramp directs to VACTUAL = 0, using the actual ramp parameters.

i Motion is proceeded with *VMAX* ≠ 0.

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.2.3. Motion Profile Configuration

Three basic motion profiles are provided. Each one of them has a different velocity value development during the drive. See table below.

For configuration of the motion profiles, do as follows:

Action:
➢ Use the bits 1 and 0 of the *RAMPMODE* register 0x20.

Result:
As specified in the table below.

You can choose different configuration options from the list below:

- No Ramp motion profile
- Trapezoidal Ramp motion profile (including sixPoint Ramp)
- S-shaped Ramp motion profiles

TMC4330A Motion Profile		
***RAMPMODE* (1:0)**	**Motion Profile**	**Function**
b'00	No Ramp	Follow *VMAX* only (rectangular velocity shape).
b'01	Trapezoidal Ramp	Consideration of acceleration and deceleration values without adaptation of these acceleration values.
	sixPoint Ramp	Consideration of acceleration and deceleration values without adaptation of these acceleration values. **Usage of start and stop velocity values. (see section 6.5. , Page 41)**
b'10	S-shaped Ramp	Use all ramp values (including bow values).

Table 15: Description of TMC4330A Motion Profiles

6.2.4. No Ramp Motion Profile

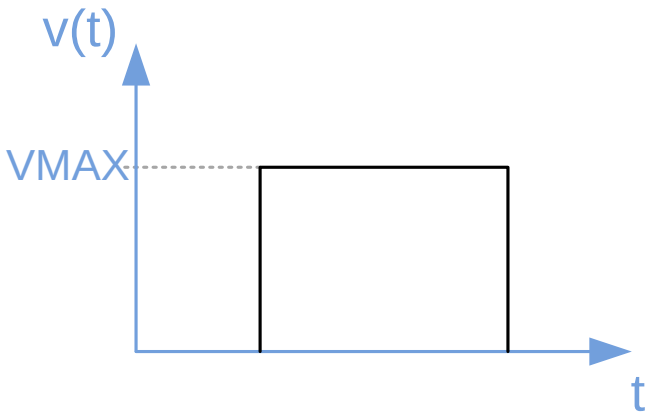


Figure 17: No Ramp Motion Profile

In order to make use of the no ramp motion profile, which is rectangular, do as follows:

Action:
- Set *RAMPMODE*(1:0) =b'00 (register 0x20).
- Set proper *VMAX* register 0x24.

Result:
The internal velocity *VACTUAL* is immediately set to *VMAX*.

Positioning Mode combined with No Ramp Motion Profile

Combining positioning mode with the no ramp motion profile determines that the ramp holds *VMAX* until *XTARGET* is reached. The motion direction depends on *XTARGET*.

In order to make use of the no ramp motion profile in combination with the positioning mode, do as follows:

Action:
- Set *RAMPMODE*(2:0) =b'100.
- Set proper *VMAX* register 0x24.
- Set proper *XTARGET* register 0x37.

Result:
VACTUAL is set instantly to 0 in case the target position is reached.

NOTE:

→ *Do NOT exceed* VMAX ≤ f_{CLK} / 4 *pulses for positioning mode.*

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.2.5. Trapezoidal 4-Point Ramp without Break Point

In order to make use of a trapezoidal 4-point ramp motion profile without break velocity, do as follows:

Action:

- Set *RAMPMODE*(1:0) =b'01 (register 0x20).
- Set *VBREAK* =0 (register 0x27).
- Set proper *AMAX* register 0x28 and *DMAX* register 0x29.
- Set proper *VMAX* register 0x24.

Result:

The internal velocity *VACTUAL* is changed successively to *VMAX* with a linear ramp. Only *AMAX* and *DMAX* define the acceleration/deceleration slopes.

NOTE:

→ *AMAX determines the rising slope from absolute low to absolute high velocities, whereas DMAX determines the falling slope from absolute high to absolute low velocities.*

→ *Acceleration slope and deceleration slopes have only one acceleration and deceleration value each.*

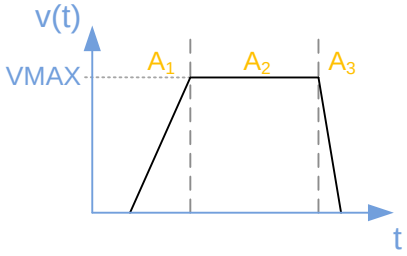


Figure 18: Trapezoidal Ramp without Break Point

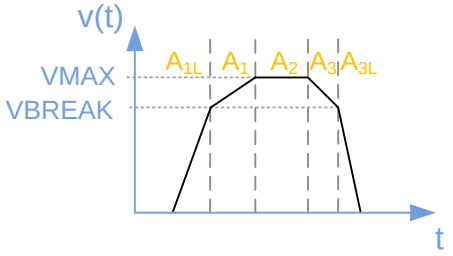


Figure 19: Trapezoidal Ramp with Break Point

6.2.6. Trapezoidal Ramp with Break Point

In order to make use of a trapezoidal ramp motion profile with break velocity, do as follows:

Action:

- Set *RAMPMODE*(1:0)=b'01 (register 0x20).
- Set proper *VBREAK* register 0x27.
- Set proper *AMAX* register 0x28 and *DMAX* register 0x29.
- Set proper *ASTART* register 0x2A and *DFINAL* register 0x2B.
- Set proper *VMAX* register 0x24.

Result:

The internal velocity *VACTUAL* is changed successively to *VMAX* with a linear ramp. In addition to *AMAX* and *DMAX*, *ASTART* and *DFINAL* define the acceleration or deceleration slopes (see Figure above).

NOTES:

→ *AMAX and ASTART determines the rising slope from absolute low to absolute high velocities.*

→ *DMAX and DFINAL determines the falling slope from absolute high to absolute low velocities.*

→ *The acceleration/deceleration factor alters at VBREAK. ASTART and DFINAL are valid below VBREAK, whereas AMAX and DMAX are valid beyond VBREAK.*

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.2.7. Position Mode combined with Trapezoidal Ramps

Motion direction depends on *XTARGET*.

In order to use a 4-point or sixPoint ramps during positioning mode, do as follows:

Action:

- Set *RAMPMODE*(2:0) =b'101 (register 0x20).
- Set Trapezoidal ramp type accordingly, as explained above.
- Set proper *XTARGET* register 0x37.

Result:
The ramp finishes exactly at the target position *XTARGET* by keeping |*VACTUAL*| = *VMAX* as long as possible.

AACTUAL Assignments for Trapezoidal Ramps

***AACTUAL* assignments apply both for 4-point and sixPoint ramps.**

The acceleration/deceleration factor *AACTUAL* register depends on the current ramp phase and the velocity that needs to be reached. The related sign assignment for different ramp phases is given in the following table:

***AACTUAL* ASSIGNMENTS for Trapezoidal Ramps**					
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

Table 16: Trapezoidal Ramps: AACTUAL Assignments during Motion

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.2.8. Configuration of S-Shaped Ramps

In order to make use of S-shaped ramps, do as follows:

Action:

- Set *RAMPMODE*(1:0)=b'10 (register 0x20).
- Set proper *BOW1* … *BOW4* registers 0x2C…0x30.
- Set proper *AMAX* register 0x28 and *DMAX* register 0x29.
- Set *ASTART* = 0 (register 0x2A).
- Set *DFINAL* = 0 (register 0x2B).
- Set proper *VMAX* register 0x24.

Result:

The internal velocity *VACTUAL* is changed successively to *VMAX* with S-shaped ramps. The acceleration/deceleration values are altered on the basis of the bow values.

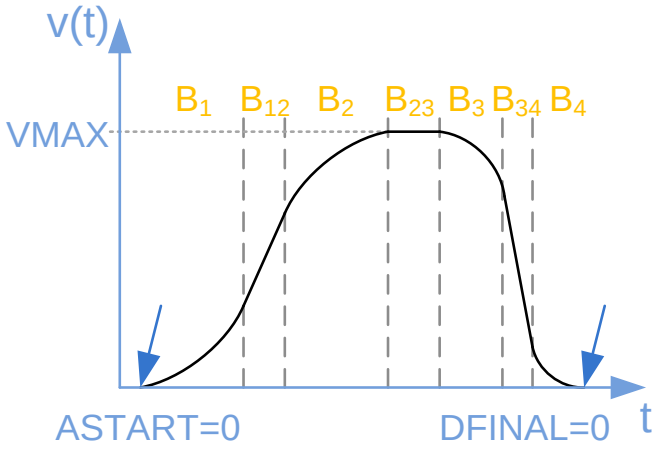


Figure 20: S-shaped Ramp without initial and final Acceleration/Deceleration Values

Definition of Rising Slope for S-shaped Ramps

Rising slope (absolute lower velocities to absolute higher velocities):

- *BOW1* determines the value which increases the absolute acceleration value.
- *BOW2* determines the value which decreases the absolute acceleration value.
- *AMAX* determines the maximum acceleration value.

Definition of Falling Slope for S-shaped Ramps

Falling slope (absolute higher velocities to absolute lower velocities):

- *BOW3* determines the value which increases the absolute deceleration value.
- *BOW4* determines the value which decreases the absolute deceleration value.
- *DMAX* determines the maximum absolute deceleration value.

•→ Description is continued on next page.

Changing ramp parameters[1] and/or operation mode during motion is not advised. However, if this is necessary, the following applies:

NOTICE

Avoid unintended system behavior during positioning mode!
Ramp parameter value changes during ramp progress can lead to:

- A temporary overshooting of *XTARGET* or mechanical stop positions.
- A temporary overshooting of *VACTUAL* beyond *VMAX* because the bows B1, B2, B3, and B4 are maintained during the ramp progress.

This will ensure smooth operation during positioning mode.

[1] Exceptions are XTARGET and VMAX. These Parameters can be changed during motion.

6.2.9. Changing Ramp Parameters during S-shaped Motion or Switching to Positiong Mode

However, if it is necessary to change ramp parameters for S-shaped ramps during motion or to swtich from velocity to positioning mode, do as follows:

Action:

➢ Set or set again proper *BOW3* registers 0x2F, regardless of wether the value changes or not.

`i` Set this parameter after all other parameters have been set.

Result:
Internal ramp calculations are reset through which the velocity ramp operates at safe mode. During this mode, the target velocity is set to 0. In case the internal ramp calculations are up-to-date, the ramp, which is configured by the actual ramp parameters, is continued.

6.2.10. Configuration of S-shaped Ramp with *ASTART* and *DFINAL*

In order to configure S-shaped ramps with starting and finishing values for acceleration or deceleration, do as follows:

Action:

➢ Set *RAMPMODE*(1:0)=b'10 (register 0x20).
➢ Set S-Shaped ramp as explained above (*BOW1 … BOW4, AMAX, DMAX)*.
➢ Set proper *ASTART* register 0x2A.
➢ Set proper *DFINAL* register 0x2B.
➢ Set proper *VMAX* register 0x24.

Result:
The internal velocity *VACTUAL* is changed successively to *VMAX* with S-Shaped ramps.

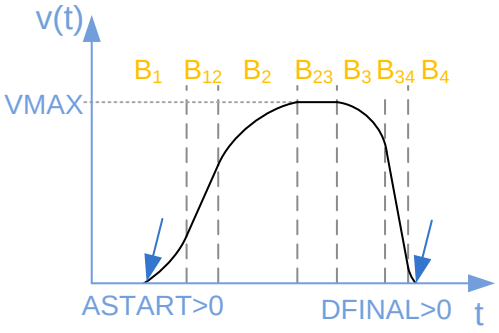


Figure 21: S-shaped Ramp with initial and final Acceleration/Deceleration Values

•→ Description is continued on next page.

Definitions for S-shaped Ramps

- The acceleration/deceleration values are altered, based on the bow values.
- The start phase and the end phase of an S-shaped ramp is accelerated/decelerated by *ASTART* and *DFINAL*.
- The ramp starts with *ASTART* and stops with *DFINAL*.
- *DFINAL* becomes valid when *AACTUAL* reaches the chosen *DFINAL* value.

`i` The parameter *DFINAL* is not considered during positioning mode.

AACTUAL Assignments for S-shaped Ramps

AACTUAL assignments and current bow value selection for S-shaped ramps. The acceleration/deceleration factor depends on the current ramp phase and alters every 64 clock cycles during the bow phases B1, B2, B3, and B4.

Details are provided in the table below:

S-shaped Ramps: Assignments for *AACTUAL* and Internal Bow Value							
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*

Table 17: Parameter Assignments for S-shaped Ramps

6.2.11. S-shaped Mode and Positioning: Fast Motion

***RAMPMODE*(2:0) =b'110**

- The ramp finishes exactly on target position; keeping |*VACTUAL*| = *VMAX* as long as possible until the ramp falls to reach *XTARGET* exactly.
- It is possible that the phases B12, B23, and B34 are left out due to given values. Therefore, the highest speed performance is possible due to a maximum speed positioning ramp.
- The fastest possible slopes are always performed if the phases B12 and/or B34 are not reached during a rising and/or falling S-shaped slope.
- 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.

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.3. Start Velocity *VSTART* and Stop Velocity *VSTOP*

S-shaped and trapezoidal velocity ramps can be configured with unsigned start and stop velocity values: V*START*, or *VSTOP*.
Per default, *VSTART* and *VSTOP* are set to 0. The sign is selected automatically, depending on the current ramp status and the target velocity, or target position. This section explains how to set up the respective values correctly.

Starting Ramps with initial Velocity

S-shaped and trapezoidal velocity ramps can be started with an initial velocity value, if you set the *VSTART* value higher than zero (see Figure below).

In order to use trapezoidal ramps with an initial start velocity, do as follows:

Action:
- Set *RAMPMODE*(1:0)=b'01 (register 0x20).
- Set Trapezoidal ramp type accordingly, as explained before.
- Set proper *VSTART* > 0 (register 0x25).
- Set *VSTOP* = 0 (register 0x26).

Result:
The trapezoidal ramp starts with initial velocity.

NOTE:

→ *The initial acceleration value is AMAX if VBREAK < VSTART, otherwise the starting acceleration value is ASTART.*

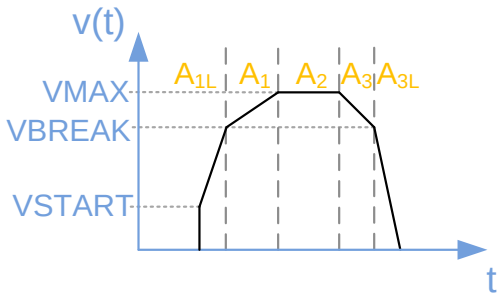


Figure 22: Trapezoidal Ramp with initial Velocity

If trapezoidal ramp with initial velocity *VSTART* is selected:

NOTICE

Avoid unintended system behavior during positioning mode!

- Use *VSTART* without setting *VSTOP* > *VSTART* only in positioning mode if there is enough distance between the current position *XACTUAL* and the target position *XTARGET*.

This will ensure smooth operation during positioning mode.

•→Turn page for information on how to configure S-shaped ramps with initial start velocity.

Read entire documentation; especially the "Supplemental Directives" on page 160.

S-shaped Ramps with initial Start Velocity

In order to use S-shaped ramps with initial start velocity, do as follows:

Action:

- Set *RAMPMODE*(1:0)=b'10 (register 0x20).
- Set S-shaped ramp type accordingly, as explained before.
- Set proper *VSTART* > 0 (register 0x25).
- Set *VSTOP* = 0 (register 0x26).

Result:
The S-shaped ramp starts with initial velocity.

PRINCIPLE:

→ *The initial acceleration value is equal to AMAX. The parameter ASTART is not considered. Consequently, ramp phase B1 is not performed.*

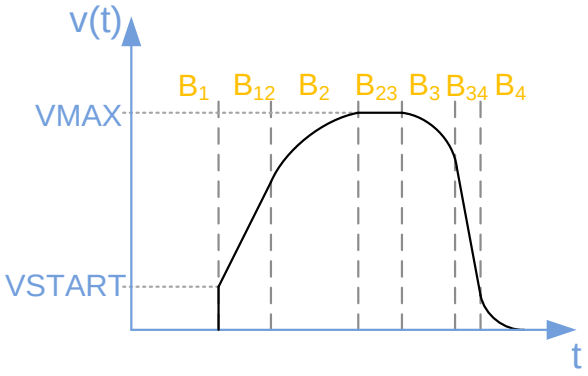


Figure 23: S-shaped Ramp with initial Start Velocity

If S-shaped ramp with initial velocity *VSTART* is selected:

NOTICE

Avoid unintended system behavior during positioning mode!

- Keep in mind that the S-shaped character of the curve is maintained. Because *AMAX* is the start acceleration value, the ramp will always execute phase B2 which could result in positioning overshoots.
- Use *VSTART* only in positioning mode if there is enough distance between the current position *XACTUAL* and the target position *XTARGET*.

This will ensure smooth operation during positioning mode.

•→Turn page for information on how to configure finishing ramps with stop velocity.

Read entire documentation; especially the "Supplemental Directives" on page 160.

Finishing Ramps with Stop Velocity

S-shaped and trapezoidal velocity ramps can be finished with a stop velocity value if you set *VSTOP* value higher than zero (see figure below).

In order to configure trapezoidal ramps with stop velocity, do as follows:

Action:

- Set *RAMPMODE*(1:0)=b'01 (register 0x20).
- Set Trapezoidal ramp type accordingly, as explained before.
- Set *VSTART* = 0 (register 0x25).
- Set proper *VSTOP* > 0 (register 0x26).

Result:
The trapezoidal ramp stops with defined velocity.

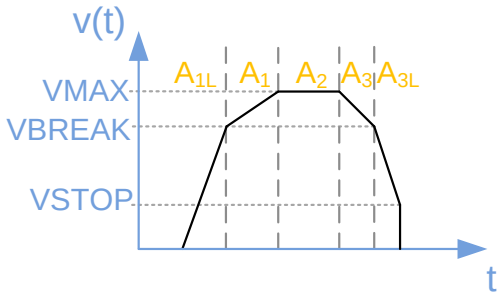


Figure 20: Trapezoidal Ramp with Stop Velocity

If trapezoidal ramps are selected (*VBREAK* > 0):

NOTICE

Avoid unintended system behavior during positioning mode!

- Set *VBREAK* > *VSTOP*.
- Set *VSTART* < *VSTOP*.

This will ensure smooth operation during positioning mode.

•→Turn page for configuration information on S-shaped ramps with stop velocity.

Read entire documentation; especially the "Supplemental Directives" on page 160.

S-shaped Ramps with Stop Velocity

In order to use S-shaped ramps with stop velocity, do as follows:

Action:

- Set *RAMPMODE*(1:0)=b'10 (register 0x20).
- Set S-shaped ramp type accordingly, as explained before.
- Set *VSTART* = 0 (register 0x25).
- Set proper *VSTOP* > 0 (register 0x26).

Result:
The S-shaped ramp finishes with stop velocity.

NOTE:

→ *The final deceleration value is equal to DMAX. The parameter DFINAL is not considered. Consequently, ramp phase B4 is not performed.*

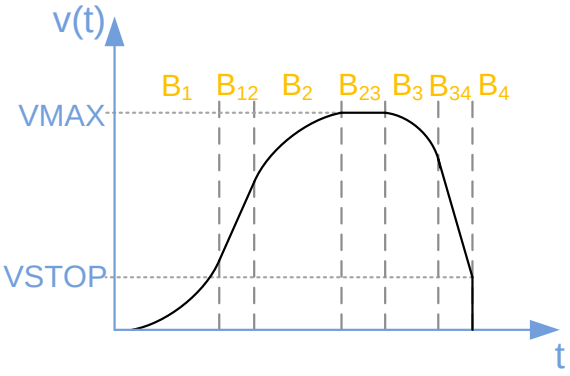


Figure 24: S-shaped Ramp with Stop Velocity

Interaction of *VSTART*, *VSTOP*, *VACTUAL* and *VMAX*:

- *VSTOP* can be used in positioning mode, if the target position is reached. In velocity mode, *VSTOP* is also used if *VACTUAL* ≠ 0 and the target velocity *VMAX* is assigned to 0.
- *VSTART* and *VSTOP* are not only used to start or end 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, using *VSTART* or *VSTOP*.
- The unsigned values *VSTART* and *VSTOP* are valid for both velocity directions.
- Every register value change is assigned immediately.

•→Turn page for information on how to configure S-shaped ramps with start and stop velocity.

6.3.1. S-shaped Ramps with Start and Stop Velocity

S-shaped ramps can be configured with a combination of *VSTART* and *VSTOP*. It is possible to include both processes in one S-Shaped ramp to decrease the time between start and stop of the ramp.

In order to use S-Shaped ramps with a combination of start and stop velocity, do as follows:

Action:

- Set *RAMPMODE*(1:0)=b'10.
- Set S-shaped ramp type accordingly, as explained before, but with BOW2 ≠ BOW4.
- Set proper *VSTART* > 0 (register 0x25).
- Set proper *VSTOP* > 0 (register 0x26).

Result:
The S-shaped ramp starts with initial velocity and stops with defined velocity.

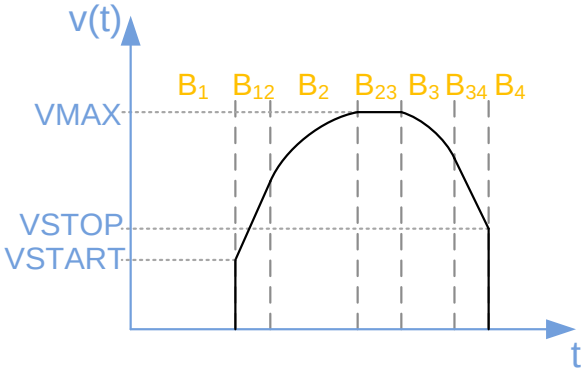


Figure 25: S-shaped Ramp with Start and Stop Velocity

If S-shaped ramp with initial velocity *VSTART* and stop velocity *VSTOP* is selected:

NOTICE

Avoid unintended system behavior during positioning mode!

- Keep in mind that the S-shaped character of the curve is maintained. Because *AMAX* is the start acceleration value, the ramp will always execute phase B2, which could result in positioning overshoots.
- Use *VSTART* in positioning mode, if there is enough distance between the current position *XACTUAL* and the target position *XTARGET*.

This will ensure smooth operation during positioning mode.

•→Turn page for information on how to use VSTART and ASTART for S-shaped ramps.

6.3.2. Combined Use of *VSTART* and *ASTART* for S-shaped Ramps

For some S-shaped ramp applications it can be useful to start with a defined velocity value (*VSTART* > 0);but not with the maximum acceleration value *AMAX*.

In order to start with a defined velocity value, do as follows:

Action:

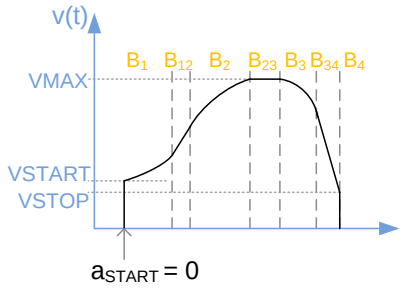
- Set *RAMPMODE*(1:0) =b'10 (register 0x20).
- Set S-shaped ramp type accordingly, as explained before.
- Set proper *VSTART* > 0 (register 0x25).
- Set proper *VSTOP* > 0 (register 0x26).
- Set *use_astart_and_vstart* =1 (bit0 of the *GENERAL_CONF* register 0x00).

Result:

The following special ramp types can be generated in this way, as shown below.

i Section B1 is passed through although *VSTART* is used.

Using VSTART *and starting acceleration of 0 for S-shaped ramps*



Using VSTART *and starting acceleration, which is smaller than* AMAX *for S-shaped ramps*

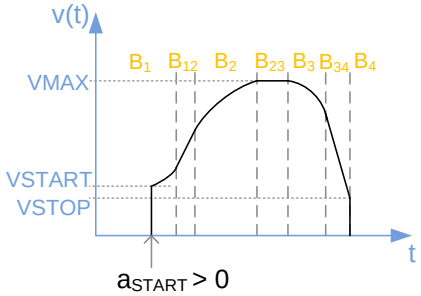


Figure 26: S-shaped Ramps with combined VSTART *and* ASTART *Parameters*

If S-shaped ramp with *VSTART*, *ASTART*, and *VSTOP* is selected:

NOTICE

Avoid unintended system behavior during positioning mode!

- Keep in mind that the S-shaped character of the curve is maintained. Because *ASTART* is the start acceleration value, the ramp will always execute phase B2, which could result in positioning overshoots.
- Use *VSTART* and *ASTART* > 0 without setting *VSTOP* > *VSTART* only in positioning mode, if there is enough distance between the current position *XACTUAL* and the target position *XTARGET*.

This will ensure smooth operation during positioning mode.

6.4. sixPoint Ramps

sixPoint ramps are trapezoidal ramps with initial and stop velocity values that also make use of two acceleration and two deceleration values.

Configuration of sixPoint Ramps

sixPoint ramps are trapezoidal velocity ramps that can be configured with a combination of *VSTART* and *VSTOP.*

In order to use trapezoidal ramps with a combination of start and stop velocity, do as follows:

Action:

- Set *RAMPMODE*(1:0)=b'01 (register 0x20).
- Set a Trapezoidal ramp type appropriately as explained in section 6.2.6, page 30.
- Set proper *VSTART* > 0 (register 0x25).
- Set proper *VSTOP* > 0 (register 0x26).
- Set proper *VBREAK* > 0 (register 0x27).

Result:
The sixPoint ramp starts with an initial velocity and stops with a defined velocity.

Diagram of sixPoint Ramp

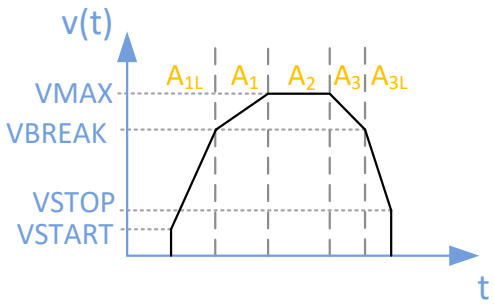


Figure 27: sixPoint Ramp: Trapezoidal Ramp with Start and Stop Velocity

If a sixPoint ramp is used:

NOTICE

Avoid unintended system behavior during positioning mode!

- Set *VBREAK* > *VSTOP*.
- Set *VSTART* < *VSTOP*.

This will ensure smooth operation during positioning mode.

6.5. U-Turn Behavior

The process that is triggered when motion direction changes during motion, is described below, and applies to all ramp types.

U-Turn Behavior

In case the motion direction is changed during motion in velocity mode (by direct assignment of *VMAX*) or in positioning mode (due to *XTARGET* reassignment), the following process is triggered:

1. Motion is directed to *VACTUAL* = 0.
   - `i` If *VSTOP* is used (≠ 0), motion terminates at *VSTOP*.
2. A standstill phase of *TZEROWAIT* clock cycles (register 0x7B) occurs.
   - `i` It is recommended to assign *TZEROWAIT* > 0, if *VSTOP* and/or a trapezoidal ramp type are used, because motor oscillations can occur that must peter out.
3. Motion continues to the actual *XTARGET* (positioning mode), or to the newly assigned *VMAX* (velocity mode).
   - `i` If *VSTART* is used (≠ 0), motion begins with *VSTART* if *TZEROWAIT* > 0.

Example: U-Turn for sixPoint Ramps

After reaching *VSTOP*, *TZEROWAIT* clock cycles are waited until motion continues to peter out motor oscillations.

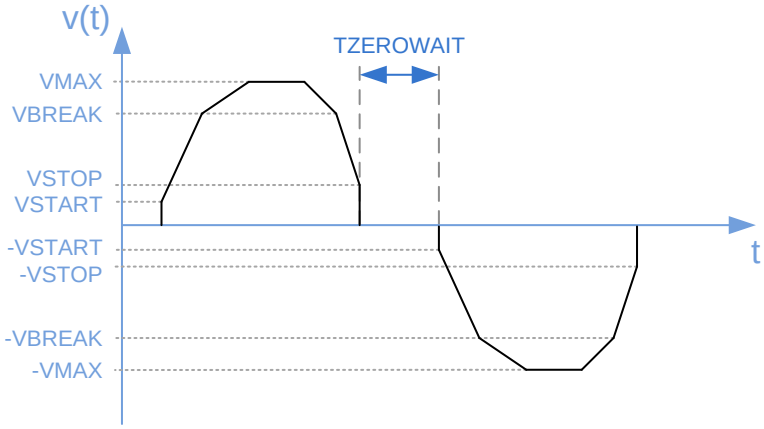


Figure 28: Example for U-Turn Behavior of sixPoint Ramp

•→ *Turn page for information on U-Turn for S-shaped ramps.*

Example: U-Turn for S-shaped Ramps

When *VACTUAL* = 0 is reached, motion immediately continues. In most S-shaped ramp applications that do not use *VSTOP*, a standstill phase is not required.
If *ASTART* > 0 and/or *DFINAL* > 0, these parameters are also used during U-Turn.

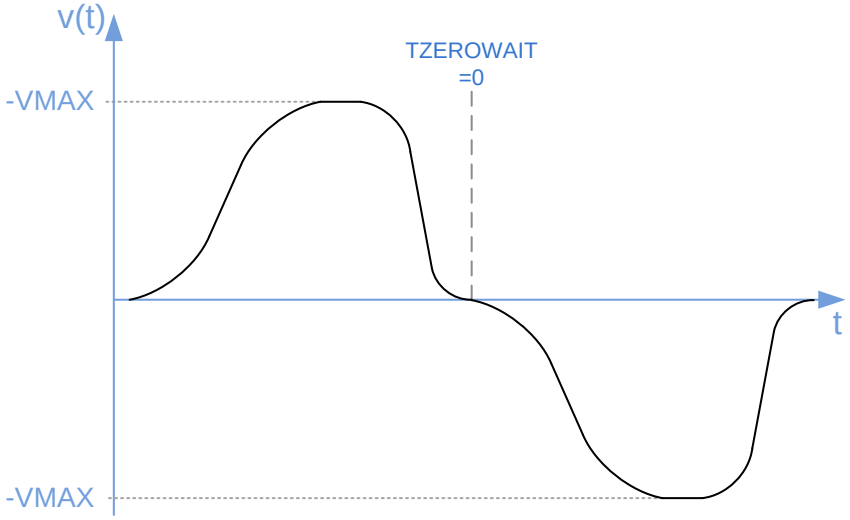


Figure 29: Example for U-Turn Behavior of S-shaped Ramp

6.5.1. Continuous Velocity Motion Profile for S-shaped Ramps

There is one exception to the above explained U-Turn process:
In case ***BOW2* equals *BOW4***, the S-shaped ramp is not stopped at *VACTUAL* = 0. While passing *VACTUAL* = 0, motion acceleration does not equal 0. Thus, the fastest possible U-Turn behavior for this ramp is created.

In the figure below, this velocity ramp behavior is depicted as bold black line, whereas the velocity ramp behavior of the process explained above is depicted gray line:

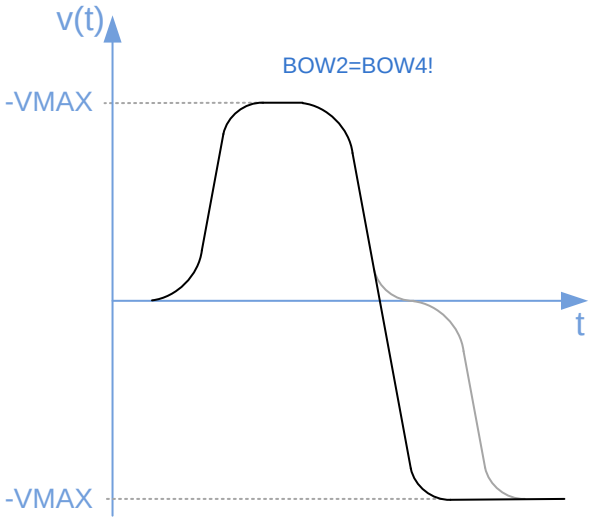


Figure 30: Direct transition via VACTUAL=0 for S-shaped Ramps

Read entire documentation; especially the "Supplemental Directives" on page 160.

6.6. Internal Ramp Generator Units

This section provides information about the arithmetical units of the ramp parameters.

6.6.1. Clock Frequency

All parameter units are real arithmetical units.
Therefore, it is necessary to set the *CLK_FREQ* register 0x31 to proper [Hz] value, which is defined by the external clock frequency f_{CLK}. Any value between f_{CLK} = 4.2 MHz and 32 MHz can be selected.
Default configuration is 16 MHz.

6.6.2. Velocity Value Units

Velocity values are always defined as pulses per second [pps].
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.

The maximum velocity *VMAX* is restricted as follows:

Velocity mode: $|VMAX| \leq ½ \text{ pulse} \cdot f_{CLK}$
Positioning mode: $|VMAX| \leq ¼ \text{ pulse} \cdot f_{CLK}$

NOTE:

→ *In case VACTUAL exceeds this limit INCORRECT step pulses at STPOUT output occur and/or positioning is not executed properly.*

Furthermore, VMAX have to be the highest nominal value of all velocity values:

$|VMAX| > \max(VSTART;VSTOP;VBREAK)$

6.6.3. Acceleration Value Units

The unsigned values *AMAX*, *DMAX*, *ASTART*, *DFINAL,* and *DSTOP* consist of 22 digits and 2 decimal places.
AACTUAL shows a 32-bit nondecimal signed value. Acceleration and deceleration units are defined per default as pulses per second² [pps²].

If higher acceleration/deceleration values are required for short and steep ramps, do as follows:

Action:
➢ Set *direct_acc_val_en* =1 (*GENERAL_CONF* register 0x00).

Result:
The parameters are defined as velocity value change per clock cycle with 24-bit unsigned decimal places (MSB =2^{-14}). The values are calculated as follows:

$AMAX\,[pps^2] = AMAX / 2^{37} \cdot f_{CLK}^2$
$DMAX\,[pps^2] = DMAX / 2^{37} \cdot f_{CLK}^2$
$ASTART\,[pps^2] = ASTART / 2^{37} \cdot f_{CLK}^2$
$DFINAL\,[pps^2] = DFINAL / 2^{37} \cdot f_{CLK}^2$
$DSTOP\,[pps^2] = DSTOP / 2^{37} \cdot f_{CLK}^2$

The maximum acceleration or deceleration values are as follows:

$\max(AMAX;DMAX;ASTART;DFINAL;DSTOP)\ [pps^2] \leq VMAX \cdot f_{CLK} / 1024$

In case *direct_acc_val_en* = 1, the maximum value is also limited to:

$\max(AMAX;DMAX;ASTART;DFINAL;DSTOP) \leq 2^{20}$

•→ *Continued on next page.*

6.6.4. Bow Value Units

Bow values BOW1…BOW4:
Bow values are unsigned 24-bit values without decimal places. They are defined per default as pulses per second³ [pps³].

In case higher bow values are required for short and steep ramps, do as follows:

Action:
➢ Set *direct_bow_val_en* =1 (*GENERAL_CONF* register 0x00)

Result:
The parameters are defined as acceleration value change per clock cycle with 24-bit unsigned decimal places with the MSB defined as 2^{-29}.

The particular bow values *BOW1*, *BOW2*, *BOW3*, *BOW4* are calculated as follows:

$$BOWx\ [pps^3] = BOWx / 2^{53} \cdot f_{CLK}^3$$

The maximum bow are as follows:

$$\max(BOW1…4)\ [pps^2] \leq \max(AMAX;DMAX)\ [pps^2] \cdot f_{CLK} / 1024$$

In case *direct_bow_val_en* = 1, the maximum value is also limited to:

$$\max(BOW1…4) \leq 2^{20}$$

6.6.5. Overview of Minimum and Maximum Values:

Minimum and Maximum Values (Frequency Mode and in general)				
Value Classes	**Velocity**	**Acceleration**	**Bow**	**Clock**
Affected Registers	*VMAX, VSTART, VSTOP, VBREAK*	*AMAX, DMAX, ASTART, DFINAL*	*BOW1, BOW2, BOW3, BOW4*	*CLK_FREQ* (f_{CLK})
Minimum Nominal Value	3.906 mpps	0.25 mpps²	1 mpps³	4.194 MHz
Maximum Nominal Value	8.388 Mpps	4.194 Mpps²	16.777 Mpps³	32 MHz
Maximum Related Value	Velocity mode: ½ pulse · f_{CLK}	*VMAX* · f_{CLK} / 1024	max(*AMAX;DMAX*) · f_{CLK} / 1024	
	Positioning mode: ¼ pulse · f_{CLK}			
	|*VMAX*| > max(*VSTART*;*VSTOP*;*VBREAK*)			

Table 18: Minimum and Maximum Values if Real World Units are selected

Minimum and Maximum Values for Steep Slopes (Direct Mode, example with f_{CLK} =16MHz)		
Value Classes	**Acceleration (*direct_acc_val_en* =1)**	**Bow (*direct_bow_val_en* =1)**
Affected Registers	*AMAX, DMAX, ASTART, DFINAL, DSTOP*	*BOW1, BOW2, BOW3, BOW4*
Calculation	a[pps²] = (Δv/clk_cycle) / $2^{37} \cdot f_{CLK}^2$	bow[pps³] = (Δa/clk_cycle) / $2^{53} \cdot f_{CLK}^3$
Minimum Nominal Value	~1.86 kpps²	~454.75 kpps³
Maximum Nominal Value	~1.95 Gpps²	~476.837 Gpps³
Maximum Related Value	*VMAX* · 15625 Hz	max(*AMAX;DMAX*) · 15625 Hz

Table 19: Minimum and Maximum Values for Steep Slopes for f_{CLK} =16MHz

Read entire documentation; especially the "Supplemental Directives" on page 160.

7. External Step Control and Electronic Gearing

Steps can also be generated by external steps that are manipulated internally by an electronic gearing process. In the following chapter, steps generation by external control and electronic gearing is presented.

Pins for External Step Control		
Pin Names	**Type**	**Remarks**
STPIN	Input	Step input signal.
DIRIN	Input	Direction input signal.

Table 20: Pins used for External Step Control

Registers used for external Step Control			
Register Name	**Register Address**		**Remarks**
GENERAL_CONF	0x00	RW	Bits 9:6, 26.
GEAR_RATIO	0x12	RW	Electronic gearing factor; signed; 32 bits=8+24 (8-bit digits, 24-bit decimal places).

Table 21: Registers used for External Step Control

Enabling External Step Control

In order to synchronize with other motion controllers, TMC4330A offers a step direction input interface at the STPIN and DIRIN input pins.

`i` Three options are available. In case one of these options is selected, the internal step generator is disabled.

OPTION 1: HIGH ACTIVE EXTERNAL STEPS

Action:

➢ Set *sdin_mode* = b'01 (*GENERAL_CONF* register 0x00).

Result:

As soon as the STPIN input signal switches to high state the control unit recognizes an external step.

OPTION 2: LOW ACTIVE EXTERNAL STEPS

Action:

➢ Set *sdin_mode* = b'10 (*GENERAL_CONF* register 0x00).

Result:

As soon as the STPIN input signal switches to low state the control unit recognizes an external step.

OPTION 3: TOGGLING EXTERNAL STEPS

Action:

➢ Set *sdin_mode* = b'11 (*GENERAL_CONF* register 0x00).

Result:

As soon as the STPIN input signal switches to low or high state the control unit recognizes an external step.

•→ Continued on next page.

Selecting the Input Direction Polarity

DIRIN polarity can be assigned. Per default, the negative direction is indicated by DIRIN = 0.

In order to change this polarity:

Action:
➢ Set *pol_dir_in* = 1 (*GENERAL_CONF* register 0x00).

Result:
A negative input direction is assigned by DIRIN = 1.

7.1. Description of Electronic Gearing

If an external step is not congruent with an internal step, the *GEAR_RATIO* register 0x12 must be set accordingly. This signed parameter consists of eight bit digits and 24 bits decimal places.
With every external step the assigned *GEAR_RATIO* value is added to an internal accumulation register. As soon as an overflow occurs, an internal step is generated and the remainder will be kept for the next external step.
Any absolute gearing value between 2^{-24} and 127 is possible.

NOTE:

→ *Gearing ratios beyond 1 generate a burst of steps at the STPOUT pin.*
→ *A negative gearing factor* GEAR_RATIO *< 0 inverts the interpretation of the input direction which is determined by DIRIN and* pol_dir_in*.*

7.2. Indirect External Control

It is possible to use the internal ramp generator in combination with the external S/D interface.

In this case, the external step impulses transferred via STPIN and DIRIN cannot influence the internal *XACTUAL* counter directly. Instead, the *XTARGET* register is altered by 1 or -1 with every *GEAR_RATIO* accumulation register overflow.

NOTE:

→ *Whether XTARGET is increased or decreased is determined similarly to the direct electronic gearing control. The accumulation register overflow direction indicates the target alteration. Respectively, the accumulation direction is determined by the GEAR_RATIO sign, by pol_dir_in, and by DIRIN.*
→ *Consecutive input steps must occur with a distance of minimum 64 clock cycles.*

i This feature allows a synchronized motion of different positioning ramps for different TMC4330A chips with differently configured ramps.

In order to select indirect external control, do as follows:

Action:
➢ Set *sdin_mode* ≠ b'00 according to the required external control option.
➢ Set *sd_indirect_control* = 1 (*GENERAL_CONF* register 0x00).

Result:
As soon as an external step is generated, *XTARGET* is increased or decreased, according to the accumulation direction.

7.3. Switching from External to Internal Control

In some cases, it is useful to switch from external to internal ramp generation during motion.
TMC4330A supports a smooth transfer from direct external control to an internal ramp. The only parameter you need to know and apply is the current velocity when the switching occurs. In more detail, this means that when the external control is switched off, *VSTART* takes over the definition of the actual velocity value. The ramp direction is then selected automatically. The time step of the last internal step is also taken into account in order to provide a smooth transition from external to internal ramp control.

In order to select automatic switching from external to internal control, do as follows:

PRECONDITION (EXTERNAL DIRECT CONTROL IS ACTIVE):

Action:
- Set *sdin_mode* ≠ b'00 (*GENERAL_CONF* register 0x00).
- Set *sd_indirect_control* = 0 (*GENERAL_CONF* register 0x00).
- Set *ASTART* = 0 (register 0x2A).

PROCEED WITH:

Action:
- Set *automatic_direct_sdin_switch_off* = 1 (*GENERAL_CONF* register 0x00) once before switching to internal control.
- Continually adapt *VSTART* register 0x25 according to the actual velocity of the TMC4330A that must be calculated in the µC.
- If switching must be prompted, set *sdin_mode* = b'00.

Result:
The internal ramp velocity is started with the value of *VSTART*, and the direction is set automatically on the basis of the external steps that have occurred before.

Smooth Switching for S-shaped Ramps

In order to also support a smooth S-shaped ramp transition - when the external step control is switched off - the starting acceleration value can also be set separately at *ASTART* register 0x2A.

i In contrast to the automatic direction assignment, the sign of *ASTART* must be set manually.

In order to select automatic switching from external to internal control with a starting acceleration value, do as follows:

PRECONDITION (EXTERNAL DIRECT CONTROL IS ACTIVE):

Action:
- Set *sdin_mode* ≠ b'00 (*GENERAL_CONF* register 0x00).
- Set *sd_indirect_control* = 0 (*GENERAL_CONF* register 0x00).

PROCEED WITH:

Action:
- Set *automatic_direct_sdin_switch_off* = 1 once before switching to internal control.
- Continually adapt *VSTART* register 0x25 according to the actual velocity of the TMC4330A — that must be calculated in the µC.
- Continually adapt *ASTART* according to the actual acceleration (unsigned value) of the TMC4330A — that must be calculated in the µC.
- Continually set *ASTART*(31) = 0 or 1 according to the acceleration direction.
- If switching must be prompted, set *sdin_mode* = b'00.

Result:
The internal ramp velocity is started with the value of *VSTART*, and the direction is set automatically on the basis of the external steps that have occurred before. The internal acceleration value is set to: *+ASTART* if *ASTART*(31) = 0 or *−ASTART* if *ASTART*(31) = 1.

Read entire documentation; especially the "Supplemental Directives" on page 160.

8. Reference Switches

The reference input signals of the TMC4330A function partly as safety features. The TMC4330A provides a range of reference switch settings that can be configured for many different applications. The TMC4330A offers two hardware switches (STOPL, STOPR) and two additional virtual stop switches (*VIRT_STOP_LEFT*, *VIRT_STOP_RIGHT*). A home reference switch HOME_REF is also available.

Pins used for 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.*

Table 22: Pins used for Reference Switches

Dedicated Registers for Reference Switches			
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.

Table 23: Dedicated Registers for Reference Switches

Read entire documentation; especially the "Supplemental Directives" on page 160.

8.1. Hardware Switch Support

The TMC4330A offers two hardware switches that can be configured according to your design.

STOPL and STOPR

The hardware provides a left and a right stop in order to stop the drive immediately in case one of them is triggered. Therefore, pin 12 and pin 14 of the motion controller must be used.

NOTE:

→ *Both switches must be enabled before motion occurs.*

In order to enable STOPL correctly, do as follows:

Action:

- Determine the active polarity voltage of STOPL and set *pol_stop_left* (*REFERENCE_CONF* register 0x01) accordingly.
- Set *stop_left_en* =1 (*REFERENCE_CONF* register 0x01).

Result:
The current velocity ramp stops in case the STOPL voltage level matches *pol_stop_left* and *VACTUAL* < 0.

In order to enable STOPR correctly, do as follows:

Action:

- Determine the active polarity voltage of STOPR and set *pol_stop_right* (*REFERENCE_CONF* register 0x01) accordingly.
- Set *stop_right_en* =1 (*REFERENCE_CONF* register 0x01).

Result:
The current velocity ramp stops in case STOPR voltage level matches *pol_stop_right* and *VACTUAL* > 0.

8.1.1. Stop Slope Configuration for Hard or Linear Stop Slopes

The stop slope can be configured for hard or linear stop slopes. Per default, hard stops are selected.

If hard stops are required, do as follows:

OPTION 1: HARD STOP SLOPES

Action:

- Set *soft_stop_en* =0 (*REFERENCE_CONF* register 0x01).

Result:
If one of the stop switches is active and enabled, the velocity ramp is set immediately to *VACTUAL* = 0.

OPTION 2: LINEAR STOP SLOPES

If linear stop ramps are required:

Action:

- Set proper *DSTOP* > max(*DMAX*; *DFINAL*) (register 0x2C).
- Set *soft_stop_en* =1 (*REFERENCE_CONF* register 0x01).

Result:
If one of the stop switches is active and enabled, the velocity ramp is stopped with a linear deceleration slope until *VACTUAL* = 0 is reached. In this case the deceleration factor is determined by *DSTOP*. *VSTOP* is not considered during the stop deceleration slope.

8.1.2. How Active Stops are indicated and reset to Free Motion

When an enabled stop switch becomes active the related status flag is set in the *STATUS* flags register 0x0F. The flag remains active as long as the stop switch remains active.
The particular event is also released in the *EVENTS* register 0x0E, which remains active until the event bit is reset manually. When *VACTUAL* = 0 is reached after the stop event no motion toward this particular direction is possible.

In order to move into the locked direction, the following is required:

PRECONDITION 1:

The particular stop switch is NOT active anymore.

AND/OR

PRECONDITION 2:

The stop switch is disabled (*stop_left/right_en* = 0).

Action:
- Set back the active event by reading out the *EVENTS* register 0x0E.

`i` See information about clearing events provided in section 5.1. , page 21.

Result:
The active stop event is reset to free motion into the locked direction.

8.1.3. How to latch Internal Position on Switch Events

It is possible to select four different events to store the current internal position *XACTUAL* in the register *X_LATCH*.

The table below show which transition of the reference signal leads to the *X_LATCH* transfer. For each transition process the specified reference configurations in the *REFERENCE_CONF* register 0x01 must be set accordingly.

Reference Configuration	***pol_stop_left*=0**	***pol_stop_left*=1**	***pol_stop_right*=0**	***pol_stop_right*=1**
latch_x_on_inactive_l=1	STOPL=0 → 1	STOPL=1 → 0	---	---
latch_x_on_active_l=1	STOPL=1 → 0	STOPL=0 → 1	---	---
latch_x_on_inactive_r=1	---	---	STOPR=0 → 1	STOPR = 1→0
latch_x_on_active_r=1	---	---	STOPR=1 → 0	STOPR = 0→1

Table 24: Reference Configuration and Corresponding Transition of particular Reference Switch

Interchange the Reference Switches without Physical Reconnection

If you need to change the directions of the reference switches, do as follows:

Action:
- Set *invert_stop_direction*=1 (*REFERENCE_CONF* register 0x01).

Result:
STOPL is now the right reference switch and STOPR is now the left reference switch. Consequently, all configuration parameters for STOPL become valid for STOPR and vice versa.

8.2. Virtual Stop Switches

TMC4330A provides additional virtual limits; which trigger stop slopes in case the specific virtual stop switch microstep position is reached. Virtual stop positions are assigned using the *VIRTUAL_STOP_LEFT* register 0x33 and *VIRTUAL_STOP_RIGHT* register 0x34. In this section, configuration details for virtual stop switches are provided for various design-in purposes.

NOTE:

→ *Virtual stop switches must be enabled in the same manner as nonvirtual reference switches. Hitting a virtual limit switch - by receiving the assigned position - triggers the same process as hitting STOPL or STOPR.*

8.2.1. Enabling Virtual Stop Switches

In order to enable left virtual stop correctly, do as follows:

Action:
- Set *VIRTUAL_STOP_LEFT* register 0x33 according to left stop position.
- Set *virtual_left_limit_en* =1 (*REFERENCE_CONF* register 0x01).

Result:
The actual velocity ramp stops in case *XACTUAL* ≤ *VIRT_STOP_LEFT*. The ramp is stopped according to the selected ramp type.

In order to enable right virtual stop correctly, do as follows:

Action:
- Set *VIRTUAL_STOP_RIGHT* register 0x34 according to right stop position.
- Set *virtual_right_limit_en* =1 (*REFERENCE_CONF* register 0x01).

Result:
The actual velocity ramp stops in case *XACTUAL* ≥ *VIRT_STOP_RIGHT*. The ramp is stopped according to the selected ramp type.

8.2.2. Virtual Stop Slope Configuration

The virtual stop slope can also be configured for hard or linear stop slopes.

If virtual hard stops are required, do as follows:

Action:
- Set *virt_stop_mode* = b'01 (*REFERENCE_CONF* register 0x01).

Result:
If one of the virtual stop switches is active and enabled, the velocity ramp will be set immediately to *VACTUAL* = 0.

If virtual linear stop ramps are required, do as follows:

Action:
- Set proper *DSTOP* > max(*DMAX*; *DFINAL*) (register 0x2C).
- Set *virt_stop_mode* = b'10 (*REFERENCE_CONF* register 0x01).

Result:
If one of the virtual stop switches is active and enabled, the velocity ramp is stopped with a linear deceleration slope until *VACTUAL* = 0 is reached. In this case the deceleration factor is determined by *DSTOP*. *VSTOP* is not considered during the stop deceleration slope.

•→ *Continued on next page.*

8.2.3. How Active Virtual Stops are indicated and reset to Free Motion

At the same time when an enabled virtual stop switch becomes active the related status flag is activated in the *STATUS* flags register 0x0F. The flag remains active as long as the stop switch remains active.

The particular event is also released in the *EVENTS* register 0x0E, which remains active until the event is reset manually. When *VACTUAL* = 0 is reached after the stop event no motion in the particular direction is possible.

In order to move into the locked direction, the following is required:

PRECONDITION 1:

The particular stop switch is NOT active anymore because the actual position does not exceed the specified limit.

AND/OR

PRECONDITION 2:

Virtual stop switch is disabled (*virtual_left/right_limit_en* = 0).

Action:

➢ Set back active event by reading out *EVENTS* register 0x0E.

i See information about clearing events provided in section 5.1. , page 21.

Result:

The active virtual stop event bit is reset to free motion into the direction that was locked beforehand.

i *invert_stop_direction* has no influence on *VIRTUAL_STOP_LEFT* and *VIRTUAL_STOP_RIGHT*.

8.3. Home Reference Configuration

In this section home reference switch handling is explained with information about home tracking modes, possible home event configurations and home event monitoring. For monitoring, the switch reference input HOME_REF is provided.

Switch Reference Input HOME_REF

Perform the following to initiate the homing process:

Action:

- Assign a ramp according to your needs for the homing process.
- Enable the home tracking mode with *start_home_tracking* = 1 (*REFERENCE_CONF* register 0x01).
- Set the correct *home_event* (*REFERENCE_CONF* register 0x01) for the HOME_REF input pin (see table below).
- Start the ramp towards the home switch HOME_REF.

Result:

- When the next home event is recognized, *XACTUAL* is latched to *X_HOME*.
- At the same time, the *start_home_tracking* switch is disabled automatically in case *XLATCH_DONE* event is cleared.
- The *XLATCH_DONE* event is released in the events register 0x0E. This event can be used for an interrupt routine for the homing process to avoid polling.

i If an incremental encoder is used to monitor the motion, the N channel can be used to fine-tune the homing position (*home_event* = b'0000). After performing the homing process - as explained before - the N channel events can be used to obtain a more precise home position.

i *X_HOME* can be overwritten manually.

8.3.1. Home Event Selection

Nine different home events are possible.

i Except for the *home_event* = b'0000, which uses the index channel of an incremental encoder, home events are related to the the HOME_REF input pin:

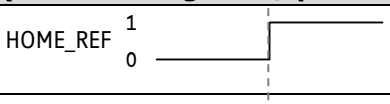
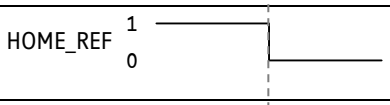
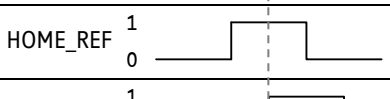
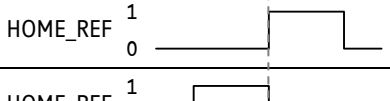
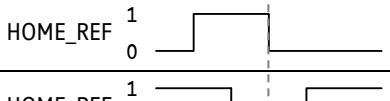
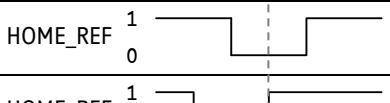
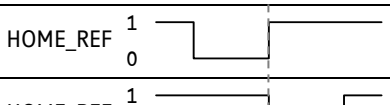
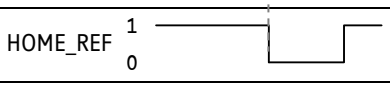
Home Event Selection Table			
home_event	**Description**		**X_HOME (direction: negative / positive)**
b'0011	*HOME_REF* = 0 indicates negative direction in reference to *X_HOME*		
b'1100	*HOME_REF* = 0 indicates positive direction in reference to *X_HOME*		
b'0110	*HOME_REF* = 1 indicates home position	X_HOME in center	
b'0010		X_HOME on the left side	
b'0100		X_HOME on the right side	
b'1001	*HOME_REF* = 0 indicates home position	X_HOME in center	
b'1011		X_HOME on the right side	
b'1101		X_HOME on the left side	

Table 25: Overview of different home_event Settings

Read entire documentation; especially the "Supplemental Directives" on page 160.

8.3.2. HOME_REF Monitoring

An error flag *HOME_ERROR_F* is permanently evaluated. This error flag indicates whether the current voltage level of the HOME_REF reference input is valid in regard to *X_HOME* and the selected home_event.

Defining a Home Range around HOME_REF

In order to avoid false error flags (*HOME_ERROR_F*) because of mechanical inaccuracies, it is possible to setup an uncertainty home range around *X_HOME*. In this range, the error flag is not evaluated.

If you want to define an uncertainty area around *X_HOME*, do as follows:

Action:

➢ Set *HOME_SAFETY_MARGIN* register 0x1E according to the required range [ustep].

Result:

The homing uncertainties – related to the application environment – are considered for the ongoing motion. The error flag is NOT evaluated in the following range:

$$X\_HOME - HOME\_SAFETY\_MARGIN \leq XACTUAL \leq X\_HOME + HOME\_SAFETY\_MARGIN$$

NOTE:

→ *It is recommended to assign to a higher range value for HOME_SAFETY_MARGIN in which the HOME_REF level is active for the home_events b'0110, b'0010, b'0100, b'1001, b'1011, and b'1101. It avoids false positive HOME_ERROR_Flags.*

→ *After homing with the index channel (home_event = b'0000) for a precise assignment of X_HOME the correct home_event has to be assigned in order to activate the generation of HOME_ERROR_Flags. Note that home_event = b'0000 results in HOME_ERROR_Flag=0 permanently.*

→ *The following examples illustrate the points at which the error flag is release – based on the selected home_event – here for home_event = b'0011 (*), b'1100 (**), b'0110 (***), b'0010 (***), b'0100 (***), b'1001 (****), b'1011 (****), and b'1101 (****).*

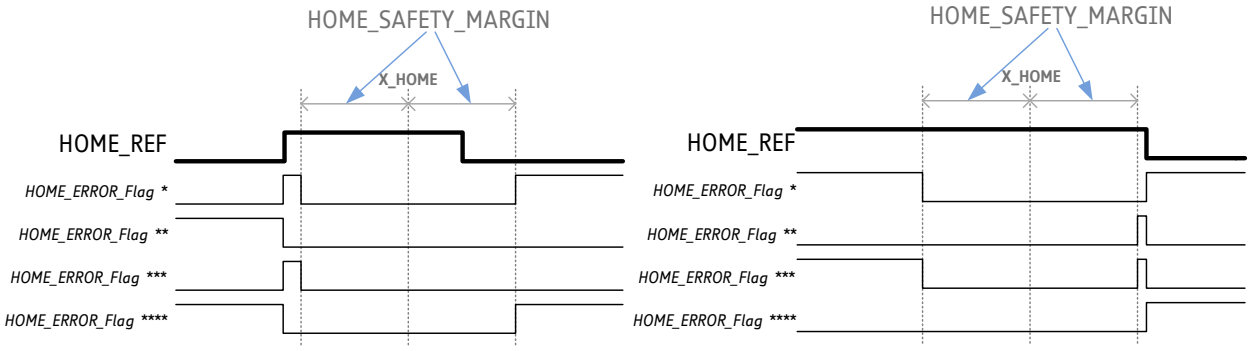


Figure 31: HOME_REF Monitoring and HOME_ERROR_FLAG

8.3.3. Homing with STOPL or STOPR

STOPL and STOPR inputs can also be used as HOME_REF inputs.

OPTION 1: STOPL IS THE HOME SWITCH

Action:

- Set *stop_left_is_home* = 1 (*REFERENCE_CONF* register 0x01).

Result:
The stop event at STOPL only occurs when the home range is crossed after STOPL becomes active. The home range is given by *X_HOME* and *HOME_SAFETY_MARGIN*.

OPTION 2: STOPR IS HOME SWITCH

Action:

- Set *stop_right_is_home* = 1 (*REFERENCE_CONF* register 0x01).

Result:
The stop event at STOPR only occurs when the home region is crossed after STOPR becomes active. The home region is given by *X_HOME* and *HOME_SAFETY_MARGIN*.

8.4. Target Reached / Position Comparison

In this section, TARGET_REACHED output pin configuration options are explained, as well as different ways how to compare different values internally.

Target Reached Output Pin

TARGET_REACHED output pin forwards the *TARGET_REACHED_Flag*. As soon as *XACTUAL* equals *XTARGET*, TARGET_REACHED is active. Per default, the TARGET_REACHED pin is high active.

To change the TARGET_REACHED output polarity, do the following:

Action:
- Set *invert_pol_target_reached* = 1 (bit16 of the *GENERAL_CONF* register 0x00).

Result:
TARGET_REACHED pin is low active.

8.4.1. Connecting several Target-reached Pins

TARGET_REACHED pins can also be configured for a shared signal line in the same way as several INTR pins can configured for one interrupt signal transfer (see section 5.4. (page 23).

To use a Wired-Or or Wired-And behavior, the below described order of action must be executed:

Action:

- **Step 1:** Set *intr_tr_pu_pd_en* = 1 (*GENERAL_CONF* register 0x00).

OPTION 1: WIRED-OR

Action:
- **Step 2:** Set *tr_as_wired_and* = 0 (*GENERAL_CONF* register 0x00).

Result:
The TARGET_REACHED pin works efficiently as Wired-Or (default configuration).

`i` In case TARGET_REACHED pin is inactive, the pin drive has a weak inactive polarity output. During active state, the output is driven strongly. Consequently, if one of the connected pins is activated, the whole line is set to active polarity.

OPTION 2: WIRED-AND

Action:

- **Step 2:** Set *tr_as_wired_and* = 1 (*GENERAL_CONF* register 0x00).

Result:
As long as the target position is not reached, the TARGET_REACHED pin has a strong inactive polarity output. During active state, the pin drive has a weak active polarity output. Consequently, the whole signal line is activated if all connected pins are forwarding the active polarity.

8.4.2. Use of TARGET_REACHED Output

Per default, TARGET_REACHED pin forwards the *TARGET_REACHED_*flag that signifies *XACTUAL* = *XTARGET*. The pin can also be used to forward three other flags: *VELOCITY_REACHED_*flag, *ENC_FAIL_*flag, *POS_COMP_REACHED_*flag.

NOTE:

→ *Only one option can be selected.*

Four Options for TARGET_REACHED

The TARGET_REACHED output pin configuration switch is available at *REFERENCE_CONF* register 0x01.

The available optons are as follows:

TARGET_REACHED Output Pin Configuration	
If *pos_comp_output*...	**Then TARGET_REACHED forwards...**
b'00	*TARGET_REACHED_*flag
b'01	*VELOCITY_REACHED_*flag
b'10	*ENC_FAIL_*flag
b'11	*POS_COMP_REACHED_*flag

Table 26: TARGET_REACHED Output Pin Configuration

8.4.3. Position Comparison of Internal Values

TMC4330A provides several ways of comparing internal values. The position comparison process is permanently active and associated with one flag and one event. A positive comparison result can be forwarded through the INTR pin using the *POS_COMP_REACHED* event as interrupt source or by using the TARGET_REACHED pin as explained in section 8.4.2, page 58.

Basic Comparison Settings

How to compare the internal position with an arbitrary value:

Action:
- Select a comparison value in the *POS_COMP* register 0x32.
- Select *pos_comp_source* = 0 (*REFERENCE_CONF* register 0x01).

Result:
XACTUAL is compared with *POS_COMP*. When *POS_COMP* equals *XACTUAL* the *POS_COMP_REACHED_Flag* becomes set and the *POS_COMP_REACHED* event becomes released.

Select External Position as Comparison Base

How to compare the external position with an arbitrary value:

Action:
- Select a comparison value in the *POS_COMP* register 0x32.
- Select *pos_comp_source* = 1 (*REFERENCE _CONF* register 0x01).

Result:
ENC_POS is compared with *POS_COMP*. When *POS_COMP* equals *ENC_POS* the *POS_COMP_REACHED_Flag* becomes set and the *POS_COMP_REACHED* event becomes released.

NOTE:

→ *Because ENC_POS represents microsteps and not encoder steps, POS_COMP represents also microsteps for the comparison process with external positions.*

→ *In case ENC_POS moves past POS_COMP without assuming the same value as POS_COMP, the POS_COMP_REACHED event is not flagged but is nonetheless listed in the EVENTS register in order to indicate that it has traversed.*

Comparison selection grid

In addition to comparing *XACTUAL* / *ENC_POS* with *POS_COMP*, it is also possible to conduct a comparison of one of both parameters with *X_HOME* or *X_LATCH* resp. *ENC_LATCH*. TMC4330A also allows comparison of the revolution counter *REV_CNT* against *POS_COMP*.

SETTINGS ALERT

Only the selected combination generates the *POS_COMP_REACHED*_Flag and the corresponding event. Therefore, select *modified_pos_compare* in the *REFERENCE_CONF* register 0x01 as outlined in the table below:

Comparison Selection Grid		
	pos_comp_source	
modified_pos_compare	**'0'**	**'1'**
'00'	*XACTUAL* vs. *POS_COMP*	*ENC_POS* vs. *POS_COMP*
'01'	*XACTUAL* vs. *X_HOME*	*ENC_POS* vs. *X_HOME*
'10'	*XACTUAL* vs. *X_LATCH*	*ENC_POS* vs. *ENC_LATCH*
'11'	*REV_CNT* vs. *POS_COMP*	

Table 27: Comparison Selection Grid to generate POS_COMP_REACHED_Flag

8.5. Repetitive and Circular Motion

TMC4330A also provides options for auto-repetitive or auto-circular motion. In this section configuration options are explained.

8.5.1. Repetitive Motion to XTARGET

Per default, reaching *XTARGET* in positioning mode finishes a positioning ramp.

In order to continuously repeat the specified ramp, do as follows:

PRECONDITION:

- Set *RAMPMODE*(2) = 1 (positioning mode is active).
- Configure a velocity ramp according to your requirements.

Action:

- Set *clr_pos_at_target* =1 (*REFERENCE_CONF* register 0x01).

Result:

After *XTARGET* is reached (*TARGET_REACHED_F*lag is active), *XACTUAL* is set to 0. As long as *XTARGET* is NOT 0, the ramp restarts in order to reach *XTARGET* again. This leads to repetitious positioning ramps from 0 towards *XTARGET*.

NOTE:

→ *It is possible to change XTARGET during repetitive motion. The reset of XACTUAL to 0 is always executed when XACTUAL equals XTARGET.*

8.5.2. Activating Circular Motion

If circular motion profiles are necessary for your application, TMC4330A offers a position limitation range of *XACTUAL* with an automatic overflow processing. As soon as *XACTUAL* reaches one of the two position range limits (positive / negative), the value of *XACTUAL* is set automatically to the value of the opposite range limit.

In order to activate circular motion, do as follows:

PRECONDITION:

If you want to activate circular motion, *XACTUAL* must be located within the defined range.

PROCEED WITH:

Action:

- Set *X_RANGE* ≠ 0 (register 0x36, only writing access!).
- Set *circular_motion* = 1 (*REFERENCE_CONF* register 0x01).

Result:

The positioning range of *XACTUAL* is limited to: $-X\_RANGE \leq XACTUAL < X\_RANGE$.

When *XACTUAL* reaches the most positive position (*X_RANGE* – 1) and the motion proceeds in positive direction; the next *XACTUAL* value is set to $-X\_RANGE$. The same applies to proceeding in negative direction; where (*X_RANGE* – 1) is the position after $-X\_RANGE$.

i During positioning mode, the motion direction will be dependent on the shortest path to the target position *XTARGET*. For example, if *XACTUAL* = 200, *X_RANGE* = 300 and *XTARGET* = −200, the positioning ramp will find its way across the overflow position (299 → −300) (see Figure A) in *Table* 27 (page 63).

8.5.3. Uneven or Noninteger Microsteps per Revolution

Due to definition of the limitation range, one revolution only consists of an even number of microsteps. TMC4330A provides an option to overcome this limitation.

- Some applications demand different requirements because a revolution consists of an uneven or noninteger number of microsteps.
- TMC4330A allows a high adjustment range of microsteps by using:
  CIRCULAR_DEC register 0x7C.
  This value represents one digit and 31 decimal places as extension for the number of microsteps per one revolution.
- A revolution is completed at overflow position. With every completed revolution the *CIRCULAR_DEC* value is added to an internal accumulation register. In case this register has an overflow, *XACTUAL* remains at its overflow position for one step.
- On average, this leads to the following microsteps per revolution:
  Microsteps/rev = $(2 \cdot X\_RANGE) + CIRCULAR\_DEC / 2^{31}$.

Example 1: Uneven Number of Microsteps per Revolution

One revolution consists of 601 microsteps.
A definition of *X_RANGE* = 300 will only provide:

600 microsteps per revolution ($-300 \le XACTUAL \le 299$).

Whereas *X_RANGE* = 301 will result in:

602 microsteps per revolution ($-301 \le XACTUAL \le 300$).

By setting:

CIRCULAR_DEC = 0x80000000 ($= 2^{31} / 2^{31} = 1$).

An overflow is generated at the decimals accumulation register with every revolution. Therefore, *XACTUAL* prolongs the step at the overflow position for one step every time position overflow is overstepped. This results in a microstep count of 601 per revolution.

Example 2: Noninteger Number of Microsteps per Revolution

One revolution consists of 600.5 microsteps.

By setting:

CIRCULAR_DEC = 0x40000000 ($= 2^{30} / 2^{31} = 0.5$).

Every second revolution an overflow is produced at the decimals' accumulation register. This leads to a microstep count of 600 every second revolution and 601 for the other half of the revolutions. On average, this leads to 600.5 microsteps per revolution.

Example 3: Noninteger and uneven Number of Microsteps per Revolution

One revolution consists of 601.25 microsteps.

By setting:

CIRCULAR_DEC = 0xA0000000 ($= (2^{31} + 2^{29}) / 2^{31} = 1.25$).

With every revolution an overflow is produced at the decimals' accumulation register. Furthermore, at every fourth revolution an additional overflow occurs, which leads to another prolonged step. This leads to a microstep count of 601 for three of four revolutions and 602 for every fourth revolution. On average, this results in 601.25 microsteps per revolution.

8.5.4. Release of the Revolution Counter

By overstepping the position overflow, the internal *REV_CNT* register is increased by one revolution as soon as *XACTUAL* oversteps from (*X_RANGE* – 1) to -*X_RANGE* or is decreased by one revolution as soon as *XACTUAL* oversteps in the opposite direction.

The information about the number of revolutions can be obtained by reading out register 0x36, which by default is the *X_LATCH* register (read only).

In order to gain information on the number of revolutions:

Action:
- Set *circular_cnt_as_xlatch* = 1 (*GENERAL_CONF* register 0x00).

Result:
Register 0x36 cease to display the *X_LATCH* value. Instead, the revolution counter *REV_CNT* can be read out at this register address.

NOTE:

→ *As soon as circular motion is inactive (*circular_motion*=0),* REV_CNT *is reset to 0.*

8.6. Blocking Zones

8.6.1. Activating Blocking Zones during Circular Motion

During circular motion, virtual stops can be used to set blocking zones. Positions inside these blocking zones are NOT dedicated for motion.

In order to activate the blocking zone, do as follows:

PRECONDITION:

Circular motion is activated (*circular_motion* = 0) and properly assigned (*X_RANGE* ≠ 0).

PROCEED WITH:

Action:
- Set *VIRTUAL_STOP_LEFT* register 0x33 as left limit for the blocking zone.
- Set *VIRTUAL_STOP_RIGHT* register 0x34 as right limit for the blocking zone.
- Enable both virtual limits as explained in section 8.2.1 (page 52).

Result:
The blocking zone reaches from *VIRTUAL_STOP_LEFT* to *VIRTUAL_STOP_RIGHT*. During positioning, the path from *XACTUAL* to *XTARGET* does not lead through the blocking zone; which can result in a longer path compared to the direct path through the blocking zone (see Figure B1 in Table 28, page 63).

However, the selected virtual stop deceleration ramp is initiated as soon as one of the limits is reached. This can result from the velocity mode or if the target *XTARGET* is located in the blocking zone.

•→ *Continued on next page.*

Blocking Zone Definition

The following positions are located within the blocking zone:

XACTUAL ≤ *VIRT_STOP_LEFT*

AND / OR

XACTUAL ≥ *VIRT_STOP_RIGHT*

NOTE:

→ *In case* VIRTUAL_STOP_LEFT < VIRTUAL_STOP_RIGHT*, one of these conditions must be met in order to be located inside the blocking zone.*

→ *In case* VIRTUAL_STOP_LEFT > VIRTUAL_STOP_RIGHT*, both conditions must be met in order to be located inside the blocking zone.*

8.6.2. Circular Motion with and without Blocking Zone

The table below shows circular motion (*X_RANGE* = 300). The green arrow depicts the path which is chosen for positioning.
The shortest path selection is shown in Figure A and the consideration of blocking zones are shown in Figures B1 and B2.

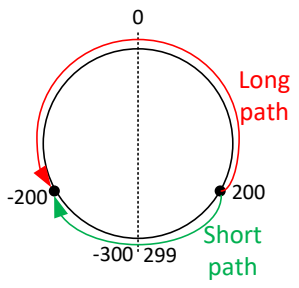
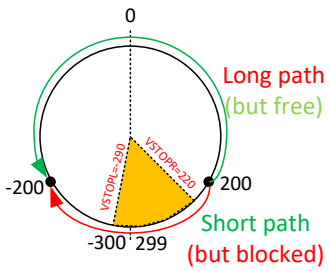
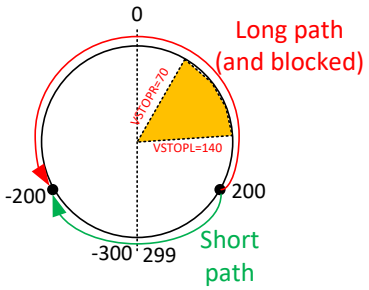
Circular Motion with (B1, B2) and Without (A) Blocking Zone		
A	**B1**	**B2**
		

Table 28: Circular motion (X_RANGE = 300)

Moving out of the Blocking Zone

When *XACTUAL* is located inside the blocking zone, it is possible to move out without redefining the blocking zone.

In order to get out of the blocking zone, do the following:

Action:

- Activate positioning mode: *RAMPMODE*(2) = 1.
- Configure velocity ramp according to your needs.
- Clear virtual stop events by reading out *EVENTS* register 0x0E.
- Set regular target position *XTARGET* outside of the blocking zone.

Result:
TMC4330A initiates a ramp with the shortest way to the target *XTARGET*.

`i` In order to match an incremental encoder in the same manner, select *circular_enc_en* =1 (*REFERENCE_CONF* register 0x01).

9. Ramp Timing and Synchronization

TMC4330A provides various options to initiate a new ramp. By default, every external register change is assigned immediately to the internal registers via an SPI input. With a proper start configuration, ramp sequences can be programmed without any intervention in between.

Synchronization Opportunities

Three levels of ramp start complexity are available. Predefined ramp starts are available, which are independent of SPI data transfer that are explained in the subsequent section 9.1. (page 65).

Two optional features can be configured that can either be used individually or combined, which are as follows:

Shadow Register Set

A complete shadow motion register set can be loaded into the actual motion registers in order to start the next ramp with an altered motion profile.

Target Position Pipeline

Different target positions can be predefined, which are then activated successively. This pipeline can be configured as cyclic; and/or it can also be utilized to sequence different parameters.

Masterless Synchronization

Also, another start state "busy" can be assigned in order to synchronize several motion controllers for one single start event without a master.

Dedicated Ramp Timing Pins		
Pin Names	**Type**	**Remarks**
START	Input and Output	External start input to get a start signal or external start output to indicate an internal start event.

Table 29: Dedicated Ramp Timing Pins

Dedicated Ramp Timing Registers			
Register Name	**Register Address**		**Remarks**
START_CONF	0x02	RW	The configuration register of the synchronization unit.
START_OUT_ADD	0x11	RW	Additional active output length of external start signal.
START_DELAY	0x13	RW	Delay time between start triggers and start signal.
X_PIPE0… 7	0x38…0x3F	RW	Target positions pipeline and/or parameter pipeline.
SH_REG0…12	0x40…0x4C	RW	Shadow register set

Table 30: Dedicated Ramp Timing Registers

9.1. Basic Synchronization Settings

Usually, a ramp can be initiated internally or externally. Note that a start trigger is not the start signal itself but the transition slope to the active start state. After a defined delay, the internal start signal is generated.

9.1.1. Start Signal Trigger Selection

For ramp start configuration, consider the following steps:

Action:

- ➢ Choose internal or external start trigger(s).
- ➢ Set the triggers according to the table below.

i All triggers can be used separately or in combination.

Start Trigger Configuration Table	
***trigger_events* = *START_CONF*(8:5)**	**Result**
b'0000	No start signal will be generated or processed further.
b'xxx0	Set *trigger_events*(0) = 0 for internal start triggers only. The internally generated start signal is forwarded to the **START pin** that is assigned as **output**.
b'xxx1	Set *trigger_events*(0) = 1 for an external start trigger. The **START pin** is assigned as **input**. For START input take filter settings into consideration. See chapter 4, page 17.
b'xx1x	TARGET_REACHED event is assigned as start signal trigger for the ramp timer.
b'x1xx	VELOCITY_REACHED event is assigned as start signal trigger for the ramp timer.
b'1xxx	POSCOMP_REACHED event is assigned as start signal trigger for the ramp timer.

Table 31: Start Trigger Configuration

9.1.2. User-specified Impact Configuration of Timing Procedure

Per default, every SPI datagram is processed immediately. By selecting one of the following enable switches, the assignment of SPI requests to registers *XTARGET*, *VMAX*, *RAMP_MODE,* and *GEAR_RATIO* is uncoupled from the SPI transfer. The value assignment is only processed after an internally generated start signal.

In order to influence the impact of the start signal on internal parameter assignments, do the following:

Action:

- ➢ Choose between the following options as shown in the table below.

Start Enable Switch Configuration Table *(All switches can be used separately or in combination.)*	
start_en* = START_CONF*(4:0)	**Result**
b'xxxx1	*XTARGET* is altered only after an internally generated start signal.
b'xxx1x	*VMAX* is altered only after an internally generated start signal.
b'xx1xx	*RAMPMODE* is altered only after an internally generated start signal.
b'x1xxx	*GEAR_RATIO* is altered only after an internally generated start signal.
b'1xxxx	Shadow register is assigned as active ramp parameters after an internally generated start signal. This is explained in more detail in section 9.2. (page 70).

Table 32: Start Enable Switch Configuration

9.1.3. Delay Definition between Trigger and internally generated Start Signal

Per default, the trigger is closely followed by the internal start signal.

In order to delay the generation of the internal start signal, do the following:

Action:
➢ Set *START_DELAY* register 0x13 according to your specification.

Result:
When a start trigger is recognized, the internal start signal is generated after *START_DELAY* clock cycles.

Prioritizing External Input

Per default, an external trigger is also delayed for the internal start signal generation.

In order to immediately prompt an external start, trigger to an internally generated start signal (regardless of a defined delay), do the following:

Action:
➢ Set *immediate_start_in* = 1 (*START_CONF* register 0x02).

Result:
When an external start trigger is recognized, the internal start signal is generated immediately, even if the internal start triggers have already initiated a timing process with an active delay.

START Pin Polarity

The START pin can be used either as input or as output pin. However, the active voltage level polarity of the START pin can be selected with one configuration switch in the *START_CONF* register 0x02.

Per default, the voltage level transition from high to low triggers a start signal (START is an input), or START output indicates an active START event by switching from high to low level.

In order to invert active START polarity, do as follows:

Action:
➢ Set *pol_start_signal* = 1 (*START_CONF* register 0x02).

Result:
The START pin is high active. The voltage level transition from low to high triggers a start signal (START is an input), or START output indicates an active START event by switching from low to high level.

9.1.4. Active START Pin Output Configuration

Per default, the active output voltage level of the START pin lasts one clock cycle.

In order to extend this time span, do the following:

Condition:
➢ START pin is assigned as output: *trigger_events*(0) = 1.

Action:
➢ Set *START_OUT_ADD* register 0x11 according to your specification.

Result:
The active voltage level lasts (*START_OUT_ADD* + 1) clock cycles.

Read entire documentation; especially the "Supplemental Directives" on page 160.

9.1.5. Ramp Timing Examples

The following three examples depict SPI datagrams, internal and external signal levels, corresponding velocity ramps, and additional explanations. SPI data is transferred internally at the end of each datagram.

Ramp Timing Example 1

In this example, the velocity value change is executed immediately.

Process Description

- The new *XTARGET* value is assigned after TARGET_REACHED has been set and *START_DELAY* has elapsed.
- A new ramp does not start at the end of the second ramp because no new *XTARGET* value is assigned.
- START is an output.
- Internal start signal forwards with a step length of (*START_OUT_ADD* + 1) clock cycles.

This is how external devices can be synchronized:

Parameter Settings Timing Example 1	
Parameter	**Setting**
RAMPMODE	b'101
start_en	b'00001
trigger_events	b'0010
START_DELAY	>0
START_OUT_ADD	>0
pol_start_signal	1

Table 33: Parameter Settings Timing Example 1

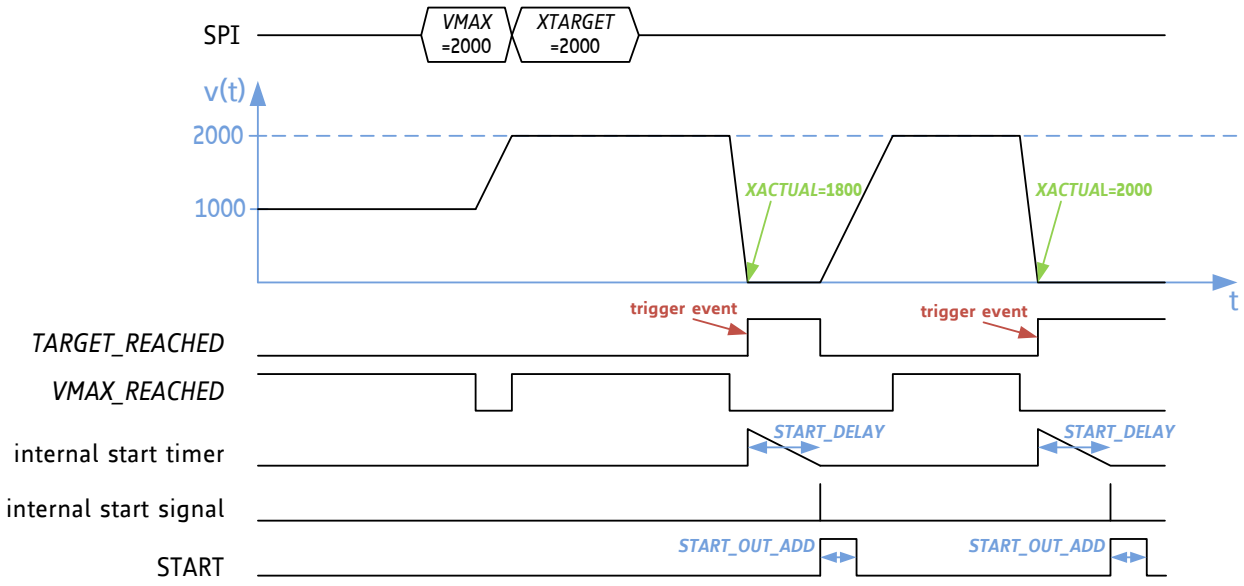


Figure 32: Ramp Timing Example 1

Read entire documentation; especially the "Supplemental Directives" on page 160.

Ramp Timing Example 2

In this example, the velocity value and the ramp mode value change is executed after the first start signal.

Process Description

- The new ramp mode becomes positioning mode with S-shaped ramps.
- The ramp then stops at target position *XTARGET* because of the ramp mode change.
- A further *XTARGET* change starts the ramp again.
- The ramp is initiated as soon as the start delay is completed, which was triggered by the first *TARGET_REACHED* event.
- The active START output signal lasts only one clock cycle.

Parameter Settings Timing Example 2	
Parameter	**Setting**
RAMPMODE	b'001 → b'110
start_en	b'00111
trigger_events	b'0110
START_DELAY	>0
START_OUT_ADD	0
pol_start_signal	0

Table 34: Parameter Settings Timing Example 2

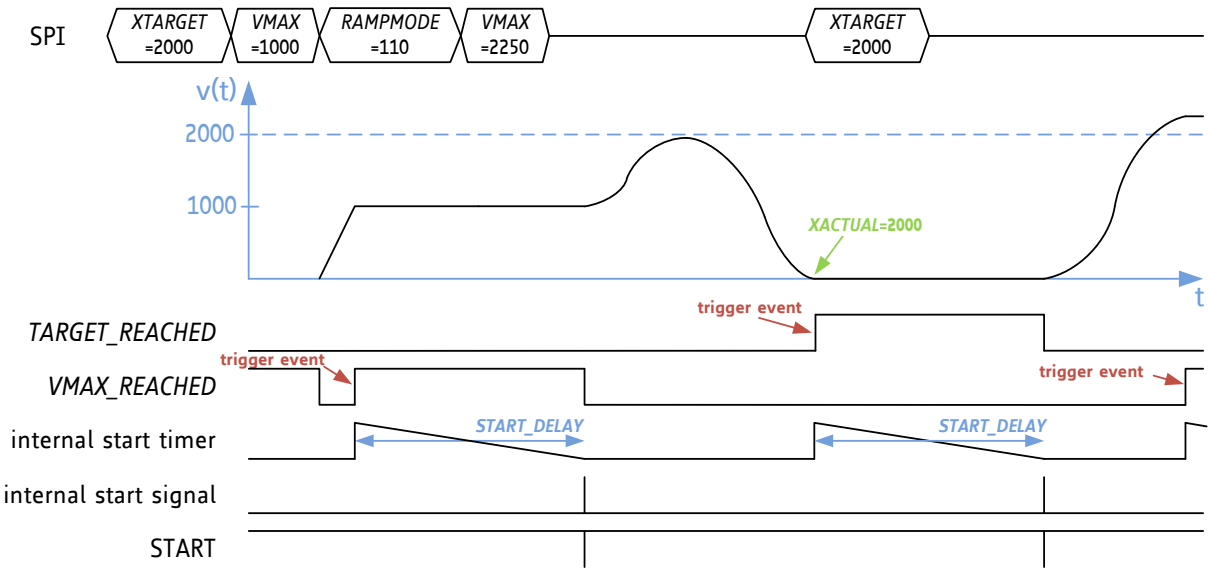


Figure 33: Ramp Timing Example 2

Read entire documentation; especially the "Supplemental Directives" on page 160.

Ramp Timing Example 3

In this example external start signal triggers are prioritized by making use of *START_DELAY* > 0 and simultaneously setting *immediate_start_in* to 1.

Process Description

- When *XACTUAL* equals *POSCOMP* the start timer is activated and the external start signal in between is ignored.
- The second start event is triggered by an external start signal. The *POSCOMP_REACHED* event is ignored.

  The third start timer process is disrupted by the external START signal, which is forced to be executed immediately due to the setting of: *immediate_start_in* = 1.

Parameter Settings Timing Example 3	
Parameter	**Setting**
RAMPMODE	b'000
start_en	b'00010
trigger_events	b'1001
immediate_start_in	0 → 1
START_DELAY	>0
pol_start_signal	1

Table 35: Parameter Settings Timing Example 3

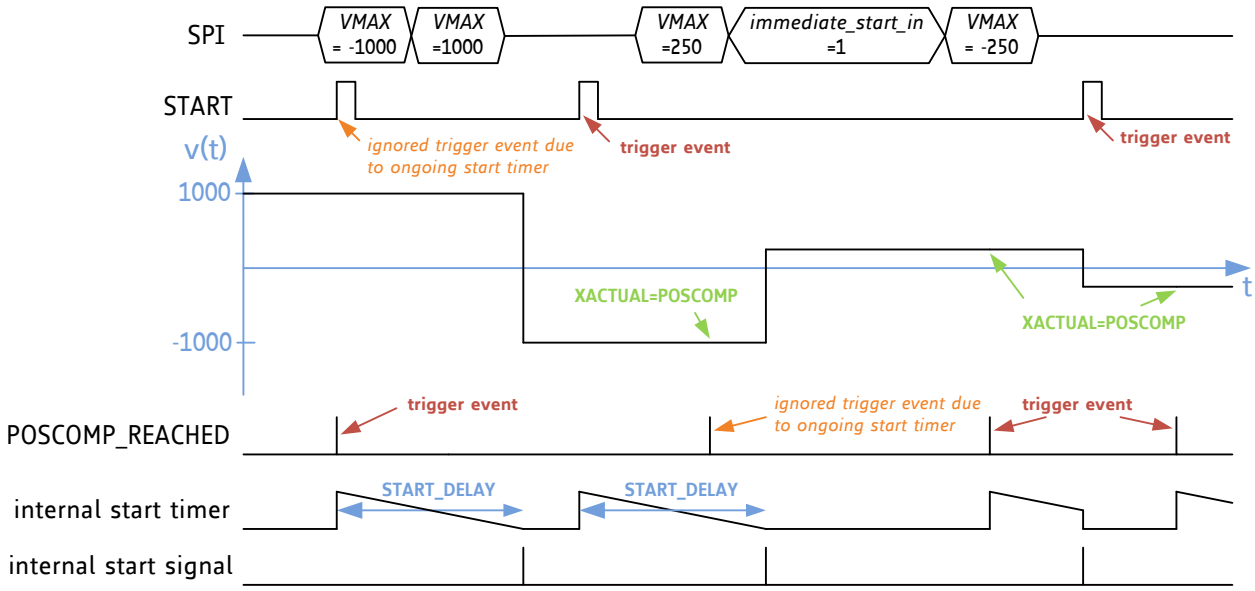


Figure 34: Ramp Timing Example 3

9.2. Shadow Register Settings

Some applications require a complete new ramp parameter set for a specific ramp situation / point in time. TMC4330A provides up to 14 shadow registers, which are loaded into the corresponding ramp parameter registers after an internal start signal is generated.

Enabling Shadow Registers

In order to enable shadow registers, do as follows:

Action

- Set *start_en*(4) = 1 and select one or more *trigger_events* (*START_CONF* register 0x02), see section 9.1.2 (page 65).

Result:
With every successive internal start signal the shadow registers are loaded into the corresponding active ramp register.

Enabling Cyclic Shadow Registers

It is also possible to write back the current motion profile into the shadow motion registers to swap ramp motion profiles continually.

In order to enable cyclic shadow registers, do as follows:

Action

- Set *start_en*(4) = 1 and select one or more *trigger_events* (*START_CONF* register 0x02) , see section 9.1.2 (page 65).
- Set *cyclic_shadow_regs* = 1 (*START_CONF* register 0x02).

Result:
With every successive internal start signal the shadow registers are loaded into the corresponding active ramp register, whereas the active motion profile is loaded into the shadow registers.

•→ *Continued on next page.*

9.2.1. Shadow Register Configuration Options

Four different optional shadow register assignments are available to match the shadow register set according to your selected ramp type. The available options are described on the next pages.

i Please note that the only difference between the configuration of shadow option 3 and 4 is that *VSTART* is exchanged by *VSTOP* for the transfer of the shadow registers.

Option 1: Shadow Default Configuration

If the whole ramp register is needed to set in a single level stack, do as follows:

Action:

- Set *shadow_option* = b'00 (*START_CONF* register 0x02).
- Set *start_en*(4) = 1 and select one or more *trigger_events* (*START_CONF* register 0x02)

Action:

- **Default configuration:** Set *cyclic_shadow_regs* = 0 (*START_CONF* register 0x02)
- **Optional configuration:** Set *cyclic_shadow_regs* = 1 (*START_CONF* register 0x02)

Result:

Every relevant motion parameter is altered at the next internal start signal by the corresponding shadow register parameter. In case cyclic shadow registers are used, the shadow register set is altered by the current motion profile set.

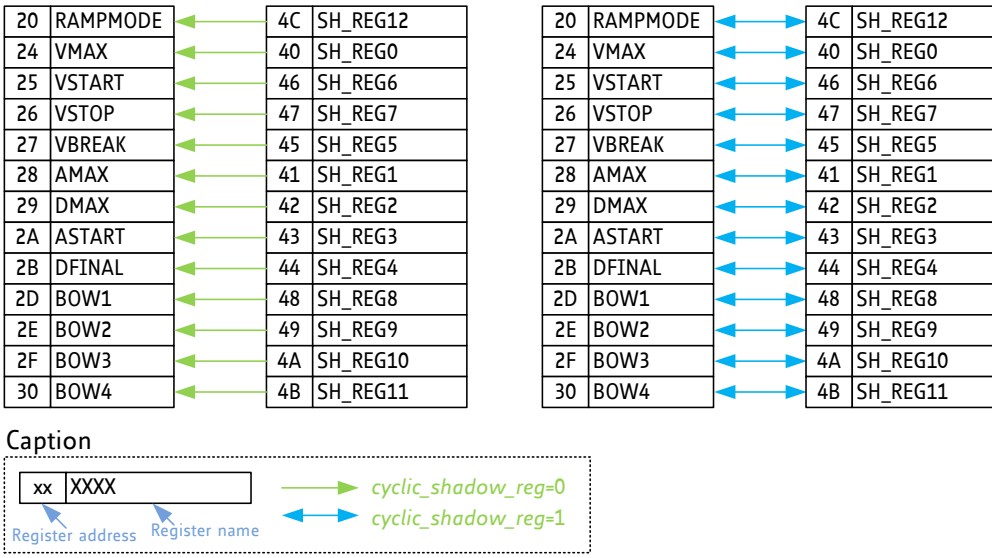


Figure 35: Single-level Shadow Register Option to replace complete Ramp Motion Profile.

i Green arrows show default settings

i Blue arrows show optional settings.

AREAS OF SPECIAL CONCERN

In case an S-shaped ramp type is selected and operation mode is switched from velocity to positioning mode (triggered by shadow register transfer), *SH_REG10* must not be equal to *BOW3*; to ensure safe operation mode switching.

•→ On the following pages more options are explained. Pleae turn page.

Option 2: Double-stage Shadow Register Set for S-shaped Ramps

In case S-shaped ramps are configured, a double-stage shadow register set can be used. Seven relevant motion parameters for S-shaped ramps are affected when the shadow registers become active.

In order to use a double-stage shadow register pipeline for S-shaped ramps, do as follows:

Action:

- Set *shadow_option* = b'01 (*START_CONF* register 0x02).
- Set *start_en*(4) = 1 and select one or more *trigger_events* (*START_CONF* register 0x02).

Action:

- **Default configuration:** Set *cyclic_shadow_regs* = 0 (*START_CONF* register 0x02).
- **Optional configuration:** Set *cyclic_shadow_regs* =1 (*START_CONF* register 0x02)

Result:

Seven motion parameters (*VMAX*, *AMAX*, *DMAX*, *BOW1...4*) are altered at the next internal start signal by the corresponding shadow register parameters (*SH_REG0...6*). Simultaneously, these shadow registers are exchanged with the parameters of the second shadow stage (*SH_REG7…13*).

In case cyclic shadow registers are used, the second shadow register set (*SH_REG7…13*) is altered by the current motion profile set, e.g. 0x28 (*AMAX*) is written back to 0x48 (*SH_REG8*).

The other ramp registers remain unaltered.

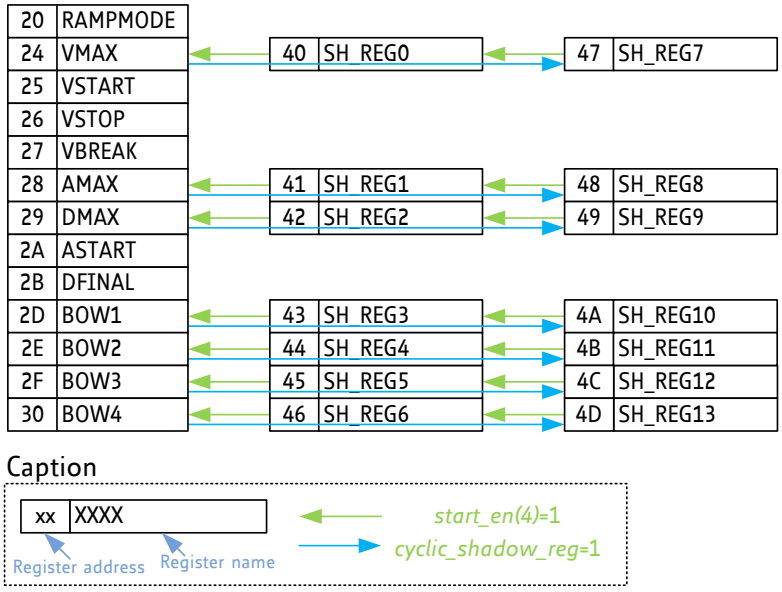


Figure 36: Double-stage Shadow Register Option 1, suitable for S-shaped Ramps.

i Green arrows show default settings

i Blue arrows show optional settings.

•→ Description is continued on next page.

Option 3: Double-stage Shadow Register Set for Trapezoidal Ramps (*VSTART*)

In case trapezoidal ramps are configured, a double-stage shadow register set can be used. Seven relevant motion parameters for trapezoidal ramps are affected when the shadow registers become active.

In order to use a double-stage shadow register pipeline for trapezoidal ramps, do as follows:

Action:

- Set *shadow_option* = b'10 (*START_CONF* register 0x02).
- Set *start_en*(4) = 1 and select one or more *trigger_events* (*START_CONF* register 0x02)

Action:

- **Default configuration:** Set *cyclic_shadow_regs* = 0 (*START_CONF* register 0x02).
- **Optional configuration:** Set *cyclic_shadow_regs* = 1 (*START_CONF* register 0x02).

Result:

Seven motion parameters (*VMAX*, *AMAX*, *DMAX, ASTART, DFINAL, VBREAK*, and *VSTART*) are altered at the next internal start signal by the corresponding shadow register parameters (*SH_REG0...6*). Simultaneously, these shadow registers are exchanged with the parameters of the second shadow stage (*SH_REG7…13*).
If cyclic shadow registers are used, the second shadow register set (*SH_REG7…13*) is altered by the current motion profile set, e.g. 0x27 (*VBREAK*) is written back to 0x4C (*SH_REG12*). The other ramp registers remain unaltered.

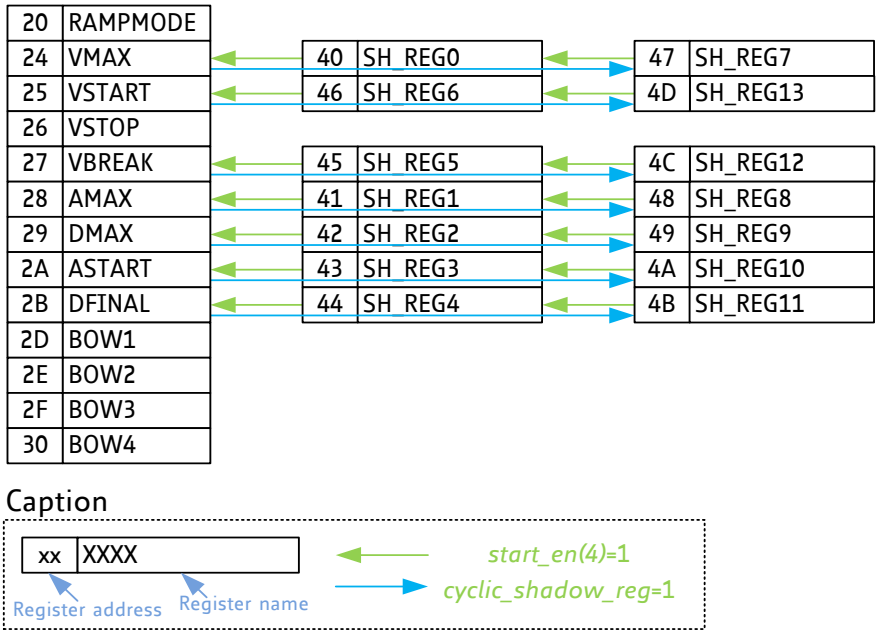


Figure 37: Double-stage Shadow Register Option 2, suitable for Trapezoidal Ramps.

i Green arrows show default settings.

i Blue arrows show optional settings.

•→ Description is continued on next page.

Option 4: Double-stage Shadow Register Set for Trapezoidal Ramps (*VSTOP*)

In case trapezoidal ramps are configured, a double-stage shadow register set can be used. Seven relevant motion parameters for trapezoidal ramps are affected when the shadow registers become active.

In order to use a double-stage shadow register pipeline for trapezoidal ramps, do as follows:

Action:

- Set *shadow_option* = b'10 (*START_CONF* register 0x02).
- Set *start_en*(4) = 1 and select one or more *trigger_events* (*START_CONF* register 0x02)

Action:

- **Default configuration:** Set *cyclic_shadow_regs* = 0 (*START_CONF* register 0x02).
- **Optional configuration:** Set *cyclic_shadow_regs* = 1 (*START_CONF* register 0x02)

Result:

Seven motion parameters (*VMAX*, *AMAX*, *DMAX*, *ASTART*, *DFINAL*, *VBREAK*, and *VSTOP*) are altered at the next internal start signal by the corresponding shadow register parameters (*SH_REG0...6*). Simultaneously, these shadow registers are exchanged with the parameters of the second shadow stage (*SH_REG7…13*).
If cyclic shadow registers are used, the second shadow register set (*SH_REG7…13*) is altered by the current motion profile set, e.g. 0x26 (*VSTOP*) is written back to 0x4D (*SH_REG13*). The other ramp registers remain unaltered.

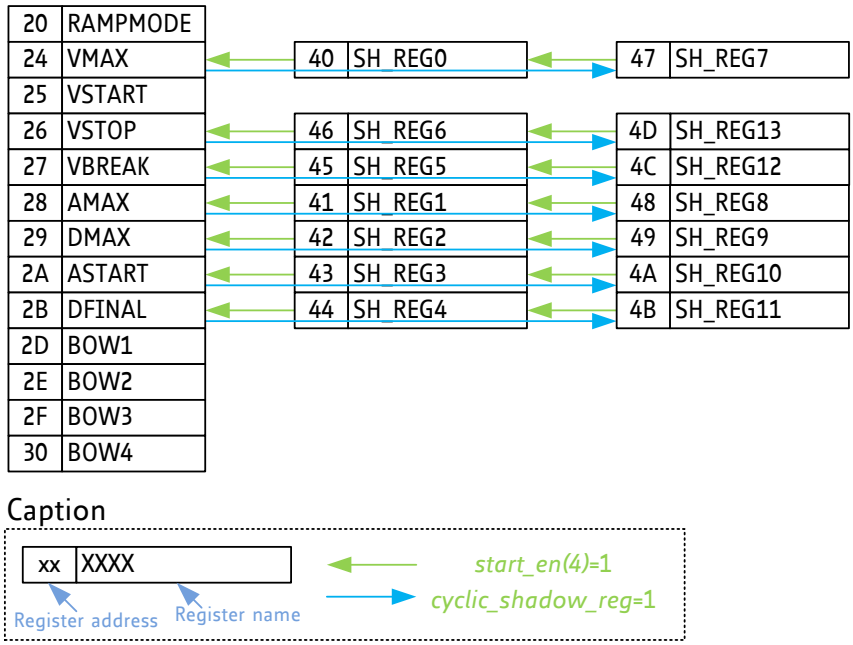


Figure 38: Double-Stage Shadow Register Option 3, suitable for Trapezoidal Ramps

i Green arrows show default settings.

i Blue Arrows show optional settings.

*•→ Turn page to see **Areas of Special Concern** pertaining to this section.*

AREAS OF SPECIAL CONCERN

The values of ramp parameters, which are not selected by one of the four shadow options stay as originally configured, until the register is changed through an SPI write request.
Also, the last stage of the shadow register pipeline retains the values until they are overwritten by an SPI write request if no cyclic shadow registers are selected.

9.2.2. Delayed Shadow Transfer

Up to 15 internal start signals can be skipped before the shadow register transfer is executed.

In order to skip a defined number of internal start signals for the shadow transfer, do as follows:

Action:
- Set *shadow_option* according to your specification.
- Set *start_en*(4) = 1 and select one or more *trigger_events* (*START_CONF* register 0x02)
- ***OPTIONAL CONFIGURATION:*** Set *cyclic_shadow_regs* = 1.
- Set *SHADOW_MISS_CNT* ≠ 0 (*START_CONF* register 0x02) according to the number of consecutive internal start signals that you specify to be ignored.

Result:
The shadow register transfer is not executed with every internal start signal. Instead, the specified number of start signals is ignored until the shadow transfer is executed through the (*SHADOW_MISS_CNT*+1)th start signal.

The following figure shows an example of how to make use of *SHADOW_MISS_CNT*, in which the shadow register transfer is illustrated by an internal signal sh_reg_transfer. The signal miss counter *CURRENT_MISS_CNT* can be read out at register address *START_CONF* (23:20):

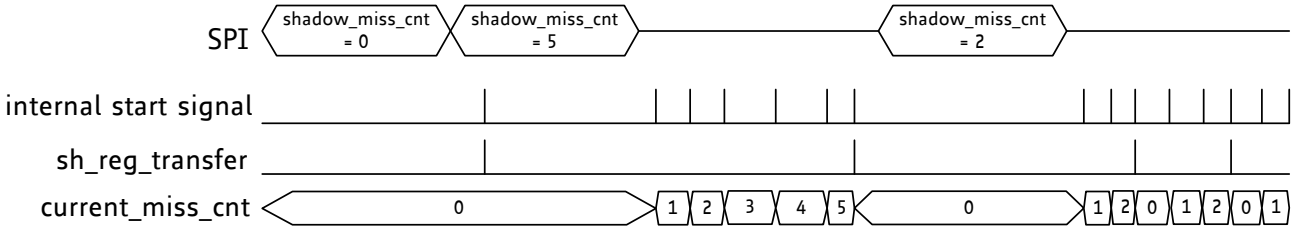


Figure 39: SHADOW_MISS_CNT Parameter for several internal Start Signals

AREAS OF SPECIAL CONCERN

Internal calculations to transfer the requested shadow BOW values into internal structures require at most (320 / f_{CLK}) [sec]. before any shadow register transfer is prompted, it is necessary to wait for the completion of all internal calculations for the shadow bow parameters.

In order to make this better understood the following example is provided for a double-stage shadow pipeline for S-shaped ramps:

PRECONDITION:

Shadow register transfer is activated (*start_en*(1) = 1 and one or more *trigger_events* are selected) for S-shaped ramps (*shadow_option* = b'01)

Action
- Set *SH_REG0*, *SH_REG1*, *SH_REG2* (shadow register for *VMAX*, *AMAX*, *DMAX*).
- Set *SH_REG3*, *SH_REG4*, *SH_REG5*, *SH_REG6* (shadow register for *BOW1…4*).
- Ensure that no shadow register transfer occurs during the next 320 / f_{CLK} [s].

Result:
Shadow register transfer can be initiated after this time span.

9.3. Pipelining Internal Parameters

TMC4330A provides a target pipeline for sequencing subordinate targets in order to easily arrange a complex target structure.

9.3.1. Configuration and Activation of Target Pipeline

The different target values must be assigned to the *X_PIPE0…7* register. If the target pipeline is enabled, a new assignment cycle is initiated as soon as an internal start signal is generated; moving the values*, as described,* simultaneously:

PROCESS DESCRIPTION:

- A new *XTARGET* value is assigned that takes over the value of *X_PIPE0.*
- Every *X_PIPE*n register takes over the value of its successor:
  *X_PIPE*n = *X_PIPE*n+1

In order to activate the target pipeline, do as follows:

Action:

➢ Set *pipeline_en* = b'0001 (*START_CONF* register 0x02).

Result:

The above mentioned process description is executed with every new internal start signal prompting.

Configuration of a cyclic Target Pipeline

It is also possible to reassign the value of *XTARGET* to one (or more) of the pipeline registers *X_PIPE0…7.* Thereby, a cyclic target pipeline is created.

In order to enable a cyclic target pipeline, do as follows:

Action:

➢ Set *pipeline_en* = b'0001 (*START_CONF* register 0x02).

➢ Set *XPIPE_REWRITE_REG* in relation to the pipeline register where *XTARGET* have to written back (e.g. *XPIPE_REWRITE_REG* = b'00010000).

Result:

The above mentioned process description is executed with every new internal start signal prompting, and *XTARGET* is written back to the selected *X_PIPEx* register (e.g. *XPIPE_REWRITE_REG* = 0x10 → *XTARGET* is written back to *X_PIPE4*).
The processes and actions described on the previous page, are depicted in the following figure. The assignment cycle that is initiated when an internal start signal occurs is depicted.

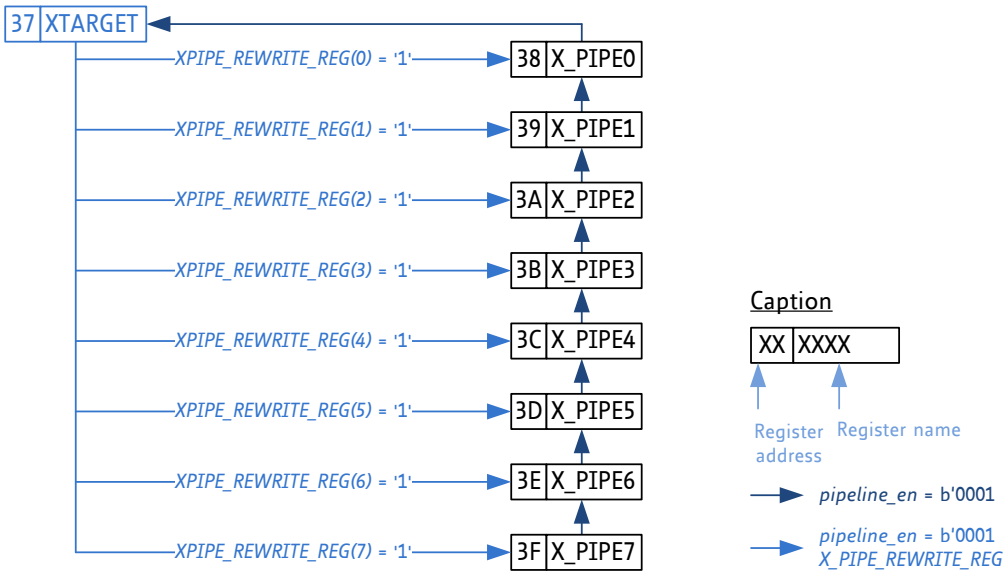


Figure 40: Target Pipeline with Configuration Options

Read entire documentation; especially the "Supplemental Directives" on page 160.

9.3.2. Using the Pipeline for different internal Registers

The TMC4330A pipeline (registers 0x38…0x3F) can be configured so that it splits up into maximal four segments. These segments can be used to feed the following internal parameters:

- *XTARGET* register 0x37
- *POS_COMP* register 0x32
- *GEAR_RATIO* register 0x12
- *GENERAL_CONF* 0x00

Consequently, these definite parameter value changes can be of importance concerning a continuous ramp motion and/or for reduced overhead synchronizing of several motion controllers.

The *POS_COMP* value can be used to initiate a start signal generation during motion. Therefore, it can be useful to pipeline this parameter in order to avoid dependence on SPI transfer speed.
For instance, if the distance between two *POS_COMP* values is very close and the current velocity is high enough that it misses the second value before the SPI transfer is finished, it is advisable to change *POS_COMP* immediately after the start signal.

The same is true for the *GEAR_RATIO* parameter, which defines the step response on incoming step impulses. Some applications require very quick gear factor alteration of the slave controller. Note that when the start signal is prompted directly, an immediate change can be very useful instead of altering the parameter by an SPI transfer.

Likewise, it can (but must not) be essential to change general configuration parameters at a defined point in time. A suitable application is a clearly defined transfer from a direct external control (*sd_in_mode* = b'01) to an internal ramp (*sd_in_mode* = b'00) or vice versa because in this case the master/slave relationship is interchanged.

The following pipeline options are available, which can be adjusted accordingly:

Pipeline Activation Options	
***pipeline_en*(3:0)**	**Description**
b'xxx1	Pipeline for *XTARGET* is enabled.
b'xx1x	Pipeline for *POS_COMP* is enabled.
b'x1xx	Pipeline for *GEAR_RATIO* is enabled.
b'1xxx	Pipeline for *GENERAL_CONF* is enabled.

Table 36: Pipeline Activation Options

9.3.3. Pipeline Mapping Overview

The *pipeline_en* parameter offers an open configuration for 16 different combinations of the pipeline segregation. As a result, the number of pipelines range from 0 to 4. This also has an impact on the pipeline depth. The possible options are as follows: eight stages, four stages, three stages and two stages.

In the "Pipeline Mapping" table below, the arrangement and depth of the pipeline is allocated according to the pipeline setup. The final register destination of pipeline registers are also depicted in order to illustrate from which pipeline registers (*X_PIPE0…7*) the final target registers (*XTARGET*, *POS_COMP*, *GEAR_RATIO*, *GENERAL_CONF*) are fed.

For example, if *POS_COMP* and *GEAR_RATIO* are chosen as parameters that are to be fed by the pipeline, two 4-stage pipelines are created. When an internal start signal is generated, *POS_COMP* assumes the value of *X_PIPE0*, whereas *X_PIPE4* feeds the *GEAR_RATIO* register.

But if *POS_COMP*, *GEAR_RATIO* and *XTARGET* are selected as parameter destinations, two 3-stage pipelines and one double-stage pipeline are created. When an internal start signal is generated, *XTARGET* assumes the value of *X_PIPE0*, *POS_COMP* assumes the value of *X_PIPE3*, whereas *X_PIPE6* feeds the *GEAR_RATIO* register.

Pipeline Mapping Table

More examples are described in detail on the following pages - *explaining some of the possible configurations and referencing examples* - listed in the Table below.

Pipeline Mapping						
Ex.	***pipeline_en* (3:0)**	**Arrangement**	*Final transfer register for…*			
			GENERAL_CONF →pipeline_en(3)	*GEAR_RATIO* →pipeline_en(2)	*POS_COMP* →pipeline_en(1)	*XTARGET* →pipeline_en(0)
-	*b'0000*	*No Pipelining*	-	-	-	-
-	*b'0001*	*One 8-stage pipeline*	-	-	-	X_PIPE0
A	*b'0010*		-	-	X_PIPE0	-
B	*b'0100*		-	X_PIPE0	-	-
-	*b'1000*		X_PIPE0	-	-	-
C	*b'0011*	*Two 4-stage pipelines*	-	-	X_PIPE4	X_PIPE0
-	*b'0101*		-	X_PIPE4	-	X_PIPE0
-	*b'1001*		X_PIPE4	-	-	X_PIPE0
-	*b'0110*		-	X_PIPE4	X_PIPE0	-
-	*b'1010*		X_PIPE4	-	X_PIPE0	-
D	*b'1100*		X_PIPE4	X_PIPE0	-	-
F	*b'0111*	*Two 3-stage pipelines and one double-stage pipeline*	-	X_PIPE6	X_PIPE3	X_PIPE0
-	*b'1011*		X_PIPE6	-	X_PIPE3	X_PIPE0
E	*b'1101*		X_PIPE6	X_PIPE3	-	X_PIPE0
-	*b'1110*		X_PIPE6	X_PIPE3	X_PIPE0	-
G/H	*b'1111*	*Four double-stage pipelines*	X_PIPE6	X_PIPE4	X_PIPE2	X_PIPE0

Table 37: Pipeline Mapping for different Pipeline Configurations

Read entire documentation; especially the "Supplemental Directives" on page 160.

9.3.4. Cyclic Pipelining

For all of the above shown configuration examples, it is possible to write back the current values of the selected registers (*XTARGET*, *POS_COMP*, *GEAR_RATIO* and/or *GENERAL_CONF*) to any of the pipeline registers of their assigned pipeline in order to generate cyclic pipelines.
By selecting proper *XPIPE_REWRITE_REG*, the value that is written back to the pipeline register is selected automatically to fit the selected pipeline mapping.

9.3.5. Pipeline Examples

Below, several pipeline mapping examples with the corresponding configuration are shown.

Examples A+B: Using one Pipeline

Example A: Cyclic pipeline for *POS_COMP,* which has eight pipeline stages.

Example B: Cyclic pipeline for *GEAR_RATIO,* which has six pipeline stages.

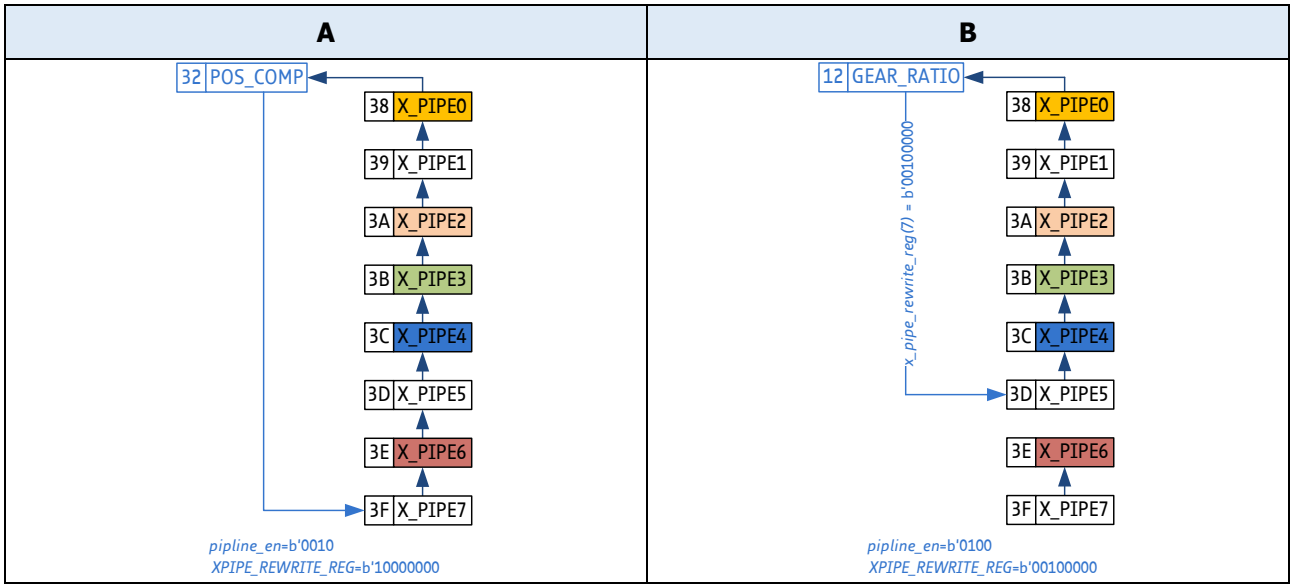


Figure 41: Pipeline Example A

Figure 42: Pipeline Example B

Examples C+D: Using two Pipelines

Example C: Cyclic pipelines for *XTARGET* and *POS_COMP,* which have four pipeline stages each.

Example D: Cyclic pipelines for *GEAR_RATIO,* which has three pipeline stages and *GENERAL_CONF,* which has two pipeline stages.

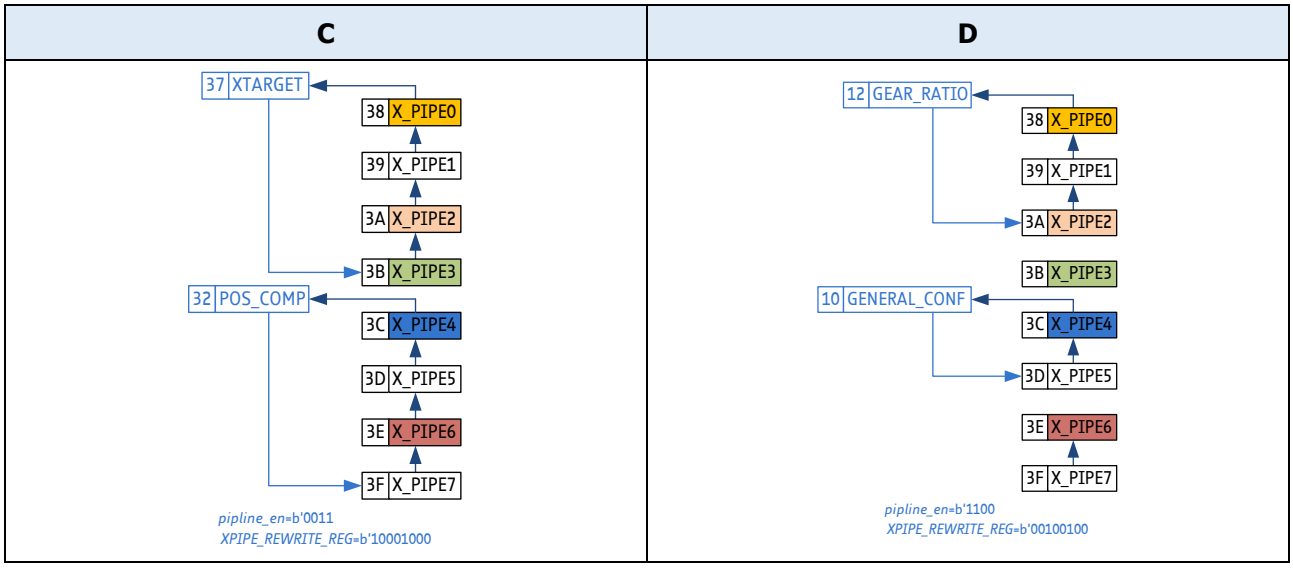


Figure 43: Pipeline Example C

Figure 44: Pipeline Example D

Read entire documentation; especially the "Supplemental Directives" on page 160.

Examples E+F: Using three Pipelines

Example E: Cyclic pipelines for *XTARGET* and *GEAR_RATIO,* which have three pipeline stages each and *GENERAL_CONF,* which has two pipeline stages.

Example F: Two cyclic pipelines for *XTARGET* and *GEAR_RATIO*, which have two pipeline stages each and a noncyclic pipeline for *GEAR_RATIO,* which has three pipeline stages.

E

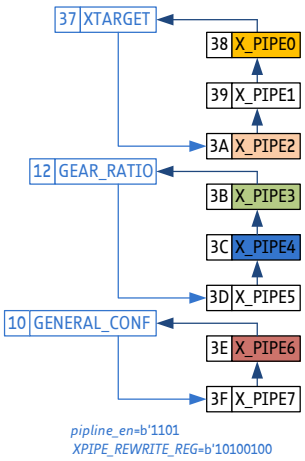


Figure 45: Pipeline Example E

F

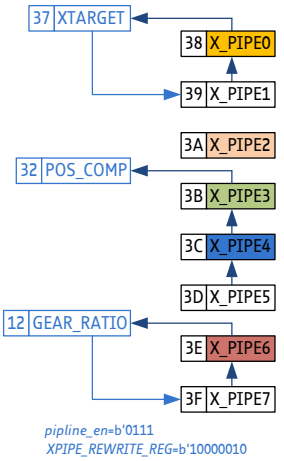


Figure 46: Pipeline Example F

Examples G+H: Using four Pipelines

Example G: Cyclic pipelines for *XTARGET*, *POS_COMP*, *GEAR_RATIO* and *GENERAL_CONF,* which have two pipeline stages each.

Example H: Four noncyclic pipelines for *XTARGET*, *POS_COMP*, *GEAR_RATIO* and *GENERAL_CONF,* which have two pipeline stages each.

G

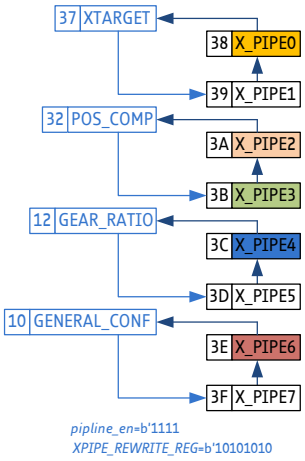


Figure 47: Pipeline Example G

H

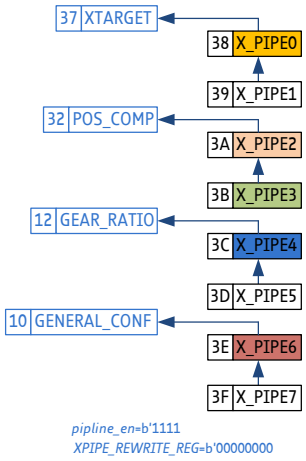


Figure 48: Pipeline Example H

Read entire documentation; especially the "Supplemental Directives" on page 160.

9.4. Masterless Synchronization of Several Motion Controllers via START Pin

START pin can also be assigned as tristate input in order to synchronize several microcontroller masterless.

Activation of the Tristate START Pin

In this case START is assigned as tristate. A busy state is enabled. During this busy state, START is set as output with a strongly driven inactive polarity. If the internal start signal is generated – after the internal start timer is expired –START pin is assigned as input. Additionally, a weak output signal is forwarded at START. During this phase, the active start polarity is emitted.

In case the signal at START input is set to active polarity, because all members of the signal line are ready, START output remains active (strong driving strength) for *START_OUT_ADD* clock cycles.

Then, busy state is active again until the next start signal occurs.

In order to activate tristate START pin, do as follows:

Action:
- Set *busy_en* = 1 (*START_CONF* register 0x02).

Result:
The above mentioned process description is executed.

START Pin Connection

In case START pin is connected with START pins of other TMC4330A devices, it is recommend that a series resistor (e.g. 220 Ω) is connected between the devices to limit the short circuit current flowing that can flow during the configuration phase when different voltage levels at the START pins of the different devices can occur.

NOTE:

→ *Avoid that short circuits last too long.*

Read entire documentation; especially the "Supplemental Directives" on page 160.

10. Controlled PWM Output

TMC4330A offers controlled PWM (Pulse Width Modulation) signals at STPOUT and DIROUT output pins. These PWM signals are generated by using the internal microstep look-up table (MSLUT) that can be adapted according to the design specifications, see section 10.3. , page 85. Additionally, these PWM vanlues can be scaled, depending on the internal velocity.

Dedicated PWM Output Pins		
Pin Names	**Type**	**Remarks**
STPOUT_PWMA	Output	PWM output for coil A.
DIROUT_PWMB	Output	PWM output for coil B.

Table 38: Dedicated PWM Output Pins

Dedicated PWM Output Registers			
Register Name	**Register Address**		**Remarks**
GENERAL_CONF	0x00	RW	Bit 21: *pwm_out_en.*
CURENT_CONF	0x05	RW	*pwm_scale_en* = *SCALE_CONF* (8): PWM scale enable switch *PWM_AMPL* = *SCALE_CONF* (31:16): PWM amplitude at *VACTUAL* = 0.
PWM_VMAX	0x17	RW	Second assignment to *VDRV_SCALE_LIMIT*: velocity at which the PWM scale parameter reaches 1 (maximum).
PWM_FREQ	0x1F	RW	Number of clock cycles that forms one PWM period.

Table 39: Dedicated PWM Output Registers

10.1.1. How to change Motion Direction

Per default, a positive internal velocity *VACTUAL* results in a forward motion through internal MSLUT. Consequently, if *VACTUAL* < 0, the MSLUT values are developed backwards.

In order to alter the default setting of the Internal Motion Direction, do as follows:

Action:

➢ Set *reverse_motor_dir* =1 (bit28 of *GENERAL_CONF* register 0x00).

Result:

A positive internal velocity for *VACTUAL* results in a backward motion through the internal MSLUT.

10.1.2. Change of Microstep Resolution

The MSLUT is based on 256 micorsteps per fullstep. By altering the microstep resolution from 256 (*MSTEP_PER_FS* = b'0000) to a lower value, an internal step results in more than one MSLUT step.
For instance, if the microstep resolution is set to 64 (*MSTEP_PER_FS* = b'0010), MSCNT is either increased or decreased by 4 per each internal step. Accordingly, the passage through the MSLUT skips three current values per each internal step to match the new microstep resolution

10.2. PWM Output Generation and Scaling Possibilities

Enable PWM Output Generation

The STPOUT and DIROUT output pins generally forward internal generated microsteps and motion direction. In contrast to that, it is possible to forward the internal MSLUT value as PWM output signals, which is dependent on the PWM frequency.

In order to generate PWM output, do as follows:

Action:
- Set *PWM_FREQ* register 0x1F to the number of clock cycles for one PWM cycle.
- Set *pwm_out_en* = 1 (*GENERAL_CONF* register 0x00).

Result:
Step/Dir output is disabled and PWM signals are forwarded via STPOUT_PWMA and DIROUT_PWMB. PWM frequency f_{PWM} is calculated by:

$$f_{PWM} = f_{CLK} / PWM\_FREQ$$

If PWM Voltage mode is selected:

NOTICE

Avoid unintended overheating to prevent motor damage during PWM mode!

- **At lower velocity values PWM voltage scaling MUST be enabled.**

This will ensure smooth operation during controlled PWM mode.

PWM Duty Cycle Scaling

The duty cycle of both signals represent the sine (STPOUT) and cosine (DIROUT) values of the MSLUT.
PWM scaling is adapted linearly, which depends on the internal ramp velocity. During Voltage PWM mode the scaling value at *VACTUAL* = 0 must be assigned, and also the velocity at which full scaling is reached.

In order to generate a scaled PWM output, do as follows:

Action:
- Set *PWM_AMPL* (bit31:16 of register 0x05) as start PWM scaling value.
- Set *PWM_VMAX* register 0x17 to the internal ramp velocity [pps] at which full PWM scaling is reached.
- Set *pwm_scale* = 1 (bit8 of *SCALE_CONF* register 0x05).

Result:
- PWM_SCALE is the actual scaling value.
- In case *VACTUAL* = 0, PWM_SCALE = (*PWM_AMPL* + 1) / 2^{17}.

i Whenever the absolute velocity value increases, the scale parameter also increases linearly until it reaches the maximum of PWM_SCALE = 0.5 at *VACTUAL* = *PWM_VMAX*.

i The minimum duty cycle is calculated by DUTY_MIN = (0.5 – PWM_SCALE).

i The maximum duty cycle is calculated by DUTY_MAX = (0.5 + PWM_SCALE).

i These values set the PWM duty cycle limits of any internal ramp velocity.

Read entire documentation; especially the "Supplemental Directives" on page 160.

10.2.1. PWM Scale Example

In *Figure* 54 below, the calculation of minimum/maximum PWM duty cycles with *PWM_AMPL* = 32767 is shown on the left side. Resulting duty cycles for different positions in the sine voltage curve are depicted on the right side. Calculated delays of minimum/maximum duty cycles are also shown.

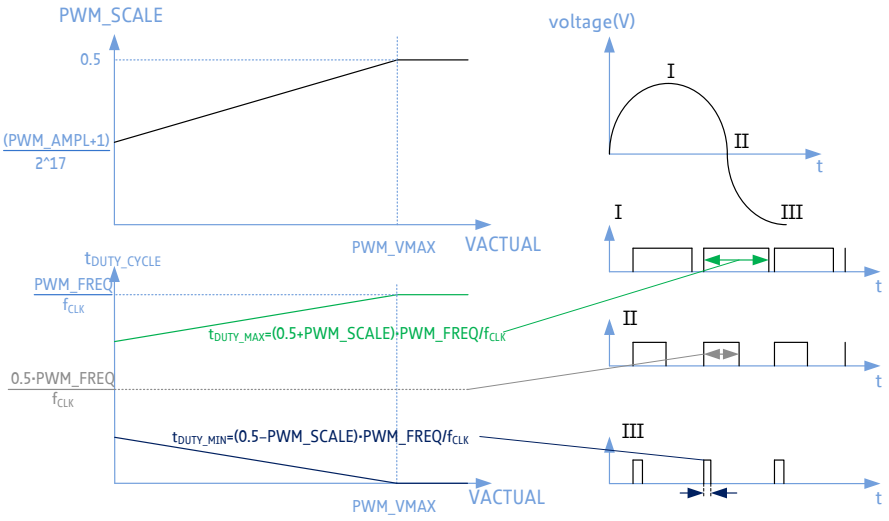


Figure 49: Calculation of PWM Duty Cycles (PWM_AMPL)

Read entire documentation; especially the "Supplemental Directives" on page 160.

10.3. Microstep Lookup Tables

TMC4330A provides a programmable lookup table (LUT) for storing the microstep values, which are the basis for the Voltage PWM output. Reprogramming the table from its predefined values to a motor-specific wave allows improved motor-reliant microstepping, particularly when using low-cost motors.

SETTINGS ALERT

TMC4330A-LA provides a default configuration of the internal microstep table MSLUT. The following explanations that are provided in this section only address engineers who use their own microstep table definition.

Programming Sine Wave Lookup Tables

The internal microstep wave table maps the microstep wave from 0° to 90° for 256 microsteps. It becomes automatically and symmetrically extended to 360° that consequently comprises 1024 microsteps. As a result, the microstep counter *MSCNT* ranges from 0 to 1023. Only a quarter of the wave is stored because this minimizes required memory and the amount of programmable data.
Therefore, only 256 bits (*ofs*00 to *ofs*255) are required to store the quarter wave. These bits are mapped to eight 32-bit registers *MSLUT*[0] (register 0x70) to *MSLUT*[7] (register 0x77).
When reading out the table the 10-bit microstep counter *MSCNT* addresses the fully extended wave table.

Sine Wave Table Structure

The MSLUT is an incremental table. This means that a certain order and succession is predefined at every next step based on the value before, using up to four flexible programmable segments within the quarter wave. The microstep limits of the four segments are controlled by the position registers X1, X2, and X3.
Within these segments the next value of the MSLUT is calculated by adding the base wave inclination Wx-1 (if *ofs*=0) or its successor Wx (if *ofs*=1). Because four segments are programmable, four base wave inclinations are available as basic increment value: 0, 1, 2, or 3. Thereby, even a negative wave inclination can be realized. This is shown in the next Figure where the values in last quarter segments are decreased or remain constant with every step towards *MSCNT*= 255.

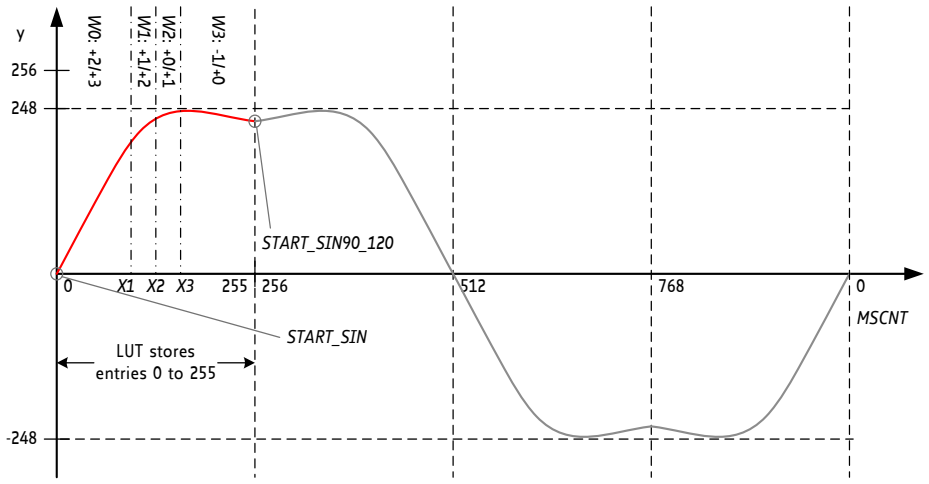


Figure 50: LUT Programming Example

Read entire documentation; especially the "Supplemental Directives" on page 160.

10.3.1. Actual Microstep Values Output

Actual Microstep Calculations

When the microstep sequencer advances within the microstep table (MSLUT), it calculates the actual microstep values for the motor coils with each microstep, and stores them to the register 0x7A , which comprises the values of both waves *USTEPA* and *USTEPB*. However, the incremental coding requires an absolute initialization – especially when the microstep table becomes modified. Therefore, *USTEPA* and *USTEPB* become re-initialized with the start values whenever *MSCNT* passes zero.

Characteristics of a 2-phase Stepper Motor Microstep Table

As mentioned above, the MSLUT can be adapted to the motor requirements. In order to understand the nature of incremental coding of the microstep table, the characteristics of the microstep wave must be understood, as described in the list below:

Characteristics of a 2-phase motor microstep table:

- In principle, it is a reverse characteristic of the motor pole behavior.
- It is a polished wave to provide a smooth motor behavior. There are no jumps within the wave.
- The phase shift between both phases is exactly 90°, because this is the optimum angle of the poles inside the motor.
- The zero transition is at 0°. The curve is symmetrical within each quadrant (like a sine wave).
- The slope of the wave is normally positive, but due to torque variations it can also be (slightly) negative.
- But it must not be strictly monotonic as shown in the figure above.

Considering these facts, it becomes clear that the wave table can be compressed. The incremental coding applied to the TMC4330A uses a format that reduces the required information - *per entry of the 8-bit by a 256-entry wave table* - to slightly more than a single bit.

10.3.2. How to Program the Internal MSLUT

Principle of Incremental Encoding

The principle of **incremental encoding** only stores the difference between the actual and the next table entry. In order to attain an absolute start value, the first entry is directly stored in *START_SIN*. Also, for ease-of-use, the first entry of the shifted table for the second motor phase is stored in *START_SIN_90_120*.

Based on these start values, every next table entry is calculated by adding an increment INC to the former value. This increment is the base wave inclination value Wx whenever its corresponding *ofs* bit is 1 or Wx – 1 if *ofs* = 0:

INC = Wx + (ofs – 1).

The base wave inclination can be set to four different values (0, 1, 2, 3), because it consists of two bits.

Because the wave inclination does not change dramatically, TMC4330A provides four wave inclination segments with the base wave inclinations (W0, W1, W2, and W3) and the segment borders (0, X1, X2, X3, and 255), as shown in the left quarter of the MSLUT diagram in *Figure* 48, page 85.

Wave Inclination Characteristics		
Wave Inclination Segment	**Base Wave Inclination**	**Segment Ranges**
0	W0	0 … X1
1	W1	X1… X2
2	W2	X2 … X3
3	W3	X3 … 255

Table 40: Wave Inclination Characteristics of Internal MSLUT

Read entire documentation; especially the "Supplemental Directives" on page 160.

10.3.3. Setup of MSLUT Segments

Base Wave Inclination and Border Values

All base wave inclination values (each consists of two bits) as well as the border values (each consists of eight bit) between the segments are adjustable. They are assigned by *MSLUTSEL* register 0x78.

In order to change the base wave inclination values and the segment borders, do as follows:

Action:

- Define the segment borders X1, X2, and X3 and the base wave inclination values W0…W3 according to the requirements
- Set register *MSLUTSEL*(31:24) = X3.
- Set register *MSLUTSEL*(23:16) = X2.
- Set register *MSLUTSEL*(15:8) = X1.
- Set register *MSLUTSEL*(7:6) = W3.
- Set register *MSLUTSEL*(5:4) = W2.
- Set register *MSLUTSEL*(3:2) = W1.
- Set register *MSLUTSEL*(1:0) = W0.

Result:
The segments and the base wave inclination values of the internal MSLUT are changed.

NOTE:

→ *It is not mandatory to define four segments. For instance, if only two segments are required, set X2 and X3 to 255. Then, W0 is valid for segment 0 between MSCNT = 0 and MSCNT = X1, and W1 is valid between MSCNT = X1 and MSCNT = 255 (segment 1).*

In order to change the *ofs* bits, do as follows:

Action:

- Set *MSLUT*[0] register 0x70 = *ofs*31…*ofs*00.
- Set *MSLUT*[1] register 0x71 = *ofs*63…*ofs*32.
- Set *MSLUT*[2] register 0x72 = *ofs*95…*ofs*64.
- Set *MSLUT*[3] register 0x73 = *ofs*127…*ofs*96.
- Set *MSLUT*[4] register 0x74 = *ofs*159…*ofs*128.
- Set *MSLUT*[5] register 0x75 = *ofs*191…*ofs*160.
- Set *MSLUT*[6] register 0x76 = *ofs*223…*ofs*192.
- Set *MSLUT*[7] register 0x77 = *ofs*255…*ofs*224.

Result:
The *ofs* bits of the internal MSLUT are changed.

AREAS OF SPECIAL CONCERN

When modifying the wave:
Special care has to be applied in order to ensure a smooth and symmetrical zero transition whenever the quarter wave becomes expanded to a full wave.

Zero Crossing

When adjusting the range:
The maximum resulting swing of the wave should be adjusted to a range of −248 to 248, in order to achieve the best possible resolution while at the same time leaving headroom for a hysteresis based chopper to add an offset.

Read entire documentation; especially the "Supplemental Directives" on page 160.

10.3.4. Microstep Waves Start Values

Starting Microstep Values of MSLUT Configuration

As both waves are shifted by 90° for two-phase stepper motors, the sine wave starts at 0° when *MSCNT* = 0. By comparison, the cosine wave begins at 90° when *MSCNT* = 256. At this starting points the microstep values are *USTEPA* = 0 for the sine wave and *USTEPB* = 247 for the cosine wave.

In contrast to the starting microstep positions that are fixed, these starting microstep values can be redefined if the default start values do not fit for the actual MSLUT.

In order to change the starting microstep values of the MSLUT, do as follows:

Action:

- Define the start values *START_SIN* and *START_SIN90_120* according to the requirements.
- Set register 0x7E (7:0) = *START_SIN*
- Set register 0x7E (23:16) = *START_SIN90_120*

Result:
The starting values for both waves are adapted to MSLUT.

10.3.5. Default MSLUT

Base Wave Inclinations

The default sine wave table in TMC drivers uses one segment with a base inclination of 2 and one segment with a base inclination of 1 (see default value of the *MSLUTSEL* register 0x78 = 0xFFFF8056).

The segment border X1 is located at *MSCNT* = 128. The base wave inclinations are

W0 = b'10 (=2) and W1 = b'01 (=1).

As a result, between *MSCNT* = 0 and 128, the increment value INC is either

1 (if *ofs* = 0) or 2 (if *ofs* = 1).

And between *MSCNT* = 128 and 255, the increment value INC is either

0 (if *ofs* = 0) or 1 (if *ofs* = 1).

This reflects the stronger rise in the first segment of the MSLUT in contrast to the second segment. The maximum value is

START_SIN90_120 = 247.

10.3.6. Explanatory Notes for Base Wave Inclinations

Definition of Segments 0,1,2,3

In the following example four segments are defined.

Each segment has a different base wave inclination to illustrate each possible entry:

Segment 0: W0 = 3 which means that the increment value is +2 or +3.
Segment 1: W0 = 2 which means that the increment value is +1 or +2.
Segment 2: W0 = 1 which means that the increment value is 0 or +1.
Segment 3: W0 = 0 which means that the increment value is −1 or 0.

i In addition to the MSLUT curve (black line), which is defined by the given *ofs* bits, all four segments show upper limits (red line); in case all *ofs* bits in the particular segments are set to 1.

i The green line shows the lower limit in case all *ofs* bits in the particular segments are set to 0.

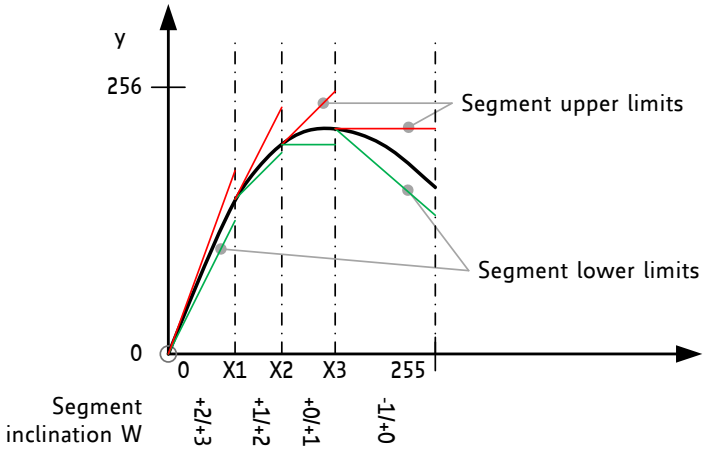


Figure 51: MSLUT Curve with all possible Base Wave Inclinations (highest Inclination first)

Standard Sine Wave Setup Considerations prior to SETUP of MSLUT

In order to set up a standard sine wave table for the MSLUT, the following considerations have to be taken into account:

PRECONSIDERATIONS:

- The microstep table for the standard sine wave begins with eight entries (0 to 7) {0, 1, 3, 4, 6, 7, 9, 10 …} etc.
- The maximum difference between two values in this section is +2, whereas the minimum difference is +1.
- While advancing according to the table, the very first time the difference between two MSLUT values is lower than +1 is between position 153 and position 154. Both entries are identical.
- The start value is 0 for the sine wave.
- The calculated value for position 256 (i.e. start of cosine wave) is 247.

•→ Description is continued on next page.

Standard Sine Wave Setup

In order to set up the standard sine wave table, proceed as follows:

Action:

- Set a starting value *START_SIN* = 0 matching sine wave entry 0.
- Set a base wave inclination range of W0 = b'10 = 2 to skip between +1 / +2, valid from 0 to X1.
- Calculate the differences between every entry: {+1, +2, +1, +2, +1, +2, +1,…}.
- Set the microstep table entries *ofs*XX to 0 for the lower value (+1); 1 for the higher value (+2). Thus, the first seven microstep table entries *ofs*00 to *ofs*06 are: {0, 1, 0, 1, 0, 1, 0 …}
- The base wave inclination must be lowered at position 153, at very latest. Use the next base wave inclination range 1 with W1 = b'01 = 1 to skip between +0 and +1.
- Set X1 = 153 in order to switch to the next inclination range. From here on, an offset *ofs*XX of 0 means add nothing; 1 means add +1.
- Set *START_SIN90_120* = 247, which is equal to the value at position 256.
- Only two of four wave segments with different base wave inclinations are used. The remaining wave inclination ranges W2 and W3 should be set to the same value as W1; and X2 and X3 can be set to 255. Thereby, only two wave inclination segments are effective.

Result:

A standard sine wave is defined as MSLUT. The following table shows an extract of this curve.

Overview of the Microstep Behavior Example													
Microstep number	0	1	2	3	4	5	6	7	…	153	154	…	255
Desired table entry	0	1	3	4	6	7	9	10	…	200	200	…	247
Difference to next entry	1	2	1	2	1	2	1	…	…	0	…	…	0
Required segment inclination	+2	+2	+2	+2	+2	+2	+2	…	…	+1	…	…	+1
***Ofs* bit entry**	0	1	0	1	0	1	0	…	…	0	…	…	0

Table 41: Overview of the Microstep Behavior Example

Read entire documentation; especially the "Supplemental Directives" on page 160.

11. Decoder Unit: Connecting ABN, SSI, or SPI Encoders correctly

TMC4330A is equipped with an encoder input interface for incremental ABN encoders, absolute SSI or SPI encoders. This chapter provides basic setup information for correct analysis of connected encoder signals.

Decoder Pins		
Pin Names	**Type**	**Remarks**
A_SCLK	Input or Output	A signal of ABN encoder or Serial Clock output for absolute SSI, or SPI encoders.
ANEG_NSCLK	Input or Output	Negated A signal of ABN encoder or Negated Serial Clock output for SSI encoder or Low active Chip Select signal for SPI encoders.
B_SDI	Input	B signal of ABN encoder or Serial Data Input of SSI, or SPI encoders.
BNEG_NSDI	Input or Output	Negated B signal of ABN encoder or Negated Serial Data Input of SSI encoders or Serial Data Output of SPI encoder.
N	Input	N signal of ABN encoder.
NNEG	Input	Negated N signal of ABN encoder.

Table 42: Dedicated Decoder Unit Pins

Decoder Unit Registers			
Register Name	**Register address**		**Remarks**
GENERAL_CONF	0x00	RW	Bit11:10: serial_enc_in_mode, Bit12: diff_enc_in_disable
INPUT_FILT_CONF	0x03	RW	Input filter configuration (SR_ENC_IN, FILT_L_ENC_IN).
ENC_IN_CONF	0x07	RW	Encoder configuration register.
ENC_IN_DATA	0x08	RW	Serial encoder input data structure.
ENC_POS	0x50	RW	Current absolute encoder position in microsteps.
ENC_LATCH	0x51	R	Latched absolute encoder position.
ENC_POS_DEV	0x52	R	Deviation between *XACTUAL* and *ENC_POS*.
ENC_CONST	0x54	R	Internally calculated encoder constant.
Encoder Register Set	0x51…58 0x62…63	W	Encoder configuration parameter.
Encoder velocity	0x65 0x66	R	Current encoder velocity (signed). Current filtered encoder velocity (signed).
ADDR_TO_ENC *DATA_TO_ENC*	0x68 0x69	W	Serial encoder request data.
ADDR_FROM_ENC *DATA_FROM_ENC*	0x6A 0x6B	R	Serial encoder request data response.
Encoder compensation	0x7D	W	Encoder compensation register set.

Table 43: Dedicated Decoder Unit Registers

Read entire documentation; especially the "Supplemental Directives" on page 160.

11.1.1. Selecting the correct Encoder

The encoder interface consists of six pins that can be connected with different encoder types. Depending on the encoder type, the pins serve as inputs or as outputs. If inputs are assigned, the incoming signals can be filtered, as explained in chapter 4, page 17. Consequently, *SR_ENC_IN* and *FILT_L_ENC_IN* must be set accordingly. In the following, three options are presented to select a connected encoder properly.

OPTION 1: INCREMENTAL ABN ENCODERS

In order to set up a connected incremental ABN encoder, do as follows:

Action:
- Set *serial_enc_in_mode* = b'00 (*GENERAL_CONF* register 0x00).

Result:
An incremental ABN encoder is selected.

OPTION 2: ABSOLUTE SSI ENCODERS

In order to set up a connected absolute SSI encoder, do as follows:

Action:
- Set *serial_enc_in_mode* = b'01 (*GENERAL_CONF* register 0x00).

Result:
An absolute SSI encoder is selected.

`i` In order to avoid an erroneous status of the connected absolute SSI encoder, a proper configuration is necessary prior to enabling; as described further down below on the subsequent pages: see section 15.4. on page 99.

OPTION 3: ABSOLUTE SPI ENCODERS

In order to set up a connected absolute SPI encoder:

Action:
- Set *serial_enc_in_mode* = b'11 (*GENERAL_CONF* register 0x00).

Result:
An absolute SPI encoder is selected.

`i` In order to avoid an erroneous status of the connected absolute SPI encoder, a proper configuration is necessary prior to enabling; as described further down below on the subsequent pages: see section 15.4. on page 99.

•→Turn page for encoder pin assignment overview.

11.1.2. Disabling digital differential Encoder Signals

If incremental ABN or absolute SSI encoders are selected, the dedicated encoder signals are treated as digital differential signals per default. For internally displaying a valid input level, the levels of a dedicated pair must be digitally inversed.

i No analog differential circuit is available.

In order to disable the digital differential input signals, do as follows:

Action:
➢ Set *diff_enc_in_disable* = 1 (*GENERAL_CONF* register 0x00).

Result:
Dedicated encoder signals are treated as single signals and every negated pin is ignored.

i Concerning absolute SPI encoders, this is done automatically.

Pin Assignment based on selected Encoder Setup						
Pin No.	**Pin Name**	**Incremental ABN**		**Absolute SSI**		**Absolute SPI**
		Differential	**Single-ended**	**Differential**	**Single-ended**	**Single-ended**
40	A_SCLK	A	A	SCLK	SCLK	SCLK
1	ANEG_NSCLK	¬A	-	¬SCLK	-	CS
10	B_SDI	B	B	SDI	SDI	SDI
11	BNEG_NSDI	¬B	-	¬SDI	-	SDO
21	N	N	N	-	-	-
22	NNEG	¬N	-	-	-	-

Table 44: Pin Assignment based on selected Encoder Setup

11.1.3. Inverting of Encoder Direction

In order to easily align the encoder direction with the motor direction it is possible to invert the encoder direction by setting one switch.

In order to invert the encoder direction, do as follows:

Action:
➢ Set *invert_direction* = 1 (*ENC_CONF* register 0x07).

Result:
The calculation of the in external position *ENC_POS* is inverted, turning increment to decrement and vice versa.

Read entire documentation; especially the "Supplemental Directives" on page 160.

11.1.4. Encoder Misalignment Compensation

If the encoder is installed correctly, the encoder values form a circle for one motor revolution. Thus, the deviation *ENC_POS_DEV* between real position *ENC_POS* und internal position *XACTUAL* forms a constant function over the whole motor revolution. Consequently, the resulting form of a deficiently installed encoder is oval-shaped. This system failure results in a new function of *ENC_POS_DEV* that is similar to a sine function. In figure A below, the position deviation is shown as function of one motor revolution, which comprises 51200 microsteps.
TMC4330A provides an option to compensate this kind of misalignment by adding a triangular shape function that counteracts the system error. This can improve the encoder value evaluation significantly. Per default, this function is constant at 0.

In order to setup the triangular compensation function, do as follows:

Action:

- Set proper *ENC_COMP_XOFFSET* register 0x7D (15:0).
- Set proper *ENC_COMP_YOFFSET* register 0x7D (23:16).
- Set proper *ENC_COMP_AMPL* register 0x7D (31:24).

Result:
ENC_COMP_XOFFSET is 16-bit register which represents a numeral figure between 0 and 1. The resulting offset on the abscissa is calculated by:

XOFF_LOW = *ENC_COMP_XOFFSET* · microsteps/rev / 65536.

A triangular function is generated, which has its **lowest point at (XOFF_LOW; *ENC_COMP_YOFFSET*)**.
The peak is shifted at a distance of half a revolution. The **peak coordinate (XOFF_PEAK;YOFF_PEAK)** is calculated as follows:

XOFF_PEAK = *ENC_COMP_XOFFSET* · microsteps/rev / 65536 + microsteps/rev / 2.

YOFF_PEAK = *ENC_COMP_YOFFSET* + *ENC_COMP_AMPL*.

In figure A below, the red line illustrates this compensation function.
Internally, the triangular function is added to the *ENC_POS* value. As a result, the position deviation is harmonized as a function of the motor revolution; which can be seen in figure B below.

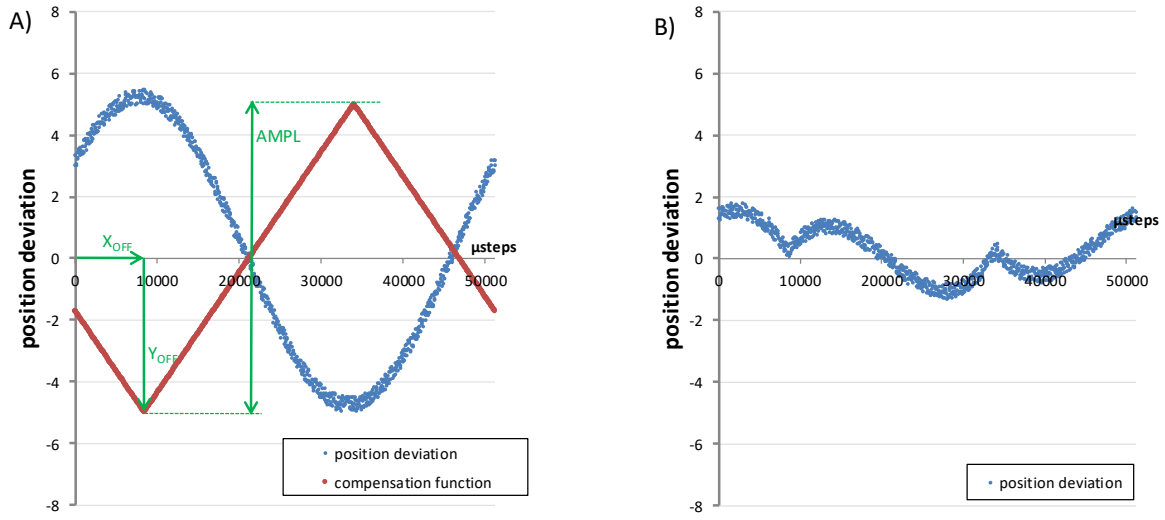


Figure 52: Triangular Function that compensates Encoder Misalignments

Read entire documentation; especially the "Supplemental Directives" on page 160.

11.2. Incremental ABN Encoder Settings

Incremental ABN encoders increment or decrement the external position counter register *ENC_POS* 0x50. This is based on A- and B-signal level transitions.

11.2.1. Automatic Constant Configuration of Incremental ABN Encoder

The external position register *ENC_POS* 0x50 is based on internal microsteps. Thus, every AB transition is transferred to microsteps by a fixed constant value. TMC4330A is able to calculated this constant automatically.

In order to configure the incremental ABN encoder constant automatically, do as follows:

Action:

- Set fullstep resolution of the motor in *FS_PER_REV* (*STEP_CONF* register 0x0A).
- Set microstep resolution *MSTEP_PER_FS* (*STEP_CONF register* 0x0A).
- Set encoder resolution – the number of AB transitions during one revolution - in register *ENC_IN_RES* 0x54 (write access).

Result:
The encoder constant value *ENC_CONST* (readable at register 0x54) is calculated as follows:

$$ENC\_CONST = MSTEP\_PER\_FS \cdot FS\_PER\_REV / ENC\_IN\_RES$$

This constant is the number of microsteps through which *ENC_POS* is incremented or decremented by one AB transition.

i *ENC_CONST* consists of 15 digits and 16 decimal places.

i In case 16 bits are not sufficient for a binary representation of the decimal places, TMC4330A tries to match them to a multiple of 10000 within these 16 decimal places. Thereby, a perfect match can be achieved in case decimal representation is preferred to a binary one.

i In case the decimal representation also does not fit completely, the type of the decimal places of *ENC_CONST* can be selected manually with *ENC_IN_CONF* (0). Set *ENC_IN_CONF* (0) to 0 for binary representation; or set it to 1 for the decimal one. Keep in mind that with this approach *ENC_POS* can slightly differ from the real position; especially the further away the position moves from 0.

11.2.2. Manual Constant Configuration of Incremental ABN Encoder

For some applications it can be useful to define the encoder constant value, which in this case does not correspond to the number of microsteps per revolution; e.g. if the encoder is not mounted directly on the motor.

In order to configure the incremental ABN encoder constant manually, do as follows:

Action:

- Set *ENC_IN_RES*(31) =1.
- Set *ENC_IN_CONF*(0) to 0 for a binary or to 1 for a decimal representation as explained in the previous section.
- Set required encoder resolution in *ENC_IN_RES* (30:0) register 0x54.

Result:
ENC_CONST consists of 15 digits and 16 decimal places. The constant is the number of microsteps by which *ENC_POS* is incremented or decremented by one AB transition.

11.3. Incremental Encoders: Index Signal: N resp. Z

The index signal (N or Z channel) represents a recurrence of the same position in one motor encoder revolution. TMC4330A makes use of this signal to clear the external position counter, or to take a snapshot of the external or internal position, which then can be used to refine the home position more precisely.

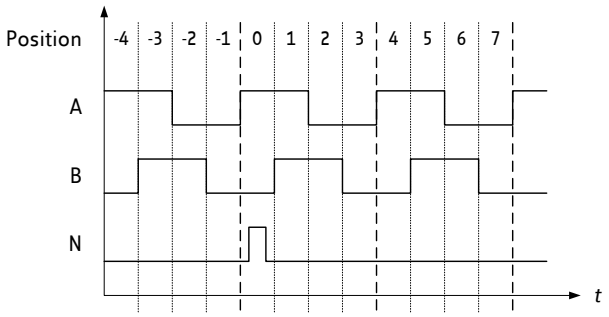


Figure 53: Outline of ABN Signals of an incremental Encoder

11.3.1. Setup of Active Polarity for Index Channel

Per default, the index channel is configured low active.

In order to set up high active polarity for the index channel, do as follows:

Action:
➢ Set *pol_n* =1 (register *ENC_CONF* 0x07).

Result:
The index channel is high active.

11.3.2. Configuration of N Event

The active polarity of the index channel can be used to clear the external position counter or to take a snapshot of the external or internal position. Therefore, N event is created internally. N event is based on the active polarity of the index channel. As addition, they can also be based on the polarities of the A and B channels.

Index Channel Sensitivity

Four active polarity configuration options for the index channel are available, which are presented below. Configuration choice depends on customer-specific design wishes.

In order to set up the index channel sensitivity based on active polarity, do as follows:

Action:
➢ Set *n_chan_sensitivity* (register *ENC_CONF* 0x07) to:

Index Channel Sensitivity	
n_chan_sensitivity	**Result**
b'00	N event is active in case index voltage level fits pol_n.
b'01	N event is triggered when the index channel switches to active polarity.
b'10	N event is triggered when the index channel switches to inactive polarity.
b'11	N event is triggered at both edges when the index channel switches to either active or inactive polarity.

Table 45: Index Channel Sensitivity

•→ *Description continued on next page.*

A and B Channel Signal Polarities for N Event

It can be useful to specify A and B channel signal polarities for N event. Per default, the polarities of both signal lines are set to 0 (low active).

In order to set up A channel polarity to high active for N event, do as follows:

Action:
- Set *pol_a_for_n* = 1 (*ENC_CONF* register 0x07).

Result:
Now, A channel signal polarity for N event is high active.

In order to set up B channel polarity to high active for N event, do as follows:

Action:
- Set *pol_b_for_n* = 1 (*ENC_CONF* register 0x07).

Result:
Now, B channel signal polarity for N event is high active.

In case A and B channel polarities do not have an influence on N event, both A and B channel polarity signals can be ignored.

In order to ignore A and B channel polarities, do as follows:

Action:
- Set *ignore_ab* = 1 (*ENC_CONF* register 0x07).

Result:
Now, the A and B channel signal polarities have no influence on N event.

11.3.3. External Position Counter *ENC_POS* Clearing

N event can be used to clear the external position register *ENC_POS* 0x50. Two choices are available: continous clearing and single clearing.

`i` Common practice is to clear to 0. However, TMC4330A offers the possibility to clear to any single microstep count.

ENC_POS Continous Clearing

In order to set *ENC_POS* on N event to continuous clearing, do as follows:

Action:
- Set *ENC_RESET_VAL* register 0x51 to the requested microstep position.
- Set *clr_latch_cont_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *clear_on_n* = 1 (*ENC_CONF* register 0x07).

Result:
On every N event *ENC_POS* is set to *ENC_RESET_VAL*.

ENC_POS Single Clearing

In order to only clear *ENC_POS* for the next N event, do as follows:

Action:
- Set *ENC_RESET_VAL* register 0x51 to the requested microstep position.
- Set *clr_latch_cont_on_n* = 0 (*ENC_CONF* register 0x07).
- Set *clr_latch_once_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *clear_on_n* = 1 (*ENC_CONF* register 0x07).

Result:
When the next N event occurs, *ENC_POS* is set to *ENC_RESET_VAL*. After the particular N event, *clr_latch_once_on_n* is automatically reset to 0.

Read entire documentation; especially the "Supplemental Directives" on page 160.

11.3.4. Latching External Position

N event can be used to latch external position register *ENC_POS* 0x50 to storage register *ENC_LATCH* 0x51 (read access). Two choices are available: Continous latching and single latching.

Continous Encoder Latching

In order to continuously latch *ENC_POS* to *ENC_LATCH* on N event, do as follows:

Action:

- Set *clr_latch_cont_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *latch_enc_on_n* = 1 (*ENC_CONF* register 0x07).

Result:
On every N event *ENC_POS* register 0x50 is latched to *ENC_LATCH* register 0x51.

Single Encoder Latching

In order to only latch *ENC_POS* to *ENC_LATCH* for the next N event, do as follows:

Action:

- Set *clr_latch_cont_on_n* = 0 (*ENC_CONF* register 0x07).
- Set *clr_latch_once_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *latch_enc_on_n* = 1 (*ENC_CONF* register 0x07).

Result:
When the next N event occurs, *ENC_POS* register 0x50 is latched to *ENC_LATCH* register 0x51. After the particular N event, *clr_latch_once_on_n* is automatically reset to 0.

11.3.5. Latching Internal Position

N event can be used to latch internal position register *X_ACTUAL* 0x21 to storage register *X_LATCH* 0x36 (read access). Two choices are available: Continous latching and single latching.

Continous Latching

In order to continuously latch *X_ACTUAL* to *X_LATCH* on N event, do as follows:

Action:

- Set *clr_latch_cont_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *latch_enc_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *latch_x_on_n* = 1 (*ENC_CONF* register 0x07).

Result:
On every N event *X_ACTUAL* register 0x21 is latched to *X_LATCH* register 0x36.

Single Latching

In order to only latch *X_ACTUAL* to *X_LATCH* for the next N event, do as follows:

Action:

- Set *clr_latch_cont_on_n* = 0 (*ENC_CONF* register 0x07).
- Set *clr_latch_once_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *latch_enc_on_n* = 1 (*ENC_CONF* register 0x07).
- Set *latch_x_on_n* = 1 (*ENC_CONF* register 0x07).

Result:
When the next N event occurs, *X_ACTUAL* register 0x21 is latched to *X_LATCH* register 0x36. After the particular N event, *clr_latch_once_on_n* is automatically reset to 0.

Read entire documentation; especially the "Supplemental Directives" on page 160.

11.4. Absolute Encoder Settings

Serial encoders provide absolute encoder angle data in contrast to step transitions, which are delivered from incremental encoders.
TMC4330A provides an external clock for the encoder in order to trigger serial data input,

11.4.1. Singleturn or Multiturn Data

TMC4330A offers singleturn and multiturn options for the serial data stream interpretation. Per default, multiturn data is not enabled. In case multiturn data is enabled, it is interpreted as unsigned count of revolutions.

In case multiturn encoder data is transmitted, do as follows:

Action:
- Set *multi_turn_in_en* = 1 (*ENC_CONF* register 0x07).
- ***OPTIONAL CONFIGURATION:*** Set *multi_turn_in_signed* = 1.
  In case multiturn data is provided as signed count of encoder revolutions.

Result:
Data from connected encoders are interpreted as multiturn data.

In case only singleturn data is transmitted TMC4330A is able to permanently calculate internally the number of encoder revolutions as if it where externally transferred multiturn data.

In case singleturn encoder data is transmitted but internally multiturn data is required, do as follows:

Action:
- Set *multi_turn_in_en* = 0 (*ENC_CONF* register 0x07).
- Set *calc_multi_turn_behav* = 1 (*ENC_CONF* register 0x07).

Result:
Data from connected singleturn encoders is internally transferred to multiturn data.

NOTE:

→ *Multiturn calculations are only correct in case two consecutive singleturn data values differ only by one step less than a half turn difference, or even less.*

11.4.2. Automatic Constant Configuration of Absolute Encoder

The external position register *ENC_POS* 0x50 is based on internal microsteps. Thus, every input data angle is transferred to microsteps by a fixed constant value. TMC4330A is able to automatically calculate this constant.

In order to configure the absolute encoder constant automatically, do as follows:

Action:

- Set fullstep resolution of the motor in *FS_PER_REV* (*STEP_CONF* register 0x0A).
- Set microstep resolution *MSTEP_PER_FS* (*STEP_CONF* register 0x0A).
- Set encoder resolution in register *ENC_IN_RES* 0x54 (write access).

Result:

The encoder constant value *ENC_CONST* (readable at register 0x54) is calculated as follows:

$$ENC\_CONST = MSTEP\_PER\_FS \cdot FS\_PER\_REV / ENC\_IN\_RES$$

The external position *ENC_POS* 0x50 is calculated by multiplying the constant with the transmitted input angle.

i *ENC_CONST* consists of 15 digits and 16 decimal places.

i In contrast to incremental ABN encoders, *ENC_CONST* is always represented as binary constant.

11.4.3. Manual Constant Configuration of incremental ABN Encoder

For some applications it can be useful to define the encoder constant value, which in this case does not correspond to the number of microsteps per revolution; e.g. if the encoder is not mounted directly on the motor.

In order to configure the absolute encoder constant manually, do as follows:

Action:

- Set *ENC_IN_RES* (31) =1.
- Set required encoder resolution in *ENC_IN_RES* (30:0) register 0x54.

Result:

ENC_CONST consists of 15 digits and 16 decimal places. The external position *ENC_POS* 0x50 is calculated by multiplying the constant with the transmitted input angle.

Read entire documentation; especially the "Supplemental Directives" on page 160.

11.4.4. Absolute Encoder Data Setup

Encoder Data must be maintained correctly. Consequently, certain settings must be configured so that TMC4330A displays them as specified.

In order to configure absolute encoder data, do as follows:

Action:

- Set *SINGLE_TURN_RES* (*ENC_IN_DATA* register 0x08) to the number of singleturn data bits -1.

OPTION A1: IF MULTITURN DATA IS TRANSMITTED

- Set *MULTI_TURN_RES* (*ENC_IN_DATA* register 0x08) to the number of multiturn data bits -1.

OR OPTION A2: IF MULTITURN DATA IS NOT TRANSMITTED

- Set *MULTI_TURN_RES* = 0 (*ENC_IN_DATA* register 0x08).

- Set *STATUS_BIT_CNT* (also register 0x08) to the number of status bits.

OPTION B1: IF STATUS FLAGS ARE ORDERED IN FRONT

- Set *left_aligned_data* = 0 (*ENC_IN_CONF* register 0x07).

OR OPTION B2: IF STATUS FLAGS ARE ORDERED IN FRONT

- Set *left_aligned_data* = 1 (*ENC_IN_CONF* register 0x07).

Result:

SINGLE_TURN_RES defines the most significant bit (MSB) of the angle data bits, whereas *MULTI_TURN_RES* defines the MSB of the revolution counter bits. Up to three status bits can be received. The number of transferred clock bits that are sent to the encoder is calculated as follows:

#SCLK Cycles= (*SINGLE_TURN_RES*+1) + (*MULTI_TURN_RES*+1) + *STATUS_BIT_CNT*

Also, the order in which the status bits occur in one encoder data stream can be configured. In Figure 54, example setups are depicted.

NOTE:

→ *In case more than three status bits or additional fill bits are sent from the encoder, clock errors can occur because the number of transferred clock bits does not fit.*

→ *In order to prevent clock failures, MULTI_TURN_RES can be set to a higher value than otherwise required; even if the encoder does not provide multiturn data. This can result in erroneous multiturn data, which can be corrected by setting multi_turn_in_en=0 in order to skip multiturn data automatically.*

→ *In order to compensate unavailable multiturn data make use of calc_multi_turn_behav, as explained in section 15.4.1 on page 99.*

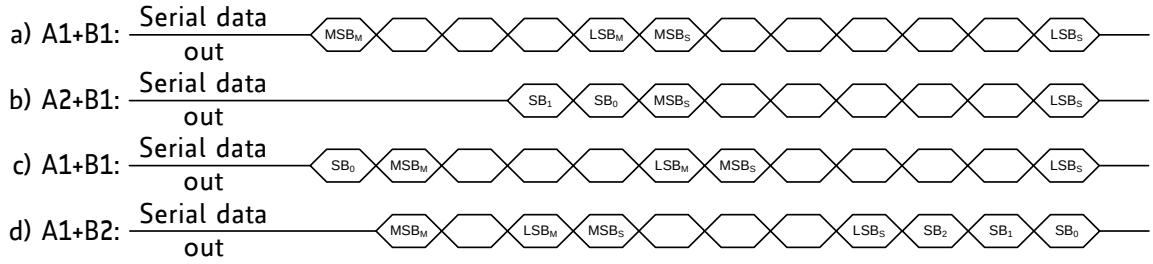


Figure 54:Serial Data Output: Four Examples

Key:

a) SINGLE_TURN_RES=6; MULTI_TURN_RES=4; STATUS_BIT_CNT=0; left_aligned_data=0

b) SINGLE_TURN_RES=6; MULTI_TURN_RES=0; STATUS_BIT_CNT=2; left_aligned_data=0

c) SINGLE_TURN_RES=5; MULTI_TURN_RES=4; STATUS_BIT_CNT=1; left_aligned_data=0

d) SINGLE_TURN_RES=4; MULTI_TURN_RES=2; STATUS_BIT_CNT=3; left_aligned_data=1

11.4.5. Emitting Encoder Data Variation

For some applications it can be useful to limit the difference between two consecutive encoder data values; for instance, if encoder data lines are subject to too much noise. Per default, encoder data values can show a difference of 1/8th per encoder revolution, only if the limitation is enabled. The difference can be configured to a smaller value, if necessary.

In order to enable and configure encoder data variation limitation, do as follows:

Action:

- ***OPTIONAL:*** Set proper *SER_ENC_VARIATION* register 0x63 (7:0).
- Set *serial_enc_variation_limit* =1 (*ENC_IN_CONF* register 0x07).

Result:

The encoder data value that is received subsequently must not exceed the previous data more than:

Maximum tolerated deviation = *SER_ENC_VARIATION* / 256 · 1/8 · *ENC_IN_RES.*

In case the variation exceeds the above mentioned limit, the new data value is rejected internally and the status flag *SER_ENC_DATA_FAIL* is raised.

i In case *SER_ENC_VARIATION* = 0, the limit is defined by 1/8 · *ENC_IN_RES.*

11.4.6. SSI Clock Generation

In order to receive encoder data from the absolute encoder, TMC4330A generates clock patterns according to SSI standard. Data transfer is initiated by switching the clock line SCLK from high to low level. The transfer starts with the next rising edge of SCLK. The number of emitted clock cycles depends on the expected data width, as explained in section 15.4.4.

Configuration Details

One clock cycle has a high and a low phase, which can be defined separately according to internal clock cycles. Per default, sample points of serial data are set at the falling edges of SCLK. Some encoders need more clock cycles – than are available during the low clock phase – in order to prepare data for transfer. Also, due to long wires, data transfer can take more time. To counteract the above mentioned issues, the delay time *SSI_IN_CLK_DELAY* (default value equals 0) for compensation can be specified in order to prolong the sampling start. Therefore, this delay configuration can automatically generate more clock cycles.
After a data request – when all clock cycles have been emitted – the serial clock must remain idle for a certain interval before the next request is automatically initiated. This interval *SER_PTIME* can also be configured in internal clock cycles.

`i` According to SSI standard, select an interval that is longer than 21 µs.

In order to configure the SSI clock generation, do as follows:

Action:

- Set *SINGLE_TURN_RES* (*ENC_IN_DATA* register 0x08) to the number of singleturn data bits -1.
- Set *MULTI_TURN_RES* (*ENC_IN_DATA* register 0x08) to the number of multiturn data bits -1 in case multiturn data is enabled and used.
- Set *STATUS_BIT_CNT* (*ENC_IN_DATA* reg. 0x08) to the number of status bits.
- Set proper *left_aligned_data* (*ENC_IN_CONF* register 0x07).
- Set proper *SER_CLK_IN_LOW* (register 0x56) in internal clock cycles.
- Set proper *SER_CLK_IN_HIGH* (register 0x56) in internal clock cycles.
- ***OPTIONAL CONFIG:*** Set proper *SSI_IN_CLK_DELAY* (register 0x57) in internal clock cycles.
- ***OPTIONAL CONFIG:*** Set proper *SER_PTIME* (reg. 0x58) in internal clk cycles.
- **Finally, set *serial_enc_in_mode* = b'01**.

Result:

TMC4330A emits serial clock streams at SCLK in order to receive absolute encoder data at SDI. If *SSI_IN_CLK_DELAY* > 0, the SDI sample points are delayed (see figures below). *SER_PTIME* defines the interval between two consecutive data requests.

`i` If differential encoder is selected, the negated clock emits at ¬SCLK; and ¬SDI is also evaluated.

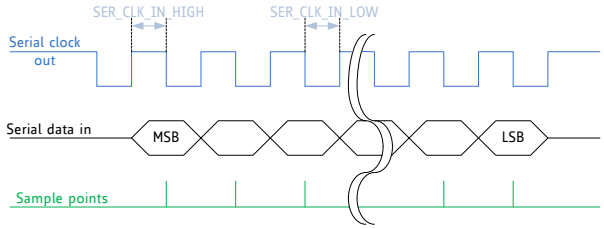


Figure 55: SSI: SSI_IN_CLK_DELAY=0

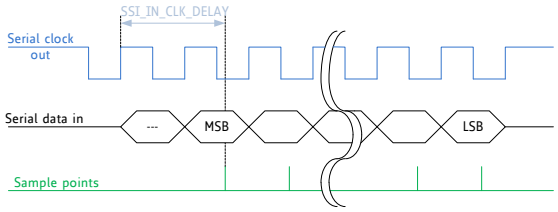


Figure 56: SSI: SSI_IN_CLK_DELAY>SER_CLK_IN_HIGH

11.4.7. Enabling Multicycle SSI request

If safe transmission must be determined, it is possible to send a second request so that the encoder repeats the same encoder data. Therefore, a second interval *SSI_WTIME* must be defined.

`i` According to SSI standard, select an interval that is shorter than 19 µs.

In order to enable multicycle requests, do as follows:

Action:
- Set *ssi_multi_cycle_data* =1 (*ENC_IN_CONF* register 0x07).
- Set proper *SSI_WTIME* (register 0x57) in internal clk cycles.

Result:
After a data request – when all clock cycles have been emitted – the serial clock remains idle for *SSI_WTIME* clock cycles. Afterwards, the second request is automatically initiated to receive the same encoder data. If the second encoder data differs from the first one, error flag *MULTI_CYCLE_FAIL* (register 0x0F) and error event *SER_ENC_DATA_FAIL* (register 0x0E) is generated.
After the second data request, the next interval lasts *SER_PTIME* clock cycles to request new encoder data.

11.4.8. Gray-encoded SSI Data Streams

Several but not all SSI encoders emit angle data, which is gray-encoded. TMC4330A is able to decode this data automatically.

In order to enable gray-encoded angle data, do as follows:

Action:
- Set *ssi_gray_code_en* =1 (*ENC_IN_CONF* register 0x07).

Result:
Encoder data is recognized as gray-encoded and thus also decoded accordingly.

11.4.9. SPI Encoder Data Evaluation

SPI encoder interfaces typically consist of four signal lines. In addition to SSI encoder signal lines (SCLK, MISO), a chip select line (CS) and a data input (MOSI) to the master is provided.

SPI Encoder Communication Process

The number of bits per transfer is calculated automatically; based on proper *multi_turn_in_en*, *SINGLE_TURN_RES*, *MULTI_TURN_RES*, and *STATUS_BIT_CNT*, as explained in sections 15.4.1 (page 99) and 15.4.4 (page 101).

A typical SPI communication process responds to any SPI data transfer request when the next transmission occurs. When TMC4330A receives an answer from the encoder, it calculates *ENC_POS* immediately. The encoder slave does not send any data without receiving a request first.
Therefore, TMC4330A always sends *ADDR_TO_ENC* value to request encoder data from the SPI encoder slave device. The LSB of the serial data output is *ADDR_TO_ENC* (0).
Received encoder data is stored in *ADDR_FROM_ENC*. Thus, encoder values can be verified and compared to microcontroller data later on.

i The clock generation works similarly to SSI clock generation, as described in section 15.4.5 on page 103; based on proper *SER_CLK_IN_HIGH, SER_PTIME,* and *SER_CLK_IN_LOW*.

In order to configure a basic SPI communication procedure, do as follows:

Action:

- Set *SINGLE_TURN_RES* (*ENC_IN_DATA* register 0x08) to the number of singleturn data bits -1.
- Set *MULTI_TURN_RES* (*ENC_IN_DATA* register 0x08) to the number of multiturn data bits -1 in case multiturn data is enabled and used.
- Set *STATUS_BIT_CNT* (*ENC_IN_DATA* register 0x08) to the number of status bits.
- Set proper *left_aligned_data* (*ENC_IN_CONF* register 0x07).
- **Set correct SPI transfer mode that is described in the next section**.
- Set *ADDR_TO_ENC* register 0x68 to the specified SPI encoder address that contains angle data.
- Set proper *SER_CLK_IN_LOW* (register 0x56) in internal clock cycles.
- Set proper *SER_CLK_IN_HIGH* (register 0x56) in internal clock cycles.
- ***OPTIONAL CONFIG:*** Set proper *SER_PTIME* (register 0x58) in internal clk cycles.
- **Finally, set *serial_enc_in_mode* = b'11**.

Result:
TMC4330A emits serial clock streams at SCLK in order to receive absolute encoder data at SDI pin. The number of generated clock cycles depends on *SINGLE_TURN_RES*, *MULTI_TURN_RES*, and *STATUS_BIT_CNT*.
Pin ANEG_NSCLK functions as negated chip select line for the SPI encoder that is generated according to the serial clock and the selected SPI mode; which is described in the next section.
Pin BNEG_NSDI is the MOSI line that transfers SPI datagrams to the SPI encoder. Datagrams, which are transferred permanently to receive angle data, consists of *ADDR_TO_ENC* data.
SER_PTIME defines the interval between two consecutive data requests.

•→Turn page for information on SPI mode selection.

11.4.10. SPI Encoder Mode Selection

Per default, SPI encoder data transfer is managed in the same way as the communication between microcontroller and TMC4330A. TMC4330A supports all four SPI modes with proper setting of switches *spi_low_before_cs* and *spi_data_on_cs*.

THE PROCESS IS AS FOLLOWS:

By setting ***spi_low_before_cs* = 0**, negated chip select line at ANEG_NSCLK is switched to active low **before** the serial clock line SCLK switches.

By setting ***spi_low_before_cs* = 1**, negated chip select line at ANEG_NSCLK is switched to active low **after** the serial clock line SCLK switches.

By setting ***spi_data_on_cs* = 0**, the first data bit at BNEG_NSDI is changed at the same time as the first slope of the serial clock SCLK.

By setting ***spi_data_on_cs* = 1**, the first data bit at BNEG_NSDI is changed at the same time as the negated chip select signal at BNEG_NSDI switches to active level.

In the table below, all four SPI modes are presented.

Per default, the delay between serial clock line and negated chip select line has a time frame of either *SER_CLK_IN_HIGH* or *SER_CLK_IN_LOW* clock cycles, which depends on the actual voltage level of the serial clock.
This particular interval does not always match the encoder behavior perfectly. Therefore, both the first and last intervals between the serial clock line and the negated chip select line can be specified separately in clock cycles at *SSI_IN_CLK_DELAY* register 0x57.

Below, the *SSI_IN_CLK_DELAY* interval is highlighted in red in all four diagrams.

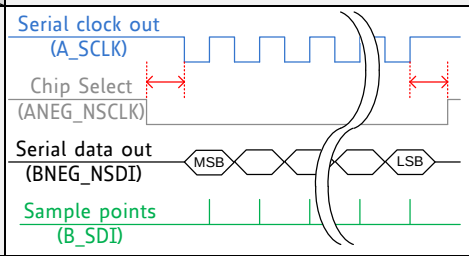
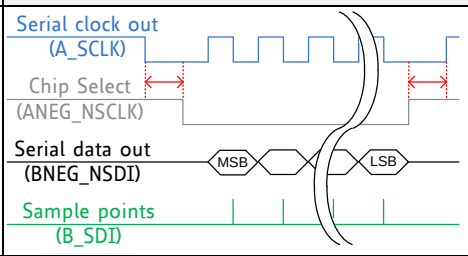
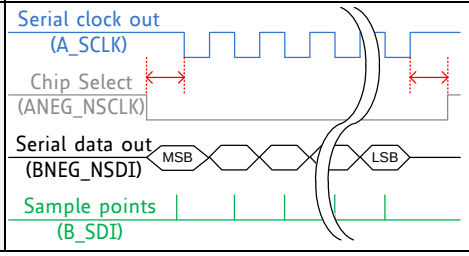
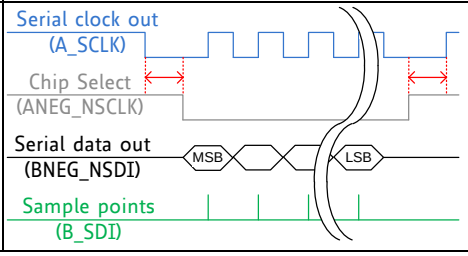
Supported SPI Encoder Data Transfer Modes		
spi_low_before_cs: / spi_data_on_cs	**0**	**1**
0		
1		

Table 46: Supported SPI Encoder Data Transfer Modes

Read entire documentation; especially the "Supplemental Directives" on page 160.

11.4.11. SPI Encoder Configuration via TMC4330A

Connected SPI encoder can be configured via TMC4330A., which renders a connection between microcontroller and encoder unnecessary.

SPI Encoder Configuration Communication Process

A configuration request is sent using the settings of *SERIAL_ADDR_BITS* and *SERIAL_DATA_BITS,* which define the transferring bit numbers.

In order to prepare SPI encoder configuration procedures, do as follows:

Action:

- Set *SERIAL_ADDR_BITS* (*ENC_IN_DATA* register 0x08) to the number of address bits of any SPI encoder configuration datagram.
- Set *SERIAL_DATA_BITS* (*ENC_IN_DATA* register 0x08) to the number of data bits of any SPI encoder configuration datagram.

Result:
In case configuration data is transferred to the SPI encoder, *SERIAL_ADDR_BITS* bits and *SERIAL_DATA_BITS* bits are sent in two SPI configuration datagrams; exactly in this order.

Because encoder data requests occur as an endless stream, it is necessary to interrupt data requests when a configuration request occurs. Consequently, a handshake behavior is implemented.

In order to transfer configuration data to the SPI encoder, do as follows:

Action:

- Set *DATA_TO_ENC* register 0x69 to any value.
- Set *ADDR_TO_ENC* register 0x68 to the configuration address of the SPI encoder.
- Set *DATA_TO_ENC* register 0x69 to the configuration data of the SPI encoder.

Result:
The first *DATA_TO_ENC* access stops the repetitive encoder data request.
After the second *DATA_TO_ENC* access, three datagrams are sent to SPI encoder:

1. One address datagram is transmitted, which contains the *ADDR_TO_ENC* value. Data that is received simultaneously with the request is not stored.
2. One data datagram is transmitted that contains the *DATA_TO_ENC* value. Data that is received simultaneously with the request is stored in *ADDR_FROM_ENC* register 0x6A because this is the response of the *ADDR_TO_ENC* request.
3. One no-operation datagram (NOP) is transmitted. Data that is received simultaneously with the request is stored in *DATA_FROM_ENC* register 0x6B because this is the response of the *DATA_TO_ENC* request.

In order to finalize the configuration procedure and continue with the encoder data requests, do as follows:

- Read out *ADDR_FROM_ENC* register 0x6A first.
- Set *ADDR_TO_ENC* register 0x68 to the specified SPI encoder address that contains angle data.
- **Obligatory at finalization: Read out *DATA_FROM_ENC* register 0x6B.**

Result:
The configuration request data is read out. After *DATA_FROM_ENC* register readout, the encoder data request stream of angle data continues.

12. Possible Regulation Options with Encoder Feedback

Beyond simple feedback monitoring, encoder feedback can be used for controlling motion controller outputs in such a way that the internal actual position matches or follows the real position *ENC_POS*. Two options are provided: PID control and closed-loop operation. Closed-loop operation is preferable if the encoder is mounted directly on the back of the motor and position data is evaluated precisely. PID control is preferable if the encoder is located on the drive side with no fixed connection between motor and drive side; e.g. belt drives.

Closed-Loop and PID Registers			
Register Name	**Register address**		**Remarks**
ENC_IN_CONF	0x07	RW	Encoder configuration register: Closed-Loop configuration switches.
CL_TR_TOLERANCE	0x51	R	Absolute tolerated deviation to trigger TARGET_REACHED during regulation.
ENC_POS_DEV	0x52	R	Deviation between *XACTUAL* and *ENC_POS*.
Closed-Loop and PID Register Set	0x59…5F 0x60…61	W	Closed-Loop and PID configuration parameters.
Encoder velocity configuration	0x63	W	Encoder velocity filter configuration parameters.
Encoder velocity	0x65 0x66	R	Current encoder velocity (signed). Current filtered encoder velocity (signed).

Table 47: Dedicated Closed-Loop and PID Registers

12.1. Feedback Monitoring

Based on the difference *ENC_POS_DEV* (readout at register 0x52) between internal position *XACTUAL* and external position *ENC_POS*, a status flag *ENC_FAIL_F* and a corresponding error event *ENC_FAIL* is generated automatically.

In order to set a tolerated position mismatch, do as follows:

Action:

- Set *ENC_POS_DEV_TOL* register 0x53 to the maximum microstep value that represents no mismatch failure.

Result:

In case $|ENC\_POS\_DEV| \leq ENC\_POS\_DEV\_TOL$, no encoder failure flag is set.
In case $|ENC\_POS\_DEV| > ENC\_POS\_DEV\_TOL$, *ENC_FAIL_F* flag is set.

i At this point, the corresponding encoder event *ENC_FAIL* is also triggered.

12.1.1. Target-Reached during Regulation

In case one of the regulation modes is selected, TARGET_REACHED event and status flag is only released when:

$$XACTUAL = XTARGET \quad \text{and} \quad |ENC\_POS\_DEV| \leq CL\_TR\_TOLERANCE.$$

Consequently, *CL_TR_TOLERANCE* register 0x52 (only write access) is the maximal tolerated position mismatch for target reached status.

Read entire documentation; especially the "Supplemental Directives" on page 160.

12.2. PID-based Control of *XACTUAL*

Based on a position difference error *PID_E* = *XACTUAL* – *ENC_POS* the PID (proportional integral differential) controller calculates a signed velocity value (v_{PID}), which is used for minimizing the position error. During this process, TMC4330A moves with v_{PID} until |*PID_E*| – *PID_TOLERANCE* ≤ 0 is reached and the position error is removed.

v_{PID} is calculated by:

$$v_{PID} = \frac{PID\_P}{256} \cdot PID\_E \cdot \left[\frac{1}{s}\right] + \frac{PID\_I}{256} \cdot \int_0^t PID\_E \cdot dt \; + \; PID\_D \cdot PID\_E \cdot \frac{d}{dt}$$

$$v_{PID} = \frac{PID\_P}{256} \cdot PID\_E \cdot \left[\frac{1}{s}\right] + \frac{PID\_I}{256} \cdot PID\_ISUM + PID\_D \cdot PID\_E \cdot \frac{d}{dt}$$

$$v_{PID} = \frac{PID\_P}{256} \cdot PID\_E \cdot \left[\frac{1}{s}\right] + \frac{PID\_I}{256} \cdot PID\_E \cdot \frac{f_{CLK}}{128} + \; PID\_D \cdot PID\_E \cdot \frac{d}{dt}$$

Key:

PID_P = proportional term; PID_I = integral term; PID_D = derivate term

12.2.1. PID Readout Parameters

The following parameters can be read out during PID operation.

PID_VEL 0x5A Actual PID output velocity.

PID_E 0x5D Actual PID position deviation between *XACTUAL* and *ENC_POS*.

PID_ISUM 0x5B Actual PID integrator sum (update frequency: $f_{CLK}/128$), which is calculated by:

$$PID\_ISUM = PID\_E \cdot f_{CLK} / 128$$

•→Turn page for information on configuration of PID regulation.

12.2.2. PID Control Parameters and Clipping Values

In order to set parameters and clipping values for PID regulation correctly, consider the following details:

PID_DV_CLIP 0x5E

Large velocity variations are avoided by limiting v_{PID} value with *PID_DV_CLIP* (register 0x5E). This clipping parameter limits both v_{PID} and *PID_VEL*.

PID_I_CLIP 0x5D (14:0)

The error sum *PID_ISUM* (read out at 0x5B) is generated by the integral term. *PID_ISUM* is limited by setting *PID_I_CLIP* register 0x5D.

- `i` The maximum value of *PID_I_CLIP* must meet the condition $PID\_I\_CLIP \leq PID\_DV\_CLIP / PID\_I$.
- `i` If the error sum *PID_ISUM* is not clipped, it is increased with each time step by $PID\_I \cdot PID\_E$. This continues as long as the motor does not follow.

PID_D_CLKDIV 0x5D (23:16)

Time scaling for deviation (with respect to error correction periods) is controlled by *PID_D_CLKDIV* register.

- `i` During error correction, fixed clock frequency $f_{PID\_INTEGRAL}$ is valid:

  $f_{PID\_INTEGRAL}[Hz] = f_{CLK}[Hz] / 128$

VEL_ACT_PID

The internal velocity VEL_ACT_PID alters actual ramp velocity *VACTUAL*. Two settings are provided:
In case *regulation_modus* = b'11, *VACTUAL* is assigned as pulse generator base value and VEL_ACT_PID is calculated by VEL_ACT_PID = *VACTUAL* + v_{PID}.
In case *regulation_modus* = b'10, zero is assigned as pulse generator base value. Now, VEL_ACT_PID = v_{PID} is valid.

PID_TOLERANCE 0x5F

TMC4330A provides the programmable hysteresis *PID_TOLERANCE* for target position stabilization; which avoids oscillations through error correction in case *XACTUAL* is close to the real mechanical position.

The PID controller of TMC4330A is programmable up to approximate 100 kHz update rate (at f_{CLK} = 16 MHz). This high speed update rate qualifies PID regulation for motion stabilization.

12.2.3. Enabling PID Regulation

Now that PID control parameters and clipping values are configured, as explained above, PID regulation can be enabled. Two options can be selected.

In order to enable PID control, do as follows:

Action:

OPTION 1: BASE PULSE GENERATOR VELOCITY = 0

➢ Set *regulation_modus* = b'10 (*ENC_IN_CONF* register 0x07).

OPTION 2: BASE PULSE GENERATOR VELOCITY = VACTUAL

➢ Set *regulation_modus* = b'11 (*ENC_IN_CONF* register 0x07).

Result:
PID regulation is enabled.

NOTE

→ *Detailed knowledge of a particular application (including dynamics of mechanics) is necessary for PID controller parameterization.*

Read entire documentation; especially the "Supplemental Directives" on page 160.

12.3. Closed-Loop Operation

The closed-loop unit of TMC4330A directly modifies Step/Dir outputs of the internal step generator; which is dependent on the feedback data. The 2-phase closed-loop control of TMC4330A follows a different approach than Field-Oriented Control (FOC); which is similar to PID control cascades. The ramp generator, which assigns target and velocity, is independent of position control (commutation angle control); which is also independent of S/D output control.

12.3.1. Basic Closed-Loop Parameters

Closed-loop does not control current values via the internal step generator. The currents values at the SPI output and the Step/Dir outputs are verified using the evaluated difference between internal position *XACTUAL* and external position *ENC_POS*; considering the calibrated offset parameter *CL_OFFSET*.

In order to set parameters and clipping values for closed-loop regulation correctly, consider the following details:

CL_OFFSET 0x59

This register contains the basic offset value between internal and external position during calibration process, which is necessary for closed-loop operation, and offers read-write access. The write access can be used if a defined fixed offset value is preferred, which is verified beforehand.

ENC_POS_DEV 0x52

The continuously updated parameter *ENC_POS_DEV* displays the deviation between *XACTUAL* and *ENC_POS*; considering *CL_OFFSET*.

CL_BETA 0x1C (8:0)

CL_BETA is the maximum commutation angle that is used to compensate an evaluated deviation *ENC_POS_DEV*. In case the deviation reaches *CL_BETA* value, the commutation angle remains stable at this value to follow the overload. Also, *CL_MAX* event is triggered at this point.

CL_TOLERANCE 0x5F (7:0)

This parameter is set to select the tolerance range for position deviation. In case $|ENC\_POS\_DEV| \leq CL\_TOLERANCE$, *CL_FIT_F* lag becomes set.
In case a mismatch between internal and external position occurs, *CL_FIT* event is triggered to signify when the mismatch is removed.

CL_DELTA_P 0x5C

CL_DELTA_P is a proportional controller that compensates a detected position deviation between internal and external position. See also Figure 57, page 112.
In case $|ENC\_POS\_DEV| \leq CL\_TOLERANCE$, *CL_DELTA_P* is automatically set to 1.0.
In case $|ENC\_POS\_DEV| > CL\_TOLERANCE$, the closed-loop unit of TMC4330A multiplies *ENC_POS_DEV* with *CL_DELTA_P* and adds the resulting value to the current *ENC_POS*. Thus, a current commutation angle for higher stiffness position maintenance, which is clipped at *CL_BETA*, is calculated.

i *CL_DELTA_P* consists of 24 bits. The last 16 bits represent decimal places. The final proportional term is thus calculated by: $p_{PID} = CL\_DELTA\_P / 65536$.

i Therefore, the higher p_{PID} the faster the reaction on position deviations.

NOTE:

→ *A high p_{PID} term can lead to oscillations that must be avoided.*

CL_CYCLE 0x63 (31:16)

In case, one absolute encoder is connected, this value represents the delay time in numbers of clock cycles between two consecutive regulation cycles. It is recommended to adjust this value to the regulation cycle; which is either equal or slower than the encoder request rate. In case incremental ABN encoder is selected, this value is automatically set to fetch the fastest possible regulation rate; which in most cases are five clock cycles.

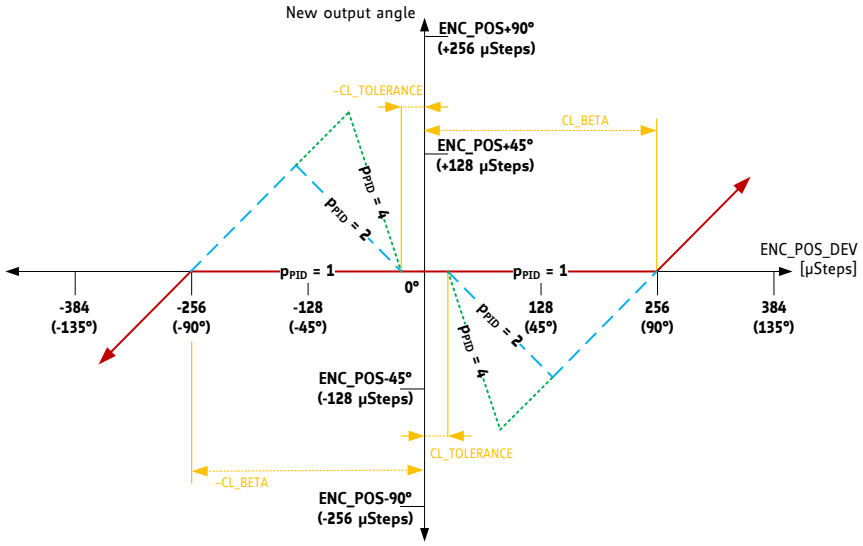


Figure 57: Calculation of the Output Angle with appropriate CL_DELTA_P

12.3.2. Enabling and calibrating Closed-Loop Operation

Now that basic closed-loop control parameters are configured, as explained above, closed-loop regulation can be enabled.

i The presented calibration process is very basic. Refer to the closed-loop Application Note for detailed calibration process information.

In order to enable and calibrate closed-loop control, do as follows:

PRECONDITION: SET TO BEST POSSIBLE MAXIMUM CURRENT SCALING

PROCEED WITH: OPTION 1: CL_OFFSET IS GENERATED DURING CALIBRATION

Action:

- Set *MSTEPS_PER_FS* = 0 (*STEP_CONF* register 0x0A) [256 microsteps per fullstep].
- Move to any fullstep position (*MSCNT* mod 128 = 0).
- Set *regulation_modus* = b'01 (*ENC_IN_CONF* register 0x07).
- Set *cl_caclibration_en* =1 (*ENC_IN_CONF* register 0x07).
- Wait for a defined time span (system settle down).
- Set *cl_caclibration_en* =0 (*ENC_IN_CONF* register 0x07).

Result:
Closed-loop operation is enabled with basic calibration. *CL_OFFSET* is set to position mismatch during calibration process.

OR PROCEED WITH OPTION 2: CL_OFFSET IS USED FOR CALIBRATION

In case *CL_OFFSET* was saved and no position loss has occurred while closed-loop operation was disabled, it can be used to replace the calibration process.

Action:

- Set *MSTEPS_PER_FS* = 0 (*STEP_CONF* register 0x0A) ➔ 256 microsteps per fullstep.
- Set *regulation_modus* = b'01 (*ENC_IN_CONF* register 0x07).
- Set *CL_OFFSET* to any preferred microstep value.

Result:
Closed-loop operation is enabled.

12.3.3. Limiting Closed-Loop Catch-Up Velocity

In order to limit catch-up velocities in case a disturbance of regular motor motion must be compensated, the following parameters can be configured accordingly:

i Refer to section 16.2. on page 109 for more information about PI regulation of the maximum velocity because it uses the same PI regulator like the position PID regulator. The base velocity is the actual ramp velocity *VACTUAL*.

CL_VMAX_CALC_P 0x5A P parameter of the PI regulator, which controls the maximum velocity.

CL_VMAX_CALC_I 0x5B I parameter of the PI regulator, which controls the maximum velocity.

PID_DV_CLIP 0x5E *PID_DV_CLIP can* be set in order to avoid large velocity variations; and also to limit the maximum velocity deviation above the maximum velocity *VMAX*.

PID_I_CLIP 0x5D This parameter is used together with *PID_DV_CLIP* in order to limit the velocity for error compensation. The error sum *PID_ISUM* is generated by the integral term. In case this error sum must be limited, set *PID_I_CLIP*.
It is advisable to set the maximum value of *PID_I_CLIP* to:

$$PID\_I\_CLIP \leq PID\_DV\_CLIP \mid PID\_I.$$

i In case the error sum *PID_ISUM* is not clipped, it is increased with each time step by $PID\_I \cdot PID\_E$. This continues as long as the motor does not follow.

12.3.4. Enabling the Limitation of the Catch-Up Velocity

Now that PI control parameters and clipping values are configured, as explained above, limiting catch-up velocities can be enabled.

In order to enable limitation of closed-loop catch-up velocity, do as follows:

Action:
➢ Set *cl_vlimit_en* = 1 (*ENC_IN_CONF* register 0x07).

Result:
Closed-loop catch-up velocity is limited according to the configured parameters.

NOTE:

→ *A higher motor velocity than specified VMAX (for negative velocity: -VMAX) is possible if the following conditions are met:*
- *Closed-loop operation is enabled.*
- *Closed-loop catch-up velocity is not enabled, or is enabled with PID_DV_CLIP > 0; and CL_VMAX_CALC_P and CL_VMAX_CALC_I are higher than 0.*
- *ENC_POS_DEV > CL_TOLERANCE resp. ENC_POS_DEV < CL_TOLERANCE.*

AREAS OF SPECIAL CONCERN

In case the internal ramp has stopped, and the position mismatch still needs to be corrected, the base velocity for catch-up velocity limitation is zero.

The mismatch correction ramp is a linear deceleration ramp, independent of the specified ramp profile. This occurs because the catch-up velocity is regulated via PI regulation, as explained above.
Thus, this final ramp for error compensation is a function of both *ENC_POS_DEV* and the PI control parameters.

•→*Turn page for information on closed-loop velocity mode.*

12.3.5. Enabling Closed-Loop Velocity Mode

Some applications only require maintaining a specified velocity value during closed-loop behavior, regardless of position mismatches. TMC4330A also provides this option.

NOTE:

→ *The closed-loop velocity mode is set independent of the internal ramp operation mode (velocity or positioning mode).*

In order to enable and calibrate closed-loop control, do as follows:

Action:

- Set the catch-up velocity parameters, as explained in detail in section 16.3.3, page 113.
- Set *cl_vlimit_en* = 1 (*ENC_IN_CONF* register 0x07).
- Set *cl_velocity_mode_en* = 1 (*ENC_IN_CONF* register 0x07).

Result:

Closed-loop operation velocity mode is enabled.
In case position mismatch |*ENC_POS_DEV*| exceeds 768 microsteps, internal position counter *XACTUAL* is set automatically to *ENC_POS* ± 768 to limit the position mismatch.
Thus, closed-loop operation maintains the specified velocity value *VMAX*.

`i` A higher motor velocity than specified *VMAX* (for negative velocity: -*VMAX*) is possible if *PID_DV_CLIP* > 0.

12.3.6. Back-EMF Compensation during Closed-loop Operation

When higher velocities are reached, a phase shift between current and voltage occurs at the motor coils. Consequently, current control is transformed into voltage control.
This motor- and setup-dependent effect must be compensated because currents are still continuously assigned for motor control. TMC4330A attributes γ-correction to the compensation process, which adds a velocity-dependent angle - in motion direction - to the current commutation angle.

Load Angle Calculation

Gamma correction constantly adds one compensation angle, GAMMA, to the actual commutation angle; because the velocity-dependent amount of the influence of Back-EMF, GAMMA is also velocity-dependent. Thus, velocity limits are assigned. These limits are based on REAL motor velocity *V_ENC* (register 0x65). The value of the motor velocity is internally calculated and can be filtered (*V_ENC_MEAN* register 0x66) to smoothen the γ-correction, which is explained in the next section.

In order to configure and enable Back-EMF compensation during closed-loop operation, do as follows:

Action:

- Set proper *CL_GAMMA* register 0x1C.
- Set proper *CL_VMIN_EMF* register 0x60.
- Set proper *CL_VMAX_EMF* register 0x61.
- Set *cl_emf_en* = 1 (*ENC_IN_CONF* register 0x07).

Result:
Back-EMF compensation during closed-loop operation is enabled. *CL_GAMMA* represents the maximum value of GAMMA. Per default, *CL_GAMMA* is set to its maximal possible value of 255, which represents a 90° angle.
The following compensation situations are possible:

1. In case $|V\_ENC\_MEAN| \leq CL\_VMIN\_EMF$, GAMMA is set to 0.
2. In case $|V\_ENC\_MEAN| > CL\_VMIN\_EMF$ and $|V\_ENC\_MEAN| \leq (CL\_VMIN\_EMF + CL\_VADD\_EMF)$, GAMMA is scaled linearly between 0 and its maximum value.
3. In case $|V\_ENC\_MEAN| > (CL\_VMIN\_EMF + CL\_VADD\_EMF)$, GAMMA = *CL_GAMMA*.

The chart below identifies the actual parameter GAMMA, which is dependent on the above described situations:

Areas of Special Concern

If γ-correction is turned on, the maximum possible commutation is (*CL_BETA* + *CL_GAMMA*).
This value must not exceed 180° (511 microsteps at 256 microsteps per fullstep) because angles of 180° or more will result in unwanted motion direction changes.

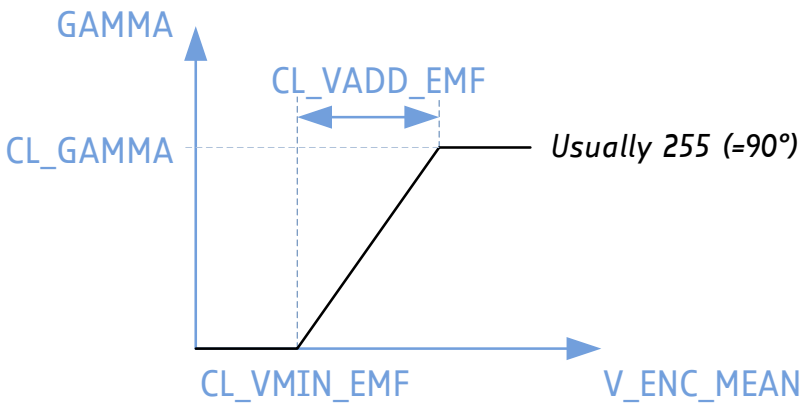


Figure 58: Calculation of the actual Load Angle GAMMA

12.3.7. Encoder Velocity Readout Parameters

In case an encoder is connected, REAL motor velocity can be read out. The actual encoder velocity flickers. This is system-immanent. TMC4330A provides filter options that back-EMF compensation is based on. The following velocity parameters can be read out.

V_ENC 0x65 Actual encoder velocity in pulses (microsteps) per second [pps].

V_ENC_MEAN 0x66 Actual filtered encoder velocity in pulses (microsteps) per second [pps].

12.3.8. Encoder Velocity Filter Configuration

In order to set filter parameters correctly, consider the following details:

ENC_VMEAN_WAIT 0x63 (7:0) *ENC_VMEAN_WAIT* represents the delay period in number of clock cycles between two consecutive *V_ENC* values that are used for the encoder filter velocity calculation. The lower this value, the faster the adaptation process of *V_ENC_MEAN* is. Accordingly: The higher the gradient of *V_ENC_MEAN* is.
In case incremental ABN encoders are connected, *ENC_VMEAN_WAIT* must be set above 32.
In case absolute encoders are connected, *ENC_VMEAN_WAIT* is automatically set to *SER_PTIME*.

ENC_VMEAN_FILTER 0x63 (11:8) This filter exponent is used for filter calculations. The lower this value, the faster the adaptation process of *V_ENC_MEAN* is. Accordingly: The higher the gradient of *V_ENC_MEAN* is. Every *ENC_VMEAN_WAIT* clock cycles, the following calculation applies:

$$V_{ENC_{MEAN}} = \mathrm{V_{ENC_{MEAN}}} - \frac{\mathrm{V_{ENC_{MEAN}}}}{2^{ENC\_VMEAN\_FILTER}} + \frac{\mathrm{V}_{ENC}}{2^{ENC\_VMEAN\_FILTER}}$$

ENC_VMEAN_INT 0x63 (31:16) The refresh frequency of high encoder velocity values *V_ENC* is determined by this encoder velocity update period.
In case incremental ABN encoders are connected, the minimum value of *ENC_VMEAN_INT* is automatically set to 256.
In case absolute encoders are connected, *ENC_VMEAN_INT* is automatically adapted to encoder value request rate.

12.3.9. Encoder Velocity equals 0 Event

Because internal calculation of low *V_ENC* values is triggered by AB signal changes and not by the refresh frequency defined by *ENC_VMEAN_INT*, any occurring idle state of the encoder is not recognized.
In order to determine that *V_ENC* = 0, it is possible to limit the number of clock cycles while no AB signal changes occur; which then signifies encoder idle state.

In order to evoke encoder idle state, do as follows:

Action:
➢ Set proper *ENC_VEL_ZERO* register 0x62.

Result:
In case no AB signal changes occur during *ENC_VEL_ZERO* clock cycles, *ENC_VEL0* event is triggered, which indicates encoder idle state.

Read entire documentation; especially the "Supplemental Directives" on page 160.

13. Reset and Clock Gating

In addition to the hardware reset pin NRST and the automatic Power-on-Reset procedure, TMC4330A provides a software reset option. If not in operation, clock gating can be used to reduce power consumption.

Reset and Clock Pins		
Pin Names	**Types**	**Remarks**
NRST	Input	Low active hardware reset.
STPIN	Input	High active wake-up signal.
CLK_EXT	Input	Connected external clock signal.

Table 48: Dedicated Reset and Clock Pins

Reset and Clock Gating Registers			
Register Name	**Register address**		**Remarks**
GENERAL_CONF	0x00	RW	Bit18:17
CLK_GATING_DELAY	0x14	RW	Dela time before clock gating is enabled.
CLK_GATING_REG	0x4F (2:0)	RW	Trigger for clock gating.
RESET_REG	0x4F (31:8)	RW	Trigger for SW-Reset.

Table 49: Dedicated Reset and Clock Gating Registers

13.1. Manual Hardware Reset

A hardware reset is provided by the NRST input pin.

In order to reset TMC4330A, do as follows:

Action:
- Set NRST input to low voltage level.

Result:
TMC4330A registers are reset to default values.

NOTE:

→ *During power-up of TMC4330A, Power-on-Reset is executed automatically.*

13.2. Manual Software Reset

In order to reset TMC4330A without use of NRST pin, do as follows:

Action:
- Set *RESET_REG* = 0x525354 (Bits31:8 of register 0x4F).

Result:
TMC4330A registers are reset to default values.

13.3. Reset Indication

***RST_EV* = EVENTS(31) is set as indicator signifying that one of the possible reset conditions was triggered.**

Read entire documentation; especially the "Supplemental Directives" on page 160.

13.4. Activating Clock Gating manually

Clock gating must be enabled before activation. In addition, the delay between activation and the active clock gating phase can be configured.

In order to activate clock gating manually, do as follows:

PRECONDITION: VEL_STATE_F = "00" INDICATING THAT VACTUAL = 0.

Action:

- Set *clk_gating_en* = 1 (bit17 of *GENERAL_CONF* register 0x00).
- Set proper *CLK_GATING_DELAY* register 0x14.
- Set *CLK_GATING_REG* = 0x7 (bit2:0 of register 0x4F).

Result:

When writing to *CLK_GATING_REG*, this activates the *CLK_GATING_DELAY* counter, which specifies the delay between clock gating trigger and activation in [number of cycles]. When the counter reaches 0, clock gating is activated. See figure below.

NOTE :

→ *In case CLK_GATING_REG = 0, clock gating is executed immediately after activating the CLK_GATING_REG register. See figure below.*

13.5. Clock Gating Wake-up

In order to conduct clock gating wake-up, do as follows:

Action:

- Set STPIN input pin to high voltage level.

Result:

Clock-gating is terminated. See figure below.

If SPI datagram transfers from microcontroller to TMC4330A prompt wake-up, do as follows:

Action:

- Set *CLK_GATING_DELAY* = 0xFFFFFFFF (register 0x14).
- Set *CLK_GATING_REG* = 0x0 (bit2:0 of register 0x4F).
- Set *CLK_GATING_REG* = 0x7 (bit2:0 of register 0x4F).
- Set *clk_gating_en* = 0 (bit17 of *GENERAL_CONF* register 0x00).

Result:

Clock-gating is terminated.

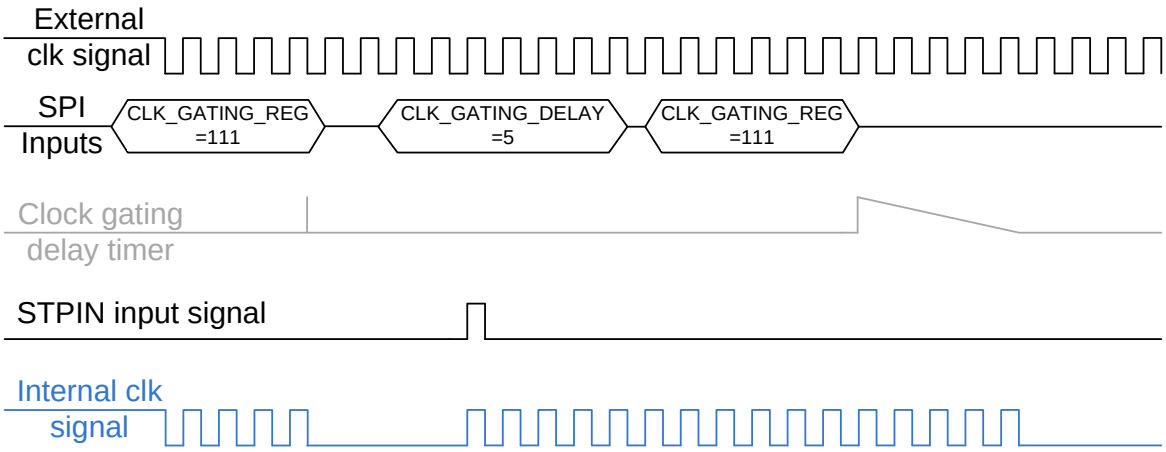


Figure 59: Manual Clock Gating Activation and Wake-Up

13.6. Automatic Clock Gating Procedure

It is possible to use TMC4330A standby phase to automatically activate clock gating.

In order to activate automatic clock gating, do as follows:

Action:

- Set the time frame for *STDBY_DELAY* register 0x15 after ramp stop, and before standby phase starts.
- Set *stdby_en* = 1 (*SCALE_CONF* register 0x05).
- Set *clk_gating_en* = 1 (bit17 of *GENERAL_CONF* register 0x00).
- Set proper *CLK_GATING_DELAY* register 0x14.
- Set *clk_gating_stdby_en* = 1 (bit17 of *GENERAL_CONF* register 0x00).

Result:
After standby phase activation, activation of clock gating counter follows. When the counter reaches 0, clock gating is activated.

In addition, the start signal generation, presented in chapter 9, page 64, can be used for an automated wake-up. An example is given in the figure below.
The chart below shows the TARGET_REACHED (=TR) signal, which signifies ramp stop at which *VACTUAL* reaches 0.
When *VACTUAL* = 0, the following process occurs:

1. The start delay timer signifies the time frame between ramp stop and next ramp start.
2. When the standby delay timer expires, the standby phase is activated.
3. When the standby phase is activated, the clock gating delay timer is started.
4. After the clock gating delay timer expires, clock gating is activated.
5. Shortly before the start delay timer expires, clock gating is disabled, which occurs so that the next ramp is started with proper assigned registers.

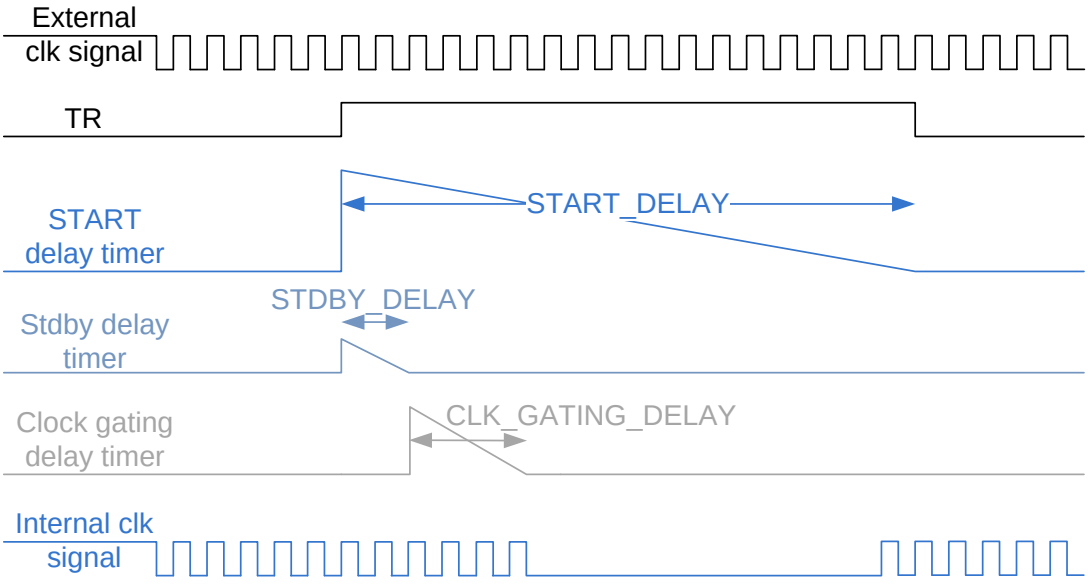


Figure 60: Automatic Clock Gating Activation and Wake-Up

TECHNICAL SPECIFICATIONS

14. Complete Register and Switches List

14.1. General Configuration Register GENERAL_CONF 0x00

GENERAL_CONF 0x00 *(Default value: 0x00006020)*			
R/W	**Bit**	**Val**	**Remarks**
RW	0	*use_astart_and_vstart*	***(only valid for S-shaped ramps)***
		0	Sets *AACTUAL* = *AMAX* or *–AMAX* at ramp start and in the case of *VSTART* ≠ 0.
		1	Sets *AACTUAL* = *ASTART* or *–ASTART* at ramp start and in the case of *VSTART* ≠ 0.
	1	*direct_acc_val_en*	
		0	Acceleration values are divided by *CLK_FREQ.*
		1	Acceleration values are set directly as steps per clock cycle.
	2	*direct_bow_val_en*	
		0	Bow values are calculated due to division by *CLK_FREQ.*
		1	Bow values are set directly as steps per clock cycle.
	3	*step_inactive_pol*	
		0	STPOUT = 1 indicates an active step.
		1	STPOUT = 0 indicates an active step.
	4	*toggle_step*	
		0	Only STPOUT transitions from inactive to active polarity indicate steps.
		1	Every level change of STPOUT indicates a step.
	5	*pol_dir_out*	
		0	DIROUT = 0 indicates negative direction.
		1	DIROUT = 1 indicates negative direction.
	7:6	*sdin_mode*	
		0	Internal step control (internal ramp generator will be used)
		1	External step control via STPIN / DIRIN interface with high active steps at STPIN
		2	External step control via STPIN / DIRIN interface with low active steps at STPIN
		3	External step control via STPIN / DIRIN interface with toggling steps at STPIN
	8	*pol_dir_in*	
		0	DIRIN = 0 indicates negative direction.
		1	DIRIN = 1 indicates negative direction.
	9	*sd_indirect_control*	
		0	STPIN/DIRIN input signals will manipulate internal steps at *XACTUAL* directly.
		1	STPIN/DIRIN input signals will manipulate *XTARGET* register value, the internal ramp generator is used.
	•→Continued on next page.		

Read entire documentation; especially the "Supplemental Directives" on page 160.

GENERAL_CONF 0x00 *(Default value: 0x00006020)*			
R/W	**Bit**	**Val**	**Remarks**
RW	11:10	*serial_enc_in_mode*	
		0	An incremental encoder is connected to encoder interface.
		1	An absolute SSI encoder is connected to encoder interface.
		2	Reserved
		3	An absolute SPI encoder is connected to encoder interface.
	12	*diff_enc_in_disable*	
		0	Differential encoder interface inputs enabled.
		1	Differential encoder interface inputs is disabled (automatically set for SPI encoder).
	14:13	Reserved. Set to 0.	
	15	*intr_pol*	
		0	INTR=0 indicates an active interrupt.
		1	INTR=1 indicates an active interrupt.
	16	*invert_pol_target_reached*	
		0	TARGET_REACHED signal is set to 1 to indicate a target reached event.
		1	TARGET_REACHED signal is set to 0 to indicate a target reached event.
	17	*clk_gating_en*	
		0	Clock gating is disabled.
		1	Internal clock gating is enabled.
	18	*clk_gating_stdby_en*	
		0	No clock gating during standby phase.
		1	Intenal clock gating during standby phase is enabled.
	22:19	Reserved. Set to 0x0.	
	23	*pwm_out_en*	
		0	PWM output is disabled. Step/Dir output is enabled at STPOUT/DIROUT.
		1	STPOUT/DIROUT output pins are used as PWM output (PWMA/PWMB).
	25:24	Reserved. Set to 0x0.	
	26	*automatic_direct_sdin_switch_off*	
		0	*VACTUAL*=0 & *AACTUAL*=0 after switching off direct external step control.
		1	*VACTUAL* = *VSTART* and *AACTUAL* = *ASTART* after switching off direct external step control.
	27	*circular_cnt_as_xlatch*	
		0	The register value of *X_LATCH* is forwarded at register 0x36.
		1	The register value of *REV_CNT* (#internal revolutions) is forwarded at register 0x36.
	28	*reverse_motor_dir*	
		0	The direction of the internal MSLUT is regularly used.
		1	The direction of internal MSLUT is reversed
	•→Continued on next page.		

Read entire documentation; especially the "Supplemental Directives" on page 160.

GENERAL_CONF 0x00 *(Default value: 0x00006020)*			
R/W	**Bit**	**Val**	**Remarks**
RW	29	*intr_tr_pu_pd_en*	
		0	INTR and TARGET_REACHED are outputs with strongly driven output values..
		1	INTR and TARGET_REACHED are used as outputs with gated pull-up and/or pull-down functionality.
	30	*intr_as_wired_and*	
		0	INTR output function is used as Wired-Or in the case of *intr_tr_pu_pd_en* = 1.
		1	INTR output function is used as Wired-And. in the case of *intr_tr_pu_pd_en* = 1.
	31	*tr_as_wired_and*	
		0	TARGET_REACHED output function is used as Wired-Or in the case of *intr_tr_pu_pd_en* = 1.
		1	TARGET_REACHED output function is used as Wired-And in the case of *intr_tr_pu_pd_en* = 1.

Table 50: General Configuration 0x00

14.2. Reference Switch Configuration Register REFERENCE_CONF 0x01

REFERENCE_CONF 0x01 *(Default value: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	0	*stop_left_en*	
		0	STOPL signal processing disabled.
		1	STOPL signal processing enabled.
	1	*stop_right_en*	
		0	STOPR signal processing disabled.
		1	STOPR signal processing enabled.
	2	*pol_stop_left*	
		0	STOPL input signal is low active.
		1	STOPL input signal is high active.
	3	*pol_stop_right*	
		0	STOPR input signal is low active.
		1	STOPR input signal is high active.
	4	*invert_stop_direction*	
		0	STOPL/STOPR stops motor in negative/positive direction.
		1	STOPL/STOPR stops motor in positive/negative direction.
	5	*soft_stop_en*	
		0	Hard stop enabled. *VACTUAL* is immediately set to 0 on any external stop event.
		1	Soft stop enabled. A linear velocity ramp is used for decreasing *VACTUAL* to v = 0.
	6	*virtual_left_limit_en*	
		0	Position limit VIRT_STOP_LEFT disabled.
		1	Position limit VIRT_STOP_LEFT enabled.
	7	*virtual_right_limit_en*	
		0	Position limit VIRT_STOP_RIGHT disabled.
		1	Position limit VIRT_STOP_RIGHT enabled.
	9:8	*virt_stop_mode*	
		0	Reserved.
		1	Hard stop: *VACTUAL* is set to 0 on a virtual stop event.
		2	Soft stop is enabled with linear velocity ramp (from *VACTUAL* to v = 0).
		3	Reserved.
	10	*latch_x_on_inactive_l*	
		0	No latch of *XACTUAL* if STOPL becomes inactive.
		1	*X_LATCH* = *XACTUAL* is stored in the case STOPL becomes inactive.
	11	*latch_x_on_active_l*	
		0	No latch of *XACTUAL* if STOPL becomes active.
		1	*X_LATCH* = *XACTUAL* is stored in the case STOPL becomes active.

•→*Continued on next page.*

REFERENCE_CONF 0x01 *(Default value: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	12	*latch_x_on_inactive_r*	
		0	No latch of *XACTUAL* if STOPR becomes inactive.
		1	*X_LATCH* = *XACTUAL* is stored in the case STOPL becomes inactive.
	13	*latch_x_on_active_r*	
		0	No latch of *XACTUAL* if STOPR becomes active.
		1	*X_LATCH* = *XACTUAL* is stored in the case STOPL becomes active.
	14	*stop_left_is_home*	
		0	STOPL input signal is not also the HOME position.
		1	STOPL input signal is also the HOME position.
	15	*stop_right_is_home*	
		0	STOPR input signal is not lso the HOME position.
		1	STOPR input signal is also the HOME position.
	19:16	*home_event*	
		0	Next active N event of connected ABN encoder signal indicates HOME position.
		2	HOME_REF = 1 indicates an active home event *X_HOME* is located at the rising edge of the active range.
		3	HOME_REF = 0 indicates negative region/position from the home position.
		4	HOME_REF = 1 indicates an active home event *X_HOME* is located at the falling edge of the active range.
		6	HOME_REF = 1 indicates an active home event *X_HOME* is located in the middle of the active range.
		9	HOME_REF = 0 indicates an active home event *X_HOME* is located in the middle of the active range.
		11	HOME_REF = 0 indicates an active home event *X_HOME* is located at the rising edge of the active range.
		12	HOME_REF = 1 indicates negative region/position from the home position.
		13	HOME_REF = 0 indicates an active home event *X_HOME* is located at the falling edge of the active range.
	20	*start_home_tracking*	
		0	No storage to *X_HOME* by passing home position.
		1	Storage of *XACTUAL* as *X_HOME* at next regular home event. An *XLATCH_DONE* event is released. In case the event is cleared, *start_home_tracking* is reset automatically.
	21	*clr_pos_at_target*	
		0	Ramp stops at *XTARGET* if positioning mode is active.
		1	Set *XACTUAL* = 0 after *XTARGET* has been reached. The next ramp starts immediately.
	22	*circular_movement_en*	
		0	Range of *XACTUAL* is not limited: $-2^{31} \leq XACTUAL \leq 2^{31}-1$
		1	Range of *XACTUAL* is limited by *X_RANGE*: $-X\_RANGE \leq XACTUAL \leq X\_RANGE - 1$
		•→Continued on next page.	

Read entire documentation; especially the "Supplemental Directives" on page 160.

REFERENCE_CONF 0x01 *(Default value: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	24:23	*pos_comp_output*	
		0	TARGET_REACHED is set active on *TARGET_REACHED_F* flag.
		1	TARGET_REACHED is set active on *VELOCITY_REACHED_F* flag.
		2	TARGET_REACHED is set active on *ENC_FAIL* flag.
		3	TARGET_REACHED triggers on *POSCOMP_REACHED_F* flag.
	25	*pos_comp_source*	
		0	*POS_COMP* is compared to internal position *XACTUAL*.
		1	*POS_COMP* is compared with external position *ENC_POS*.
	27:26	Reserved. Set to 0x0.	
	29:28	*modified_pos_compare:* *POS_COMP_REACHED_F / event is based on comparison between XACTUAL resp. ENC_POS and*	
		0	*POS_COMP*
		1	*X_HOME*
		2	*X_LATCH* resp. *ENC_LATCH*
		3	*REV_CNT*
	30	Reserved. Set to 0.	
	31	*circular_enc_en*	
		0	Range of *ENC_POS* is not limited: $-2^{31} \leq ENC\_POS \leq 2^{31}-1$
		1	Range of *ENC_POS* is limited by *X_RANGE*: $-X\_RANGE \leq ENC\_POS \leq X\_RANGE-1$

Table 51: Reference Switch Configuration 0x01

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.3. Start Switch Configuration Register START_CONF 0x02

START_CONF 0x02 *(Default value: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	4:0	*start_en*	
		xxxx1	Alteration of *XTARGET* value requires distinct start signal.
		xxx1x	Alteration of *VMAX* value requires distinct start signal.
		xx1xx	Alteration of *RAMPMODE* value requires distinct start signal.
		x1xxx	Alteration of *GEAR_RATIO* value requires distinct start signal.
		1xxxx	Shadow Register Feature Set is enabled.
	8:5	*trigger_events*	
		0000	Timing feature set is disabled because start signal generation is disabled.
		xxx0	START pin is assigned as output.
		xxx1	External start signal is enabled as timer trigger. START pin is assigned as input.
		xx1x	*TARGET_REACHED* event is assigned as start signal trigger.
		x1xx	*VELOCITY_REACHED* event is assigned as start signal trigger.
		1xxx	*POSCOMP_REACHED* event is assigned as start signal trigger.
	9	*pol_start_signal*	
		0	START pin is low active (input resp. output).
		1	START pin is high active (input resp. output).
	10	*immediate_start_in*	
		0	Active START input signal starts internal start timer.
		1	Active START input signal is executed immediately.
	11	*busy_state_en*	
		0	START pin is only assigned as input or output.
		1	Busy start state is enabled. START pin is assigned as input with a weakly driven active start polarity or as output with a strongly driven inactive start polarity.
	15:12	*pipeline_en*	
		0000	No pipelining is active.
		xxx1	*X_TARGET* is considered for pipelining.
		xx1x	*POS_COMP* is considered for pipelining.
		x1xx	*GEAR_RATIO* is considered for pipelining.
		1xxx	*GENERAL_CONF* is considered for pipelining.
	17:16	*shadow_option*	
		0	Single-level shadow registers for 13 relevant ramp parameters.
		1	Double-stage shadow registers for S-shaped ramps.
		2	Double-stage shadow registers for trapezoidal ramps (excl. *VSTOP*).
		3	Double-stage shadow registers for trapezoidal ramps (excl. *VSTART*).
		•→*Continued on next page.*	

START_CONF 0x02 *(Default value: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	18	*cyclic_shadow_regs*	
		0	Current ramp parameters are not written back to the shadow register.
		1	Current ramp parameters are written back to the appropriate shadow register.
	19	Reserved. Set to 0.	
	23:20	*SHADOW_MISS_CNT*	
		U	Number of unused start internal start signals between two consecutive shadow register transfers.
	31:24	*XPIPE_REWRITE_REG*	
			Current assigned pipeline registers – *START_CONF*(15:12) – are written back to *X_PIPEx* in the case of an internal start signal generation and if assigned in this register with a '1': *XPIPE_REWRITE_REG*(0) → *X_PIPE0* *XPIPE_REWRITE_REG*(1) → *X_PIPE1* *XPIPE_REWRITE_REG*(2) → *X_PIPE2* *XPIPE_REWRITE_REG*(3) → *X_PIPE3* *XPIPE_REWRITE_REG*(4) → *X_PIPE4* *XPIPE_REWRITE_REG*(5) → *X_PIPE5* *XPIPE_REWRITE_REG*(6) → *X_PIPE6* *XPIPE_REWRITE_REG*(7) → *X_PIPE7* Ex.: *START_CONF*(15:12) = b'0011. *START_CONF*(31:24) = b'01000010. If an internal start signal is generated, the value of *X_TARGET* is written back to *X_PIPE1*, whereas the value of *POS_COMP* is written back to *X_PIPE6*.

Table 52: Start Switch Configuration START_CONF 0x02

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.4. Input Filter Configuration Register INPUT_FILT_CONF 0x03

INPUT_FILT_CONF 0x03 *(Default value: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	2:0	*SR_ENC_IN*	
		U	Input sample rate = f_{clk} / $2^{SR\_ENC\_IN}$ for the following pins: A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NSDI, N, NNEG
	3	Reserved. Set to 0.	
	6:4	*FILT_L_ENC_IN*	
		U	Filter length for these pins: A_SCLK, ANEG_NSCLK, B_SDI, BNEG_NSDI, N, NNEG. Number of sample input bits that must have equal voltage levels to provide a valid input bit.
	7	Reserved. Set to 0.	
	10:8	*SR_REF*	
		U	Input sample rate = f_{clk} / 2^{REF} for the following pins: STOPL, HOME_REF, STOPL
	11	Reserved. Set to 0.	
	14:12	*FILT_L_REF*	
		U	Filter length for the following pins: STOPL, HOME_REF, STOPL. Number of sample input bits that must have equal voltage levels to provide a valid input bit.
	15	Reserved. Set to 0.	
	18:16	*SR_S*	
		U	Input sample rate = f_{clk} / 2^{S} for the START pin.
	19	Reserved. Set to 0.	
	22:20	*FILT_L_S*	
		U	Filter length for the START pin. Number of sample input bits that must have equal voltage levels to provide a valid input bit.
	23	Reserved. Set to 0.	
	26:24	*SR_SD_IN*	
		U	Input sample rate = f_{clk} / $2^{SR\_SD\_IN}$ for these pins: STPIN, DIRIN
	27	Reserved. Set to 0.	
	30:28	*FILT_L_ENC_OUT*	
		U	Filter length for the following pins: STPIN, DIRIN. Number of sample input bits that must have equal voltage levels to provide a valid input bit.
	31	!	***Reserved. Set to 1.***

Table 53: Input Filter Configuration Register INPUT_FILT_CONF 0x03

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.5. Scaling Configuration Register SCALE_CONF 0x05

SCALE_CONF 0x05 *(Default: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	0	*stdby_en*	
		0	Standby phase is disabled.
		1	Standby phase is enabled.
	7:1	Reserved. Set to 0x00.	
	8	*pwm_scale_en*	
		0	PWM scaling is disabled.
		1	PWM scaling is enabled.
	15:9	Reserved. Set to 0x00.	
	31:16	*PWM_AMPL*	
		U	PWM amplitude during Voltage PWM mode at *VACTUAL* = 0. **i** **Maximum duty cycle = (0.5 + (*PWM_AMPL* + 1) / 2^{17})** **Minimum duty cycle = (0.5 – (*PWM_AMPL* + 1) / 2^{17})** **PWM_AMPL = 2^{16} – 1 at *VACTUAL* = *PWM_VMAX.***

Table 54: Current Scale Configuration (0x05)

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.6. Encoder Signal Configuration (0x07)

***ENC_IN_CONF* 0x07** *(Default 0x00000400)*			
R/W	**Bit**	**Val**	**Description**
RW	0	*enc_sel_decimal*	
		0	Encoder constant represents a binary number.
		1	Encoder constant represents a decimal number (for ABN only).
	1	*clear_on_n*	
		0	*ENC_POS* is not set to *ENC_RESET_VAL*.
		1	*ENC_POS* is set to *ENC_RESET_VAL* on every N event in case *clr_latch_cont_on_n*=1, or on the next N event in case *clr_latch_once_on_n*=1. **! Do NOT use during closed-loop operation.**
	2	*clr_latch_cont_on_n*	
		0	Value of *ENC_POS* is not cleared and/or latched on every N event.
		1	Value of *ENC_POS* is cleared and/or latched on every N event.
	3	*clr_latch_once_on_n*	
		0	Value of *ENC_POS* is not cleared and/or latched on the next N event.
		1	Value of *ENC_POS* is cleared and/or latched on the next N event. **i This bit is set to 0 after latching/clearing once.**
	4	*pol_n*	
		0	Active polarity for N event is low active.
		1	Active polarity for N event is high active.
	6:5	*n_chan_sensitivity*	
		0	N event is active as long as N equals active N event polarity.
		1	N event triggers when N switches to active N event polarity.
		2	N event triggers when N switches to inactive N event polarity.
		3	N event triggers when N switches to in-/active N event polarity (both slopes).
	7	*pol_a_for_n*	
		0	A polarity has to be low for a valid N event.
		1	A polarity has to be high for a valid N event.
	8	*pol_b_for_n*	
		0	B polarity has to be low for valid N event
		1	B polarity has to be high for valid N event
	9	*ignore_ab*	
		0	TMC4330A considers A and B polarities for valid N event.
		1	Polarities of A and B signals for a valid N event are ignored.

•→*Continued on next page.*

***ENC_IN_CONF* 0x07** *(Default 0x00000400)*			
R/W	**Bit**	**Val**	**Description**
RW	10	*latch_enc_on_n*	
		0	*ENC_POS* is not latched.
		1	*ENC_POS* is latched to *ENC_LATCH* on every N event in case *clr_latch_cont_on_n*=1, or on the next N event in case *clr_latch_once_on_n*=1.
	11	*latch_x_on_n*	
		0	*XACTUAL* is not latched.
		1	*XACTUAL* is latched to *X_LATCH* on every N event in case *clr_latch_cont_on_n*=1, or on the next N event in case *clr_latch_once_on_n*=1.
	12	*multi_turn_in_en*	***(Absolute encoder only)***
		0	Connected serial encoder transmits singleturn values.
		1	Connected serial encoder input transmits singleturn and multiturn values.
	13	*multi_turn_in_signed*	***(Absolute encoder only)***
		0	Multiturn values from serial encoder input are unsigned numbers.
		1	Multiturn values from serial encoder input are signed numbers.
	14	Reserved. Set to 0.	
	15	*use_usteps_instead_of_xrange*	
		0	*X_RANGE* is valid in case circular motion is also enabled for encoders.
		1	*USTEPS_PER_REV* is valid in case circular motion is also enabled for encoders.
	16	*calc_multi_turn_behav*	***(Absolute encoder only)***
		0	No multiturn calculation.
		1	TMC4330A calculates internally multiturn data for singleturn encoder data.
	17	*ssi_multi_cycle_data*	***(Absolute encoder only)***
		0	Every SSI value request is executed once.
		1	Every SSI value request is executed twice.
	18	*ssi_gray_code_en*	***(Absolute encoder only)***
		0	SSI input data is binary-coded.
		1	SSI input data is gray-coded.
	19	*left_aligned_data*	***(Absolute encoder only)***
		0	Serial input data is aligned right (first flags, then data).
		1	Serial input data is aligned left (first data, then flags).
		•→Continued on next page.	

***ENC_IN_CONF* 0x07** *(Default 0x00000400)*			
R/W	**Bit**	**Val**	**Description**
RW	20	*spi_data_on_cs*	***(SPI encoder only)***
		0	BNEG_NSDI provides serial output data at next serial clock line (A_SCLK) transition.
		1	BNEG_NSDI provides serial output data immediately in case negated chip select line ANEG_NSCLK switches to low level.
	21	*spi_low_before_cs*	***(SPI encoder only)***
		0	Serial clock line A_SCLK switches to low level after negated chip select line ANEG_NSCLK switches to low level.
		1	Serial clock line A_SCLK switches to low level before negated chip select line ANEG_NSCLK switches to low level.
	23:22	*regulation_modus*	
		0	No internal regulation on encoder feedback data.
		1	Closed-loop operation is enabled. **! Use full microstep resolution only! (256 µSteps/FS → MSTEPS_PER_FS=0).**
		2	PID regulation is enabled. Pulse generator base velocity equals 0.
		3	PID regulation is enabled. Pulse generator base velocity equals *VACTUAL*.
	24	*cl_calibration_en*	***(Closed-loop operation only)***
		0	Closed-loop calibration is deactivated.
		1	Closed-loop calibration is active. **! Use maximum current without scaling during calibration.** **! It is recommend to keep the motor driver at fullstep position with no motion occurrence during the calibration process.**
	25	*cl_emf_en*	***(Closed-loop operation only)***
		0	Back-EMF compensation deactivated during closed-loop operation.
		1	Back-EMF compensation is enabled during closed-loop operation. Closed-loop operation compensates Back-EMF in case |*VACTUAL*| > *CL_VMIN*.
	26	*cl_clr_xact*	***(Closed-loop operation only)***
		0	*XACTUAL* is not reset to *ENC_POS* during closed-loop operation.
		1	*XACTUAL* is set to *ENC_POS* in case |*ENC_POS_DEV*| > *ENC_POS_DEV_TOL* during closed-loop operation. **! This feature must only be used if understood completely.**
	27	*cl_vlimit_en*	***(Closed-loop operation only)***
		0	No catch-up velocity limit during closed-loop regulation.
		1	Catch-up velocity during closed-loop operation is limited by internal PI regulator.
	28	*cl_velocity_mode_en*	***(Closed-loop operation only)***
		0	Closed-loop velocity mode is deactivated.
		1	Closed-loop velocity mode is deactivated. In case |*ENC_POS_DEV*| > 768, *XACTUAL* is adjusted accordingly.
	29	*invert_enc_dir*	
		0	Encoder direction is NOT inverted internally.
		1	Encoder direction is inverted internally.

•→Continued on next page.

Read entire documentation; especially the "Supplemental Directives" on page 160.

***ENC_IN_CONF* 0x07** *(Default 0x00000400)*			
R/W	**Bit**	**Val**	**Description**
RW	30	Reserved. Set to 0x0.	
	31	*no_enc_vel_preproc*	***(Incremental ABN encoder)***
		0	AB signal is preprocessed for internal encoder velocity calculation.
		1	No AB signal preprocessing. **! It is recommend to maintain AB preprocessing in order to filter encoder resonances.**
		serial_enc_variation_limit	***(Absolute encoder)***
		0	No variation limit on absolute encoder data.
		1	Two consecutive serial encoder values must no deviate from specified limit to be valid. In case $\|ENC\_POS_X - ENC\_POS_{X-1}\| > 1/8 \cdot SER\_ENC\_VARIATION \cdot ENC\_IN\_RES$, ENC_POS_X is not valid and is not assigned to *ENC_POS*.

Table 55: Encoder Signal Configuration ENC_IN_CONF (0x07)

14.7. Serial Encoder Data Input Configuration (0x08)

ENC_IN_DATA 0x08 *(Default: 0x00000000)*			
R/W	**Bit**	**Val**	**Remarks**
RW	4:0	*SINGLE_TURN_RES (Default: 0x00)*	
		U	Number of angle data bits within one revolution = *SINGLE_TURN_RES* + 1. **! Set *SINGLE_TURN_RES* < 31.**
	9:5	*MULTI_TURN_RES (Default: 0x00)*	
		U	Number of data bits for revolution count = *MULTI_TURN_RES* + 1
	11:10	*STATUS_BIT_CNT (Default: 0x0)*	
		U	Number of status data bits
	15:12	Reserved. Set to 0x0.	
	23:16	*SERIAL_ADDR_BITS (Default: 0x00)*	***(SPI encoder only)***
		U	Number of address bits within one SPI datagram for SPI encoder configuration
	31:24	*SERIAL_DATA_BITS (Default: 0x00)*	***(SPI encoder only)***
		U	Number of data bits within one SPI datagram for SPI encoder configuration

Table 56: Serial Encoder Data Input Configuration ENC_IN_DATA (0x08)

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.8. Microstep Settings Register STEP_CONF 0x0A

STEP_CONF 0x0A *(Default: 0x00FB0C80)*			
R/W	**Bit**	**Val**	**Remarks**
RW	3:0	*MSTEP_PER_FS (Default: 0x0)*	
		0	Highest microsteps resolution: 256 microsteps per fullstep. **i Set to 256 for closed-loop operation.** **i When using a Step/Dir driver, it must be capable of a 256 resolution via Step/Dir input for best performance (but lower resolution Step/Dir drivers can be used as well).**
		1	128 microsteps per fullstep.
		2	64 microsteps per fullstep.
		3	32 microsteps per fullstep.
		4	16 microsteps per fullstep.
		5	8 microsteps per fullstep.
		6	4 microsteps per fullstep.
		7	Halfsteps: 2 microsteps per fullstep.
		8	Full steps (maximum possible setting)
	15:4	*FS_PER_REV (Default: 0x0C8)*	
		U	Fullsteps per motor axis revolution
	31:16	Reserved. Set to 0x0000.	

Table 57: Motor Driver Settings (0x0A)

14.9. Event Selection Registers 0x0B..0X0D

Event Selection Registers			
R/W	**Addr**	**Bit**	**Remarks**
RW	0x0B	*SPI_STATUS_SELECTION (Default: 0x82029805)*	
		31:0	Events selection for SPI datagrams: Event bits of *EVENTS* register 0x0E that are selected (=1) in this register are forwarded to the eight status bits that are transferred with every SPI datagram (first eight bits from LSB are significant!).
	0x0C	*EVENT_CLEAR_CONF (Default: 0x00000000)*	
		31:0	Event protection configuration: Event bits of *EVENTS* register 0x0E that are selected in this register (=1) are not cleared during the readout process of *EVENTS* register 0x0E.
	0x0D	*INTR_CONF (Default: 0x00000000)*	
		31:0	Event selection for INTR output: All Event bits of *EVENTS* register 0x0E that are selected here (=1) are ORed with interrupt event register set: if any of the selected events is active, an interrupt at INTR is generated.

Table 58: Event Selection Regsiters 0x0B…0x0D

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.10. Status Event Register (0x0E)

Status Event Register *EVENTS 0x0E*		
R/W	**Bit**	**Description**
R+CW	0	*TARGET_REACHED* has been triggered.
	1	*POS_COMP_REACHED* has been triggered.
	2	*VEL_REACHED* has been triggered.
	3	*VEL_STATE* = b'00 has been triggered (*VACTUAL* = 0).
	4	*VEL_STATE* = b'01 has been triggered (*VACTUAL* > 0).
	5	*VEL_STATE* = b'10 has been triggered (*VACTUAL* < 0).
	6	*RAMP_STATE* = b'00 has been triggered (*AACTUAL* = 0, *VACTUAL* is constant).
	7	*RAMP_STATE* = b'01 has been triggered (|*VACTUAL*| increases).
	8	*RAMP_STATE* = b'10 has been triggered (|*VACTUAL*| increases).
	9	*MAX_PHASE_TRAP*: Trapezoidal ramp has reached its limit speed using maximum values for *AMAX* or *DMAX* (|*VACTUAL*| > *VBREAK*; *VBREAK*≠0).
	10	Reserved.
	11	STOPL has been triggered. Motion in negative direction is not executed until this event is cleared and (STOPL is not active any more or *stop_left_en* is set to 0).
	12	STOPR has been triggered. Motion in positive direction is not executed until this event is cleared and (STOPR is not active any more or *stop_right_en* is set to 0).
	13	*VSTOPL_ACTIVE: VSTOPL* has been activated. No further motion in negative direction until this event is cleared and (a new value is chosen for *VSTOPL* or *virtual_left_limit_en* is set to 0).
	14	*VSTOPR_ACTIVE: VSTOPR* has been activated. No further motion in positive direction until this event is cleared and (a new value is chosen for *VSTOPR or virtual_right_limit_en* is set to 0).
	15	*HOME_ERROR:* Unmatched *HOME_REF* polarity and HOME is outside of safety margin.
	16	*XLATCH_DONE* indicates if *X_LATCH* was rewritten or homing process has been completed.
	17	Reserved.
	18	*ENC_FAIL*: Mismatch between *XACTUAL* and *ENC_POS* has exceeded specified limit.
	19	*N_ACTIVE:* N event has been activated.
	20	*ENC_DONE* indicates if *ENC_LATCH* was rewritten.
	21	*SER_ENC_DATA_FAIL:* Failure during multi-cycle data evaluation or between two consecutive data requests has occured.
	22	Reserved.
	23	*SER_DATA_DONE:* Configuration data was received from serial SPI encoder.
	24	One of the *SERIAL_ENC_F*lags was set.
	25	Reserved.
	26	*ENC_VEL0:* Encoder velocity has reached 0.
	27	*CL_MAX:* Closed-loop commutation angle has reached maximum value.
	28	*CL_FIT:* Closed-loop deviation has reached inner limit.
	29	Reserved.
	30	Reserved.
	31	*RST_EV*: Reset was triggered.

Table 59: Status Event Register EVENTS (0x0E)

14.11. Status Flag Register (0x0F)

Status Flag Register *STATUS 0x0F*		
R/W	**Bit**	**Description**
R	0	*TARGET_REACHED_F* is set high if *XACTUAL* = *XTARGET*
	1	*POS_COMP_REACHED_F* is set high if *XACTUAL* = *POS_COMP*
	2	*VEL_REACHED_F* is set high if *VACTUAL* = |*VMAX*|
	4:3	*VEL_STATE_F*: Current velocity state: 0 → *VACTUAL* = 0; 1 → *VACTUAL* > 0; 2 → *VACTUAL* < 0
	6:5	*RAMP_STATE_F*: Current ramp state: 0 → *AACTUAL* = 0; 1 → *AACTUAL* increases (acceleration); 2 → *AACTUAL* decreases (deceleration)
	7	*STOPL_ACTIVE_F*: Left stop switch is active.
	8	*STOPR_ACTIVE_F*: Right stop switch is active.
	9	*VSTOPL_ACTIVE_F*: Left virtual stop switch is active.
	10	*VSTOPR_ACTIVE_F*: Right virtual stop switch is active.
	11	Reserved.
	12	*HOME_ERROR_F*: HOME_REF input signal level is not equal to expected home level.
	13	Reserved.
	14	*ENC_FAIL_F*: Mismatch between *XACTUAL* and *ENC_POS* is out of tolerated range.
	15	*N_ACTIVE_F*: N event is active.
	16	*ENC_LATCH_F*: *ENC_LATCH* is rewritten.
	17	Applies to absolute encoders only: *MULTI_CYCLE_FAIL_F* indicates a failure during last multi cycle data evaluation.
		Applies to absolute encoders only: *SER_ENC_VAR_F* indicates a failure during last serial data evaluation due to a substantial deviation between two consecutive serial data values.
	18	Reserved.
	19	*CL_FIT_F*: Active if ENC_POS_DEV < CL_TOLERANCE. The current mismatch between XACTUAL and ENC_POS is within tolerated range.
	23:20	Applies to absolute encoders only: *SERIAL_ENC_FLAGS* received from encoder. These flags are reset with a new encoder transfer request.
	31:24	Reserved.

Table 60: Status Flag Register STATUS (0x0F)

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.12. Various Configuration Registers: Synchronization, PWM, etc.

Various Configuration Registers: Closed-loop, Switches…				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x04	31:0	Reserved. Set to 0x00000000.	
	0x10	15:0	*STP_LENGTH_ADD (Default: 0x0000)*	
			U	Additional length [# clock cycles] for active step polarity to indicate an active output step at STPOUT.
		31:16	*DIR_SETUP_TIME (Default: 0x0000)*	
			U	Delay [# clock cycles] between DIROUT and STPOUT voltage level changes.
	0x11	31:0	*START_OUT_ADD (Default:0x00000000)*	
			U	Additional length [# clock cycles] for active start signal. Active start signal length = 1+START_OUT_ADD
	0x12	31:0	*GEAR_RATIO (Default:0x01000000)*	
			S	Constant value that is added to the internal position counter by an active step at STPIN. Value representation: 8 digits and 24 decimal places.
	0x13	31:0	*START_DELAY (Default:0x00000000)*	
			U	Delay time [# clock cycles] between start trigger and internal start signal release.
	0x14	31:0	*CLK_GATING_DELAY (Default:0x00000000)*	
			U	Delay time [# clock cycles] between clock gating trigger and clock gating start.
	0x15	31:0	*STDBY_DELAY (Default:0x00000000)*	
			U	Delay time [# clock cycles] between ramp stop and activating standby phase.
	0x17	23:0	*PWM_VMAX (Default:0x00000000)*	
			U	PWM velocity value at which maximal scale parameter value 1.0 is reached.
	0x1E	15:0	*HOME_SAFETY_MARGIN (Default: 0x0000)*	
			U	HOME_REF polarity can be invalid within *X_HOME* ± *HOME_SAFETY_MARGIN,* which is not flagged as error.
	0x1F	15:0	*PWM_FREQ (Default: 0x0280)*	
			U	Number of clock cycles for one PWM period.
W	0x64	31:0	Reserved. Set to 0x00000000.	
	0x7B	31:0	*TZEROWAIT (Default:0x00000000)*	
			U	Standstill phase after reaching *VACTUAL* = 0.
	0x7C	31:0	*CIRCULAR_DEC (Default:0x00000000)*	
			U	Decimal places for circular motion if one revolution is not exactly mapped to an even number of µSteps per revolution. Value representation: 1 digit and 31 decimal places.

Table 61: Various Configuration Registers: Synchronization, PWM, etc.

14.13. Ramp Generator Registers

Ramp Generator Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x20	*RAMPMODE* (Default:0x0)		
		2	Operation Mode:	
			1	**Positioning mode**: *XTARGET* is superior target of velocity ramp.
			0	**Velocitiy mode**: *VMAX* is superior target of velocity ramp.
		1:0	Motion Profile:	
			0	**No ramp:** *VACTUAL* follows only *VMAX* (rectangle velocity shape).
			1	**Trapezoidal ramp (incl. sixPoint ramp):** Consideration of acceleration and deceleration values for generating *VACTUAL* without adapting the acceleration values.
			2	**S-shaped ramp:** Consideration of all ramp values (incl. bow values) for generating *VACTUAL*.
RW	0x21	31:0	*XACTUAL (Default: 0x00000000)*	
			S	Actual internal motor position [pulses]: $-2^{31} \le XACTUAL \le 2^{31} - 1$
R	0x22	31:0	*VACTUAL (Default: 0x00000000)*	
			S	Actual ramp generator velocity [pulses per second]: 1 pps ≤ |*VACTUAL*| ≤ *CLK_FREQ* · ½ pulses (f_{CLK} = 16 MHz → 8 Mpps)
R	0x23	31:0	*AACTUAL (Default: 0x00000000)*	
			S	Actual acceleration/deceleration value [pulses per sec^2]: -2^{31} pps² ≤ *AACTUAL* ≤ $2^{31} - 1$ 1 pps² ≤ |*AACTUAL*|
RW	0x24	31:0	*VMAX (Default: 0x00000000)*	
			S	Maximum ramp generator velocity in positioning mode or
				Target ramp generator velocity in velocity mode and no ramp motion profile.
				Value representation: 23 digits and 8 decimal places **! Consider maximum values, represented in section 6.6.5, page 45**
RW	0x25	30:0	*VSTART (Default: 0x00000000)*	
			U	Absolute start velocity in *positioning mode* and *velocity mode* In case *VSTART* is used: no first bow phase B_1 for S-shaped ramps
				VSTART in positioning mode: In case *VACTUAL* = 0 and *XTARGET* ≠ *XACTUAL*: no acceleration phase for *VACTUAL* = 0 → *VSTART*.
				VSTART in velocity mode: In case *VACTUAL* = 0 and *VACTUAL* ≠ *VMAX*: no acceleration phase for *VACTUAL* = 0 → *VSTART*.
				Value representation: 23 digits and 8 decimal places. **! Consider maximum values, represented in section 6.6.5, page 45**

•→ *Continued on next page.*

Ramp Generator Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x26	30:0		*VSTOP (Default:0x00000000)*
			U	Absolute stop velocity in positioning mode and velocity mode. In case *VSTOP* is used: no last bow phase B_4 for S-shaped ramps. In case *VSTOP* is very small and positioning mode is used, it is possible that the ramp is finished with a constant *VACTUAL* = *VSTOP* until *XTARGET* is reached.
				VSTOP in positioning mode: In case *VACTUAL*≤*VSTOP* and *XTARGET*=*XACTUAL*: *VACTUAL* is immediately set to 0.
				VSTOP in velocity mode: In case *VACTUAL* ≤ *VSTOP* and *VMAX* = 0: *VACTUAL* is immediately set to 0.
				Value representation: 23 digits and 8 decimal places. **! Consider maximum values, represented in section 6.6.5, page 45**
	0x27	30:0		*VBREAK (Default:0x00000000)*
			U	Absolute break velocity in positioning mode and in velocity mode, This only applies for trapezoidal ramp motion profiles. In case *VBREAK* = 0: pure linear ramps are generated with *AMAX* / *DMAX* only.
				In case |*VACTUAL*| < *VBREAK*: |*AACTUAL*| = *ASTART* or *DFINAL* In case |*VACTUAL*| ≥ *VBREAK*: |*AACTUAL*| = *AMAX* or *DMAX*
				! Always set *VBREAK* > *VSTOP*! If VBREAK ≠ 0.
				Value representation: 23 digits and 8 decimal places. **! Consider maximum values, represented in section 6.6.5, page 45**
	0x28	23:0		*AMAX (Default: 0x000000)*
			U	S-shaped ramp motion profile: Maximum acceleration value.
				Trapezoidal ramp motion profile: Acceleration value in case |*VACTUAL*| ≥ *VBREAK* or in case *VBREAK* = 0.
				Value representation: **Frequency mode**: [pulses per sec^2] 22 digits and 2 decimal places: 250 mpps^2 ≤ *AMAX* ≤ 4 Mpps^2 **Direct mode:** [Δv per clk cycle] a[Δv per clk_cycle]= *AMAX* / 2^{37} *AMAX* [pps^2] = *AMAX* / 2^{37} • f_{CLK}^2 **! Consider maximum values, represented in section 6.6.5, page 45**
	0x29	23:0		*DMAX (Default: 0x000000)*
			U	S-shaped ramp motion profile: Maximum deceleration value.
				Trapezoidal ramp motion profile: Deceleration value if |*VACTUAL*| ≥ *VBREAK* or if *VBREAK* = 0.
				Value representation: **Frequency mode**: [pulses per sec^2] 22 digits and 2 decimal places: 250 mpps^2 ≤ *DMAX* ≤ 4 Mpps^2 **Direct mode:** [Δv per clk cycle] d[Δv per clk_cycle]= *DMAX* / 2^{37} *DMAX* [pps^2] = *DMAX* / 2^{37} • f_{CLK}^2 **! Consider maximum values, represented in section 6.6.5, page 45**

•→ *Continued on next page.*

Ramp Generator Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x2A	23:0		*ASTART (Default: 0x000000)*
			U	S-shaped ramp motion profile: start acceleration value.
				Trapezoidal ramp motion profile: Acceleration value in case |*VACTUAL*| < *VBREAK*.
				Acceleration value after switching from external to internal step control.
				Value representation: **Frequency mode**: [pulses per sec^2] 22 digits and 2 decimal places: 250 mpps^2 ≤ *ASTART* ≤ 4 Mpps^2 **Direct mode:** [Δv per clk cycle] a[Δv per clk_cycle]= *ASTART* / 2^{37} *ASTART* [pps^2] = *ASTART* / 2^{37} • f_{CLK}^2 **! Consider maximum values, represented in section 6.6.5, page 45**
		31		Sign of *AACTUAL* after switching from external to internal step control.
	0x2B	23:0		*DFINAL (Default: 0x000000)*
			U	S-shaped ramp motion profile: Stop deceleration value, which is not used during positioning mode.
				Trapezoidal ramp motion profile: Deceleration value in case |*VACTUAL*| < *VBREAK*.
				Value representation: **Frequency mode**: [pulses per sec^2] 22 digits and 2 decimal places: 250 mpps^2 ≤ *DFINAL* ≤ 4 Mpps^2 **Direct mode:** [Δv per clk cycle] d[Δv per clk_cycle]= *DFINAL* / 2^{37} *DFINAL* [pps^2] = *DFINAL* / 2^{37} • f_{CLK}^2 **! Consider maximum values, represented in section 6.6.5, page 45**
	0x2C	23		*DSTOP (Default: 0x000000)*
			U	Deceleration value for an automatic linear stop ramp to *VACTUAL* = 0. *DSTOP* is used with activated external stop switches (STOPL or STOPR) if *soft_stop_enable* is set to 1; or with activated virtual stop switches and *virt_stop_mode* is set to 2.
				Value representation: **Frequency mode**: [pulses per sec^2] 22 digits and 2 decimal places: 250 mpps^2 ≤ *DSTOP* ≤ 4 Mpps^2 **Direct mode:** [Δv per clk cycle] d[Δv per clk_cycle]= *DSTOP* / 2^{37} *DSTOP* [pps^2] = *DSTOP* / 2^{37} • f_{CLK}^2 **! Consider maximum values, represented in section 6.6.5, page 45**

•→ *Continued on next page!*

Ramp Generator Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x2D	23:0		*BOW1 (Default: 0x000000)*
			U	Bow value 1 (first bow B_1 of the acceleration ramp). Value representation: **Frequency mode**: [pulses per sec^3] 24 digits and 0 decimal places: 1 pps^3 ≤ *BOW1* ≤ 16 Mpps^3 **Direct mode:** [Δa per clk cycle] bow[av per clk_cycle]= *BOW1* / 2^{53} *BOW1* [pps^3] = *BOW1* / 2^{53} • f_{CLK}^3 **! Consider maximum values, represented in section 6.6.5, page 45**
	0x2E	23:0		*BOW2 (Default: 0x000000)*
			U	Bow value 2 (second bow B2 of the acceleration ramp). Value representation: **Frequency mode**: [pulses per sec^3] 24 digits and 0 decimal places: 1 pps^3 ≤ *BOW2* ≤ 16 Mpps^3 **Direct mode:** [Δa per clk cycle] bow[av per clk_cycle]= *BOW2* / 2^{53} *BOW2* [pps^3] = *BOW2* / 2^{53} • f_{CLK}^3 **! Consider maximum values, represented in section 6.6.5, page 45**
	0x2F	23:0		*BOW3 (Default: 0x000000)*
			U	Bow value 3 (first bow B3 of the deceleration ramp). Value representation: **Frequency mode**: [pulses per sec^3] 24 digits and 0 decimal places: 1 pps^3 ≤ *BOW3* ≤ 16 Mpps^3 **Direct mode:** [Δa per clk cycle] bow[av per clk_cycle]= *BOW3* / 2^{53} *BOW3* [pps^3] = *BOW3* / 2^{53} • f_{CLK}^3 **! Consider maximum values, represented in section 6.6.5, page 45**
	0x30	23:0		*BOW 4 (Default: 0x000000)*
			U	Bow value 4 (second bow B4 of the deceleration ramp). Value representation: **Frequency mode**: [pulses per sec^3] 24 digits and 0 decimal places: 1 pps^3 ≤ *BOW4* ≤ 16 Mpps^3 **Direct mode:** [Δa per clk cycle] bow[av per clk_cycle]= *BOW4* / 2^{53} *BOW4* [pps^3] = *BOW4* / 2^{53} • f_{CLK}^3 **! Consider maximum values, represented in section 6.6.5, page 45**

Table 62: Ramp Generator Registers

14.14. External Clock Frequency Register

External Clock Frequency Register				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x31	24:0	*CLK_FREQ (Default: 0x0F42400)*	
			U	External clock frequency value f_{CLK} [Hz] with 4.2 MHz ≤ f_{CLK} ≤ 30 MHz

Table 63: External Clock Frequency Register

14.15. Target and Compare Registers

Target and Compare Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x32	31:0	*POS_COMP (Default: 0x00000000)*	
			S	Compare position.
RW	0x33	31:0	*VIRT_STOP_LEFT (Default: 0x00000000)*	
			S	Virtual left stop position.
RW	0x34	31:0	*VIRT_STOP_RIGHT (Default: 0x00000000)*	
			S	Virtual right stop position.
RW	0x35	31:0	*X_HOME (Default: 0x00000000)*	
			S	Actual home position.
R	0x36	31:0	*X_LATCH (Default: 0x00000000)*	***(if circular_cnt_as_xlatch = 0)***
			S	Storage position for certain triggers.
			REV_CNT (Default: 0x00000000)	***(if circular_cnt_as_xlatch = 1)***
			S	Number of revolutions during circular motion.
W		30:0	*X_RANGE (Default: 0x00000000)*	
			U	Limitation for *X_ACTUAL* during circular motion: *-X_RANGE* ≤ *X_ACTUAL* ≤ *X_RANGE* - 1
RW	0x37	31:0	*X_TARGET (Default: 0x00000000)*	
			U	Target motor position in positioning mode. **! Set all other motion profile parameters before!**

Table 64: Target and Compare Registers

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.16. Pipeline Registers

Pipeline Register				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x38	31:0	S	*X_PIPE0 (Default: 0x00000000): 1st pipeline register.*
	0x39	31:0	S	*X_PIPE1 (Default: 0x00000000): 2nd pipeline register.*
	0x3A	31:0	S	*X_PIPE2 (Default: 0x00000000): 3rd pipeline register.*
	0x3B	31:0	S	*X_PIPE3 (Default: 0x00000000): 4th pipeline register.*
	0x3C	31:0	S	*X_PIPE4 (Default: 0x00000000): 5th pipeline register.*
	0x3D	31:0	S	*X_PIPE5 (Default: 0x00000000): 6th pipeline register.*
	0x3E	31:0	S	*X_PIPE6 (Default: 0x00000000): 7th pipeline register.*
	0x3F	31:0	S	*X_PIPE7 (Default: 0x00000000): 8th pipeline register.*

Table 65: Pipeline Register

14.17. Shadow Register

Shadow Register				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x40	31:0	S	*SH_REG0 (Default: 0x00000000) : 1st shadow register.*
	0x41	31:0	U	*SH_REG1 (Default: 0x00000000) : 2nd shadow register.*
	0x42	31:0	U	*SH_REG2 (Default: 0x00000000) : 3rd shadow register.*
	0x43	31:0	U	*SH_REG3 (Default: 0x00000000) : 4th shadow register.*
	0x44	31:0	U	*SH_REG4 (Default: 0x00000000) : 5th shadow register.*
	0x45	31:0	U	*SH_REG5 (Default: 0x00000000) : 6th shadow register.*
	0x46	31:0	U	*SH_REG6 (Default: 0x00000000) : 7th shadow register.*
	0x47	31:0	S/U	*SH_REG7 (Default: 0x00000000) : 8th shadow register.*
	0x48	31:0	U	*SH_REG8 (Default: 0x00000000) : 9th shadow register.*
	0x49	31:0	U	*SH_REG9 (Default: 0x00000000) : 10th shadow register.*
	0x4A	31:0	U	*SH_REG10 (Default: 0x00000000) : 11th shadow register.*
	0x4B	31:0	U	*SH_REG11 (Default: 0x00000000) : 12th shadow register.*
	0x4C	31:0	U	*SH_REG12 (Default: 0x00000000) : 13th shadow register.*
	0x4D	31:0	U	*SH_REG13 (Default: 0x00000000) : 14th shadow register.*

Table 66: Shadow Register

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.18. Reset and Clock Gating Register

Reset and Clock Gating Register				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x4F	2:0	*CLK_GATING_REG (Default: 0x0)*	
			0	Clock gating is not activated.
			7	Clock gating is activated.
		31:8	*RESET_REG (Default: 0x000000)*	
			0	No reset is activated.
			0x525354	Internal reset is activated.

Table 67: Reset and Clock Gating Register

14.19. Encoder Registers

Encoder Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x50	31:0	*ENC_POS (Default: 0x00000000)*	
			S	Actual encoder position [µsteps].
R	0x51	31:0	*ENC_LATCH (Default: 0x00000000)*	
			S	Latched encoder position.
W			*ENC_RESET_VAL(Default: 0x00000000)*	
			S	Defined reset value for *ENC_POS* in case the encoder position must be cleared to another value than 0.
R	0x52	31:0	*ENC_POS_DEV (Default: 0x00000000)*	
			S	Deviation between *XACTUAL* and *ENC_POS*.
W			*CL_TR_TOLERANCE (Default: 0x00000000)*	***(Closed-loop operation)***
			S	Tolerated absolute tolerance between *XACTUAL* and *ENC_POS* to trigger TARGET_REACHED (incl. *TARGET_REACHED_F*lag and event).
W	0x53	31:0	*ENC_POS_DEV_TOL (Default: 0xFFFFFFFF)*	
			U	Maximum tolerated value of *ENC_POS_DEV*, which is not flagged as error.
W	0x54	30:0	*ENC_IN_RES (Default: 0x00000000)*	
			U	Resolution [encoder steps per revolution] of the encoder connected to the encoder inputs.
R			*ENC_CONST (Default: 0x00000000)*	
			U	Encoder constant. **i Value representation: 15 digits and 16 decimal places**
W		31	*manual_enc_const (Default: 0)*	
				0 ENC_CONST will be calculated automatically. **1 Manual definition of ENC_CONST = ENC_IN_RES**

•→ *Continued on next page.*

Encoder Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
W	0x56	15:0		*SER_CLK_IN_HIGH (Default: 0x00A0)*
			U	High voltage level time of serial clock output [# clock cycles].
		31:16		*SER_CLK_IN_LOW (Default: 0x00A0)*
			U	Low voltage level time of serial clock output [# clock cycles].
	0x57	15:0		*SSI_IN_CLK_DELAY (Default: 0x0000)*
			U	**SSI encoder:** Delay time [# clock cycles] between next data transfer after a rising edge of serial clock output. `i` **In case *SSI_IN_CLK_DELAY* = 0:** ***SSI_IN_CLK_DELAY* = *SER_CLK_IN_HIGH***
				SPI encoder: Delay [# clock cycles] at start and end of data transfer between serial clock output and negated chip select. `i` **In case *SSI_IN_CLK_DELAY* = 0:** ***SSI_IN_CLK_DELAY* = *SER_CLK_IN_HIGH***
		31:16		*SSI_IN_WTIME (Default: 0x0F0)*
			U	Delay parameter tw [# clock cycles] between two clock sequences for a multiple data transfer (of the same data). `i` **SSI recommendation: tw < 19 µs.**
	0x58	19:0		*SER_PTIME (Default: 0x00190)*
			U	**SSI and SPI encoder:** Delay time period tp [# clock cycles] between two consecutive clock sequences for new data request. `i` **SSI recommendation: tp > 21 µs.**
	0x7D	15:0		*ENC_COMP_XOFFSET (Default:0x0000)*
			U	Start offset for triangular compensation in horizontal direction. $0 \leq$ *ENC_COMP_XOFFSET* $< 2^{16}$
		23:16		*ENC_COMP_YOFFSET (Default:0x00)*
			S	Start offset for triangular compensation in vertical direction. $-128 \leq$ *ENC_COMP_YOFFSET* ≤ 127
		31:24		*ENC_COMP_AMPL (Default:0x00)*
			U	Maximum amplitude for encoder compensation.

Table 68: Encoder Registers

14.20. PID & Closed-Loop Registers

PID and Closed-Loop Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
RW	0x1C	8:0	*CL_BETA (0x0FF)*	
			U	Maximum commutation angle for closed-loop regulation. **i Set CL_BETA > 255 carefully (esp. if cl_vlimit_en = 1).** **i Exactly 255 is recommended for best performance.**
		23:16	*CL_GAMMA (Default:0xFF)*	
			U	Maximum balancing angle to compensate back-EMF at higher velocities during closed-loop regulation.
RW	0x59	31:0	*CL_OFFSET (Default: 0x00000000)*	***(Closed-loop operation)***
			S	Offset between *ENC_POS* and *XACTUAL* after closed-loop calibration. It is set during closed-loop calibration process. It can be written manually.
W	0x5A	23:0	*PID_P (Default: 0x000000)*	***(PID regulation)***
			U	Parameter P of PID regulator. Proportional term = PID_E · *PID_P* / 256
W			*CL_VMAX_CALC_P (Default: 0x000000)*	***(Closed-loop operation)***
			U	Parameter P of PI regulator controls maximum catch-up velocity limitation.
R		31:0	*PID_VEL (Default: 0x00000000)*	***(PID regulation)***
			S	Actual PID output velocity.
W	0x5B	23:0	*PID_I (Default: 0x000000)*	***(PID regulation)***
			U	Parameter I of PID regulator. Integral term = PID_ISUM / 256 · PID_I / 256
W			*CL_VMAX_CALC_I (Default: 0x000000)*	***(Closed-loop operation)***
			U	Parameter I of PI regulator controls maximum catch-up velocity limitation.
R		31:0	*PID_ISUM_RD (Default: 0x00000000)*	***(PID regulation)***
			S	Actual PID integrator sum. Update frequency = f_{CLK}/128
W	0x5C	23:0	*PID_D (Default: 0x000000)*	***(PID regulation)***
			U	Parameter D of PID regulator. PID_E is sampled with f_{CLK} / 128 / *PID_D_CLKDIV*. Derivative term = (PID_E_{LAST} – PID_E_{ACTUAL}) · *PID_D*
W			*CL_DELTA_P (Default: 0x000000)*	***(Closed-loop operation)***
			U	Gain parameter that is multiplied with the actual position difference in order to calculate the actual commutation angle for position maintenance stiffness. Clipped at *CL_BETA*. Real value = *CL_DELTA_P* / 2^{16} ;Ex: 65536 → 1.0 (gain=1) Value representation: 8 digits and 16 decimal places.
W	0x5D	14:0	*PID_I_CLIP (Default: 0x0000)*	***(PID regulation) (Closed-loop operation)***
			U	Clipping parameter for *PID_ISUM*. Real value = *PID_ISUM* • 2^{16} • *PID_ICLIP*
W		23:16	*PID_D_CLKDIV (Default:0x00)*	***(PID regulation)***
			U	Clock divider for D part calculation.
R		31:0	*PID_E (Default:0x00000000)*	***(PID regulation)***
			S	Actual position deviation.
W	0x5E	30:0	*PID_DV_CLIP* (Default:0x00000000)	***(PID regulation) (Closed-loop operation)***
			U	Clipping parameter for *PID_VEL*.

•→ *Continued on next page.*

W	0x5F	19:0	*PID_TOLERANCE (Default:0x00000)*	***(PID regulation)***
			U	Tolerated position deviation: *PID_E* = 0 in case |*PID_E*| < *PID_TOLERANCE*
W		7:0	*CL_TOLERANCE (Default:0x00)*	***(Closed-loop operation)***
			U	Tolerated position deviation: *CL_DELTA_P* = 65536 (gain=1) in case |*ENC_POS_DEV*| < *CL_TOLERANCE*
W	0x60	23:0	*CL_VMIN_EMF (Default:0x000000)*	
			U	Encoder velocity at which back-EMF compensation starts.
W	0x61	23:0	*CL_VADD_EMF (Default:0x000000)*	
			U	Additional velocity value to calculate the encoder velocity at which back-EMF compensation reaches the maximum angle *CL_GAMMA.*
W	0x62	31:0	*ENC_VEL_ZERO (Default:0xFFFFFF)*	
			U	Delay time [# clock cycles] after the last incremental encoder change to set *V_ENC_MEAN* = 0.
W	0x63	7:0	*ENC_VMEAN_WAIT (Default:0x00)*	***(incremental encoders only)***
			U	Delay period [# clock cycles] between two consecutive actual encoder velocity values that account for calculaton of mean encoder velocity. **! Set *ENC_VMEAN_WAIT* > 32.** **i Is set automatically to *SER_PTIME* for absolute SSI/SPI encoder.**
		7:0	*SER_ENC_VARIATION (Default:0x00)*	***(absolute encoders only)***
			U	Multiplier for maximum permitted serial encoder variation between consecutive absolute encoder requests. **! Maximum permitted value = *ENC_VARIATION* / 256 • 1/8 • *ENC_IN_RES.*** **! If *ENC_VARIATION* = 0: Maximum permitted value = 1/8 • *ENC_IN_RES.***
		11:8	*ENC_VMEAN_FILTER (Default:0x0)*	
			U	Filter exponent to calculate mean encoder velocity.
		31:16	*ENC_VMEAN_INT (Default:0x0000)*	***(incremental encoders only)***
			U	Encoder velocity update time [# clock cycles]. **i Minimum value is set automatically to 256.**
		31:16	*CL_CYCLE (Default:0x0000)*	***(absolute encoders only)***
			U	Closed-loop control cycle [# clock cycles]. **i Is set automatically to *fastest* possible cycle for ABN encoders.**
R	0x65	31:0	*V_ENC (Default:0x00000000)*	
			S	Actual encoder velocity [pps].
	0x66	31:0	*V_ENC_MEAN (Default:0x00000000)*	
			S	Filtered encoder velocity [pps].

Table 69: PID and Closed-Loop Registers

Read entire documentation; especially the "Supplemental Directives" on page 160.

14.21. Transfer Registers

Transfer Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
W	0x68	31:0	*ADDR_TO_ENC (Default:0x00000000)*	***(SPI encoders only)***
			-	Address data permanently sent to get encoder angle data from the SPI encoder slave device.
				Address data sent from TMC4330A to SPI encoder for one-time data transfer.
W	0x69	31:0	*DATA_TO_ENC (Default:0x00000000)*	***(SPI encoders only)***
			-	Configuration data sent from TMC4330A to SPI encoder for one-time data transfer.
R	0x6A	31:0	*ADDR_FROM_ENC (Default:0x00000000)*	***(SPI encoders only)***
			-	Repeated request data is stored here.
				Address data received from SPI encoder as response of the one-time data transfer.
R	0x6B	31:0	*DATA_FROM_ENC (Default:0x00000000)*	***(SPI encoders only)***
			-	Data received from SPI encoder as response of the one-time data transfer.

Table 70: Transfer Registers

14.22. MSLUT Registers

MSLUT Registers				
R/W	**Addr**	**Bit**	**Val**	**Description**
W	0x70	31:0	*MSLUT[0] (Default:0xAAAAB554)*	
	0x71		*MSLUT[1] (Default:0x4A9554AA)*	
	0x72		*MSLUT[2] (Default:0x24492929)*	
	0x73		*MSLUT[3] (Default:0x10104222)*	
	0x74		*MSLUT[4] (Default:0xFBFFFFFF)*	
	0x75		*MSLUT[5] (Default:0xB5BB777D)*	
	0x76		*MSLUT[6] (Default:0x49295556)*	
	0x 77		*MSLUT[7] (Default:0x00404222)*	
			-	**! Each bit defines the difference between consecutive values in the microstep look-up table MSLUT (in combination with *MSLUTSEL*).**
W	0x78	31:0	*MSLUTSEL (Default:0xFFFF8056)*	
			-	Definition of the four segments within each quarter MSLUT wave.
R	0x79	9:0	*MSCNT (Default:0x000)*	
			U	Actual µStep position of the sine value.
R	0x7A	8:0	*USTEPA (Default:0x000)*	
			S	Actual microstep value of PWMA output (sine values).
		24:16	*USTEPB (Default:0x0F7)*	
			S	Actual microstep value of PWMB output (cosine values).
R	0x7B	8:0	*USTEPA_SCALE (Default:0x000)*	
			S	Actual scaled microstep value of PWMA output (sine values).
		24:16	*USTEPB_SCALE (Default:0x0F7)*	
			S	Actual scaled microstep value of PWMB output (cosine values).
W	0x7E	7:0	*START_SIN (Default:0x00)*	
			U	Start value for sine waveform.
		23:16	*START_SIN90 (Default:0xF7)*	
			U	Start value for cosine waveform.

Table 71: MSLUT Registers

14.23. TMC Version Register

Version Register				
R/W	**Addr**	**Bit**	**Val**	**Description**
R	0x7F	15:0	*Version No (Default:0x0002)*	
			U	TMC4330A version number.

Table 72: Version Register

Read entire documentation; especially the "Supplemental Directives" on page 160.

15. Absolute Maximum Ratings

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

Maximum Ratings: 3.3V supply				
Parameter (VCC = 3.3V nominal → TEST_MODE = 0V)	**Symbol**	**Min**	**Max**	**Unit**
Supply voltage	V_{CC}	3.0	3.6	V
Input voltage IO	V_{IN}	−0.3	3.6	V

Table 73: Maximum Ratings: 3.3V supply

Maximum Ratings: 5.0V supply				
Parameter (VCC = 5V nominal → TEST_MODE = 0V)	**Symbol**	**Min**	**Max**	**Unit**
Supply voltage	V_{CC}	4.8	5.2	V
Input voltage IO	V_{IN}	−0.3	5.2	V

Table 74: Maximum Ratings: 5.0V supply

Maximum Ratings: Temperature				
Parameter	**Symbol**	**Min**	**Max**	**Unit**
Temperature	T	−40	125	°C

Table 75: Maximum Ratings: Temperature

16. Electrical Characteristics

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

DC Characteristics						
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Extended temperature range	T_{COM}		−40°C		125	°C
Nominal core voltage	V_{DD}			1.8		V
Nominal IO voltage	V_{DD}			3.3 / 5.0		V
Nominal input voltage	V_{IN}		0.0		3.3 / 5.0	V
Input voltage low level	V_{INL}	V_{DD} = 3.3V / 5V	−0.3		0.8 / 1.2	V
Input voltage high level	V_{INH}	V_{DD} = 3.3V / 5V	2.3 / 3.5		3.6 / 5.2	V
Input with pull-down		V_{IN} = V_{DD}	5	30	110	µA
Input with pull-up		V_{IN} = 0V	−110	−30	−5	µA
Input low current		V_{IN} = 0V	−10		10	µA
Input high current		V_{IN} = V_{DD}	−10		10	µA
Output voltage low level	V_{OUTL}	V_{DD} = 3.3V / 5V			0.4	V
Output voltage high level	V_{OUTH}	V_{DD} = 3.3V / 5V	2.64 / 4.0			V
Output driver strength	I_{OUT_DRV}	V_{DD} = 3.3V / 5V		4.0		mA

Table 76: DC Characteristics

16.1. Power Dissipation

Power Dissipation						
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Static power dissipation	PD_{STAT}	All inputs at VDD or GND V_{DD} = 3.3V / 5V			1.1 / 1.7	mW
Dynamic power dissipation	PD_{DYN}	All inputs at VDD or GND f_{CLK} variable V_{DD} = 3.3V / 5V			2.7 / 4.0	mW / MHz
Total power dissipation	PD	f_{CLK} = 16 MHz V_{DD} = 3.3V / 5V			44.3 / 65.7	mW

Table 77: Power Dissipation

Read entire documentation; especially the "Supplemental Directives" on page 160.

16.2. General IO Timing Parameters

General IO Timing Parameters						
Parameter	**Symbol**	**Conditions**	**Min**	**Typ**	**Max**	**Unit**
Operation frequency	f_{CLK}	f_{CLK} = 1 / t_{CLK}	4.2[1)]	16	30	MHz
Clock Period	t_{CLK}	Rising edge to rising edge	33.5	62.5		ns
Clock time low			16.5			ns
Clock time high			16.5			ns
CLK input signal rise time	t_{RISE_IN}	20 % to 80 %			20	ns
CLK input signal fall time	t_{FALL_IN}	80 % to 20 %			20	ns
Output signal rise time	t_{RISE_OUT}	20 % to 80 % load 32 pF		3.5		ns
Output signal fall time	t_{FALL_OUT}	80 % to 20 % load 32 pF		3.5		ns
Setup time for SPI input signals in synchronous design	t_{SU}	Relative to rising clk edge	5			ns
Hold time	t_{HD}	Relative to rising clk edge	5			ns

Table 78: General IO Timing Parameters

1) The lower limit for f_{CLK} refers to the limits of the internal unit conversion to physical units. The chip will also operate at lower frequencies.

Read entire documentation; especially the "Supplemental Directives" on page 160.

16.3. Layout Examples

16.3.1. Internal Cirucit Diagram for Layout Example

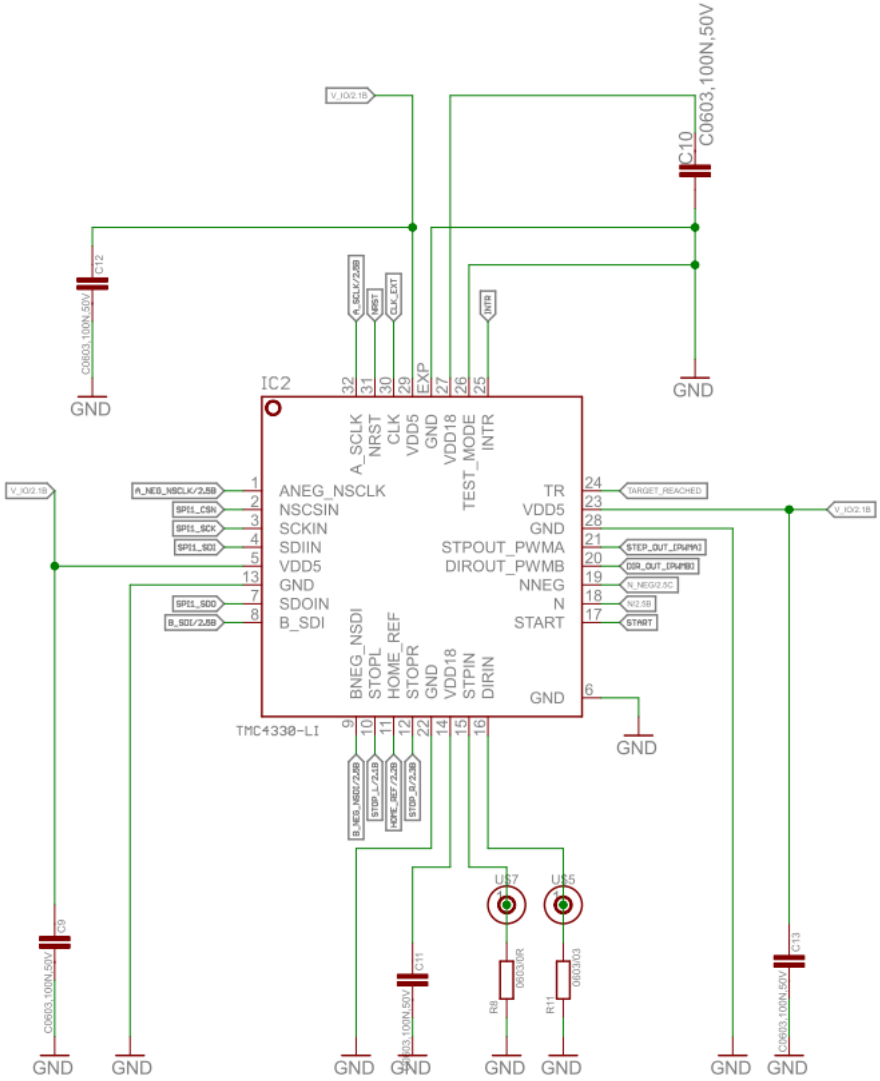


Figure 61: Internal Circuit Diagram for Layout Example

Read entire documentation; especially the "Supplemental Directives" on page 160.

16.3.2. Components Assembly for Application with Encoder

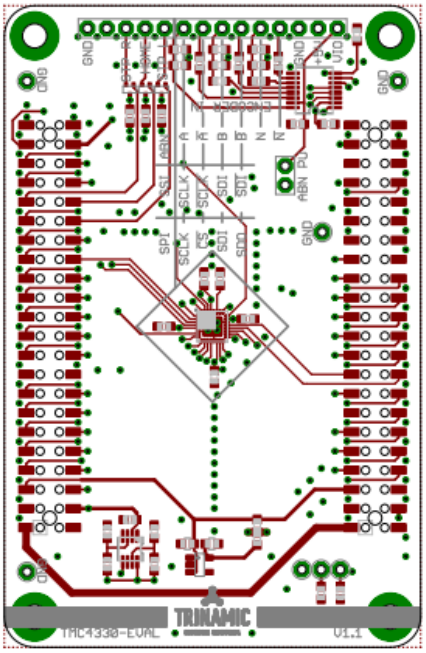


Figure 62: Components Assembly for Application with Encoder

16.3.3. Top Layer: Assembly Side

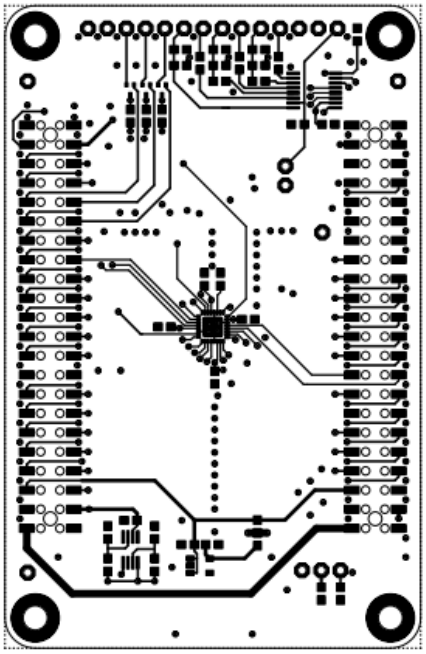


Figure 63: Top Layer: Assembly Side

Read entire documentation; especially the "Supplemental Directives" on page 160.

16.3.4. Inner Layer (GND)

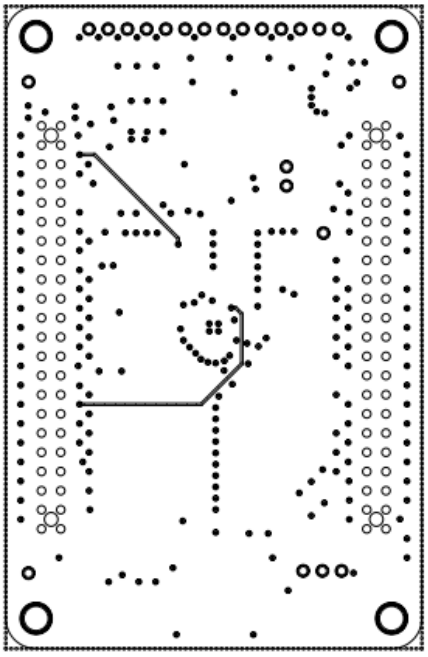


Figure 64: Inner Layer (GND)

16.3.5. Inner Layer (Supply VS)

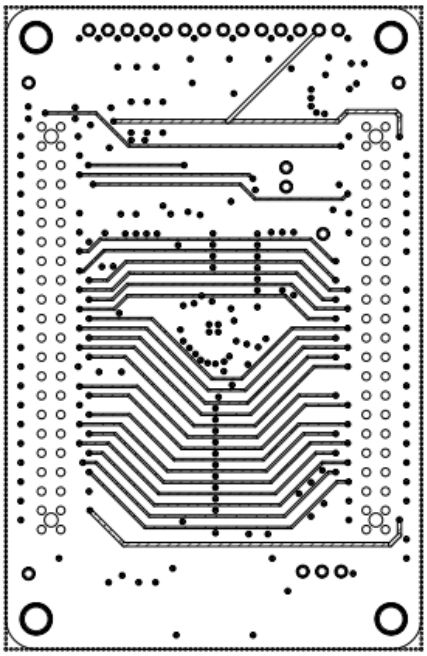
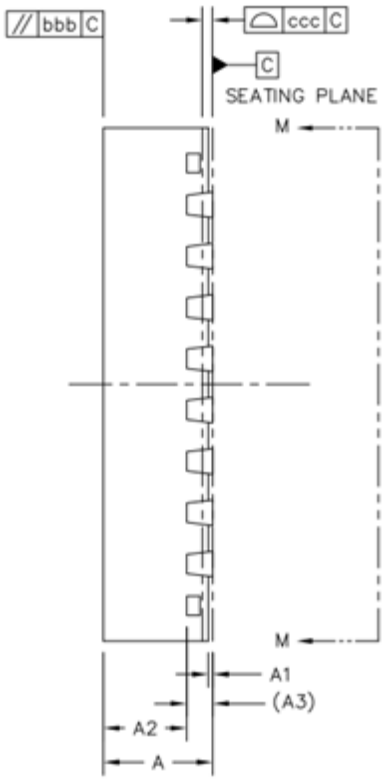


Figure 65: Inner Layer (Supply VS)

Read entire documentation; especially the "Supplemental Directives" on page 160.

16.4. Package Dimensions



Package Dimensions				
Parameter	**Ref**	**Min**	**Nom**	**Max**
Total thickness	A	0.8	0.85	0.9
Stand off	A1	0	0.035	0.05
Mold thickness	A2	-	0.65	0.67
Lead frame thickness	A3	0.203 REF		
Lead width	b	0.15	0.2	0.25
Body size X	D	4 BSC		
Body size Y	E	4 BSC		
Lead pitch	e	0.4 BSC		
Exposed die pad size X	J	2.5	2.6	2.7
Exposed die pad size Y	K	2.5	2.6	2.7
Lead length	L	0.35	0.4	0.45
	L1	0.332	0.382	0.432
Package edge tolerance	aaa	0.1		
Mold flatness	bbb	0.1		
Coplanarity	ccc	0.08		
Lead offset	ddd	0.1		
Exposed pad offset	eee	0.1		

Table 79: Package Dimensions

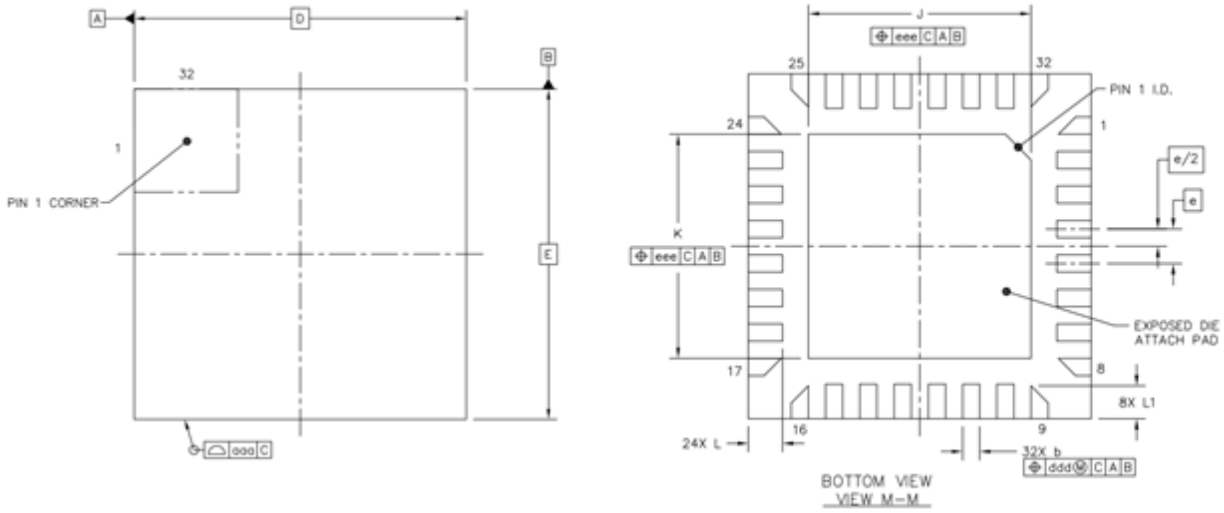


Figure 66: Package Dimensional Drawings

Read entire documentation; especially the "Supplemental Directives" on page 160.

16.5. Package Material Information

Please refer to the associated document "*TMC43xx Package Material Information, V1.00*" for information about available package dimensions and the various tray and reel package options.

This document informs you about outside dimensions per tray and/reel and the number of ICs per tray/reel. It also provides information about available packaging units and their weight, as well as box dimension and weight details for outer packaging.

The document is available for download on the TMC4330A product page at www.trinamic.com.

i Should you require a custom-made component packaging solution or a different outer packaging solution, or have questions pertaining to the component packaging choice, please contact our customer service.

NOTE:

→ *Our trays and reels are JEDEC-compliant.*

16.6. Marking Details provided on Single Chip

The marking on each single chip shows:

❶ Trinamic emblem.

❷ Product code.

❸ Date code.

❹ Location of the copyright holder,

which is TRINAMIC in Hamburg, Germany.

❺ Lot number.

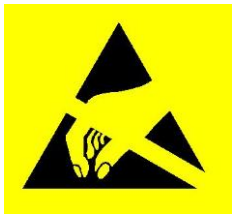
Figure 67: Marking Details on Chip[1]

[1] The image provided is not an accurate rendition of the original product but only serves as illustration.

APPENDICES

17. Supplemental Directives

ESD-DEVICE INSTRUCTIONS



This product is an ESD-sensitive CMOS device. It is sensitive to electrostatic discharge.

- Provide effective grounding to protect personnel and machines.
- Ensure work is performed in a nonstatic environment.
- Use personal ESD control footwear and ESD wrist straps, if necessary.

Failure to do so can result in defects, damages and decreased reliability.

Producer Information

The producer of the product TMC4330A is TRINAMIC GmbH & Co. KG in Hamburg, Germany; hereafter referred to as TRINAMIC. TRINAMIC is the supplier; and in this function provides the product and the production documentation to its customers.

Copyright

TRINAMIC owns the content of this user manual in its entirety, including but not limited to pictures, logos, trademarks, and resources.

Trademark Designations and Symbols

Trademark designations and symbols used in this documentation indicate that a product or feature is owned and registered as trademark and,'or patent either by TRINAMIC or by ather manufacturers, whose products are used or referred to in combination With TRINAMIC's products and TRINAMIC's product documentation. This documentation is a noncommercial publication that seeks to provide concise scientific and technical user information to the target user. Thus, we only enter trademark designations and symbols in the Short Spec of the documentation that introduces the product at a quick glance. We also enter the trademark designation 'symbol when the product or feature name occurs for the first time in the document. All trademarks used are property of their respective owners.

Target User

The documentation provided here, is for programmers and engineers only, who are equipped with the necessary skills and have been trained to work with this type of product.

The ***Target User*** knows how to responsibly make use of this product without causing harm to himself or others, and without causing damage to systems or devices, in which the user incorporates the product.

Disclaimer: Life Support Systems

TRINAMIC Motion Control GmbH & Co. KG does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Motion Control GmbH & Co. KG.

Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

Information given in this document is believed to be accurate and reliable. However, no responsibility is assumed for the consequences of its use nor for any infringement

of patents or other rights of third parties which may result from its use. Specifications are subject to change without notice.

Disclaimer: Intended Use

The data specified in this user manual is intended solely for the purpose of product description. No representations or warranties, either express or implied, of merchantability, fitness for a particular purpose or of any other nature are made hereunder with respect to information/specification or the products to which information refers and no guarantee with respect to compliance to the intended use is given.

In particular, this also applies to the stated possible applications or areas of applications of the product. TRINAMIC products are not designed for and must not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (Safety-Critical Applications) without TRINAMIC's specific written consent.

TRINAMIC products are not designed nor intended for use in military or aerospace applications or environments or in automotive applications unless specifically designated for such use by TRINAMIC. TRINAMIC conveys no patent, copyright, mask work right or other trade mark right to this product. TRINAMIC assumes no liability for any patent and/or other trade mark rights of a third party resulting from processing or handling of the product and/or any other use of the product.

Product Documentation Details

This document ***Datasheet User Manual*** contains the ***User Information*** for the ***Target User***.

The ***Short Spec*** forms the preface of the document and is aimed at providing a general product overview. The Main Manual contains detailed product information pertaining to functions, and configuration settings. It contains all other pages of this document.

Collateral Documents & Tools

This product documentation is related and/or associated with additional tool kits, firmware and other items, as provided on the product page at: www.trinamic.com .

Read entire documentation; especially the "Supplemental Directives" on page 160.

18. Tables Index

Read entire documentation; especially the "Supplemental Directives" on page 160.

19. Figures Index

20. Revision History

Document Revision History			
Version	**Date**	**Author**	**Description**
1.00	2016-NOV-25	HS	First complete version.

Table 80: Document Revision History

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for trinamic ***manufacturer:***

Other Similar products are found below :

TMC2041-EVAL-KIT TMCM-3213 TMCM-343-H-TMCL PD86-3-1180-CANOPEN QSH6018-65-28-210 TMCS-40-KIT QSH8618-95-55-700 TMC-UPS-10A70V-EVAL TMC-UPS-2A24V-EVAL TMC5031-EVAL-KIT PD42-2-1240-TMCL TMCM-1021 TMC424 PD57-2-1060-TMCL TMCM-3230-CANOPEN TMCM-3312-TMCL TMCM-3313 TMC4330-EVAL PD-1141-CABLE PD42-1-1070 PD42-2-1141 PD57-1-1160-TMCL PD60-3-1160-TMCL PD42-3-1241-CANOPEN TMCM-1241-CANOPEN PD57-1-1276-CANOPEN PD57-2-1276-CANOPEN PD57-2-1276-TMCL TMCM-1636-48V-TMCL QSH8618-96-55-700 TMCM-6214-TMCL QSH5718-51-28-101 QSH5718-76-28-189 TMCM-3212-CANOPEN TMCM-3214-CANOPEN TMCM-3214-TMCL TMCM-3215 TMCM-3312-CANOPEN TMCM-6210-CANOPEN TMCM-6210-TMCL TMCM-6212-CANOPEN TMCM-6213 PD42-1-1270-CANOPEN PD42-3-1140-TMCL PD42-3-1141 TMC261-BOB TMC8670-EVAL PD57-1-1276-TMCL PD60-3-1276-TMCL TMC-SCHRAUBSTOCK