TMC454 – DATA SHEET

- Software Compatible Successor of the TMC453 -
Single Axis Controller with Incremental Encoder Interface

TRINAMIC® Motion Control GmbH & Co. KG
Sternstraße 67
D – 20357 Hamburg
GERMANY

www.trinamic.com

1 Features

The TMC454 is a high-end motor controller for all common 2-, 3-, and 5-phase stepper motors. It provides encoder feedback for high reliability or high position resolution drives. A large set of motion control features is included on this IC. The various function blocks are hardwired and ready-to-use. All processes can be serialized using the instruction based interface. A complete linear & S-shaped ramp generation unit is integrated within the TMC454. All dynamic parameters can be adjusted for a given application in a very broad range. The CPU is relieved from all time critical tasks like ramp calculation or micro-step generation. Thus CPU and software engineer can focus on higher level motion control jobs. In many applications, the TMC454 reduces the time time-to-market, and it saves costs for software development and hardware engineering. The TMC454 turns stepper motors into easy to use peripherals.

- TMC454 software compatible - no need to change your existing TMC454 firmware
- parallel interface
- serial IIC bus interface
- RoHS conform compact 144 pin (1 mm) fine pitch BGA package of size 11 mm x 11 mm
- controls 2-,3-,and 5-phase stepper motors
- full step, half step, sine-step
- pulse generation from mHz to MHz
- S-shaped and linear ramp generation
- PID controller for holding the position
- relieves CPU from all time critical tasks
- synchronization of multiple TMC454
- incremental encoder interface (ABN)
- interrupt controller
- digital interface for external LTC1665 8-bit DACs (for emulation of former integrated TMC453 DACs)
- direct interface to TRINAMIC stepper motor drivers TMC236, TMC239, TMC246, TMC249
- StallGuard™ output when using TMC246 or TMC249 with sensorless stall detection
- interface for LTC1665 external low cost DAC for classical analog power stage control
- direct interface for TMC236 / TMC239 / TMC246 / TMC249
- StallGuard™ output (active low for reference input) when using TMC246 / TMC249
- PWM output for stepper motor driver current scaling (with an ext. RC low pass)
- power supply +3.3V IO and +1.5V core / low power operation: only 20 mA @ 16 MHz (typ.)
- operation with 3.3V CMOS compatible IOs
- control signal for 5V/3.3V level shifter (e.g. SN74LVC245)
- half clock output for micro controller (e.g. for AVR µC running @ 3.3V)
- commercial temperature range (0° ... +70°C)

Order Code : TMC454-BC

Life support policy

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.

Table of Contents

Table of Figures

Table of Tables

1.1 Functional Overview

General Features

- Seven stage command FIFO relieves host processor from all real-time requirements
- Step pulse generation from millihertz to megahertz
- Three 8 Bit DACs for direct micro step control of 2- and 3-phase stepper motors
- Optional pulse- and direction interface
- 8 Bit parallel interface / serial 2-wire interface
- Low Power 1.5V CMOS process with 3.3V inputs and outputs
- Direct interfacing to industry standard power drivers
- Package: 144 pin 1 mm fine pitch BGA of size 11 mm x 11 mm
- Temperature range 0°C ... +70°C

Ramp Generation

- Automatic generation of S-shaped ramps
- Synthesis of ramps of any shape with constant, linear and parabolic segments
- Synchronization between multiple TMC454 motion controllers
- Programmable interrupt events
- Control inputs for stop and slowdown

Sequencer

- Freely programmable half step, full step and micro step patterns
- Supports 2-, 3- ,4- and 5-phase stepper motors with unipolar or bipolar control
- Sine generator for up to 256 micro steps per full step
- Intelligent motor current control
- 128X8 RAM for user defined micro steps adapted to the motor characteristics
- Direct output of the velocity value for servo motor control

Incremental Encoder

- Supports 2-phase incremental encoders for position control and feedback control loop

Feedback controlled motion

- Stabilization against varying motor loads
- Exact position control via incremental encoder

2 Introduction

2.1 Control of Stepper Motors

Stepper motors are historically used in applications, where a positioning to preprogrammed or calculated positions is needed. Examples are linear and rotational axes in robots. The reason for using stepper motors in these applications is that they work extremely precise without the necessity to use a control loop. Precautions have to be taken to avoid overloading the motor, e.g. by too fast movements or too high accelerations. In many of these applications an incremental encoder is coupled mechanically to the motor to measure the position or velocity or to detect failures.
New applications for stepper motors have the demand for a high reliability while reducing costs, e.g. the minimization of mechanical parts, which is possible because the stepper motor provides high torque without gear and precise positioning without feedback. It is expected that stepper motors and electronically commutated motors will replace DC motors in many applications which incorporate DC motors today. The reduction in cost is possible because control electronics continues to get less expensive while costs of mechanic parts cannot be reduced in such a dramatic way.

The TMC454 is a universal controller for stepper motors. It interfaces directly to a CPU and offloads the CPU from all time critical tasks.
After setting up the chip with a number of control parameters, the motor can be controlled by simply programming the target position for a movement. The TMC454 automatically drives the motor with smooth movement curves (ramps) to reduce mechanical stress and to avoid the loss of steps. Where necessary the CPU can control all phases of the movement itself.

2.2 Microstepping

A stepper motor has an inherent resolution given by the number of fullsteps per rotation. These positions are achieved by switching the coils on and off. In applications where a higher step resolution than the inherent step resolution or its double, the half step resolution, is required, a control scheme called microstepping can be used. It allows a positioning of the motor between the full (or half) steps. Therefore the coils of the motor are driven by weighed currents. In principle, a sine-wave driving scheme would be sufficient, but since the characteristics of most motors are not linear, it is desirable to be able to program the current patterns for a number of microsteps between each two full steps. Using micro stepping can also be necessary to reduce the abruptness of the change between two full-/half step positions.

A stepper motor is very sensitive to static and changing loads in micro step positions. In cases where micro stepping is used to improve the accuracy of positioning, a feedback control system using a position encoder reduces the need for adjustment of the micro stepping table and minimizes the effect of varying loads. The TMC454 contains a programmable filter to build a PID-(Proportional-Integral-Differential) regulator for position stabilization.

2.3 More precision using motor current control

In most of today's applications based on stepper motors, the motors are driven with the full current, independent of the torque actually needed. This not only wastes energy, reduces lifetime of electronic and mechanical parts, but also reduces precision because of the thermal expansion of the mechanical parts.

Since the torque of a stepper motor is a function of the coil current, the required current depends on load and acceleration. The TMC454 supports an effective means to reduce power consumption: It monitors acceleration and speed to adjust the motor current according to a user defined table. TMC454 users have reported a dramatic reduction of thermal problems.

2.4 Conclusion

The TMC454 is an efficient and inexpensive motor controller. It integrates the complete control function set to drive all kinds of stepper motors and directly interfaces to power drivers. Further on it integrates interfaces and the logic to control DC servo motors. Programming of the TMC454 is easy, because it integrates all time critical features in hardware. Only a few parameters have to be programmed to adapt the TMC454 to a given application. Then the TMC454 does all the positioning by just programming the desired target position and motor velocity. The TMC454 provides the ultimate function set to get the highest possible resolution without the necessity for feedback while also integrating the complete logic for feedback control without processor overhead.

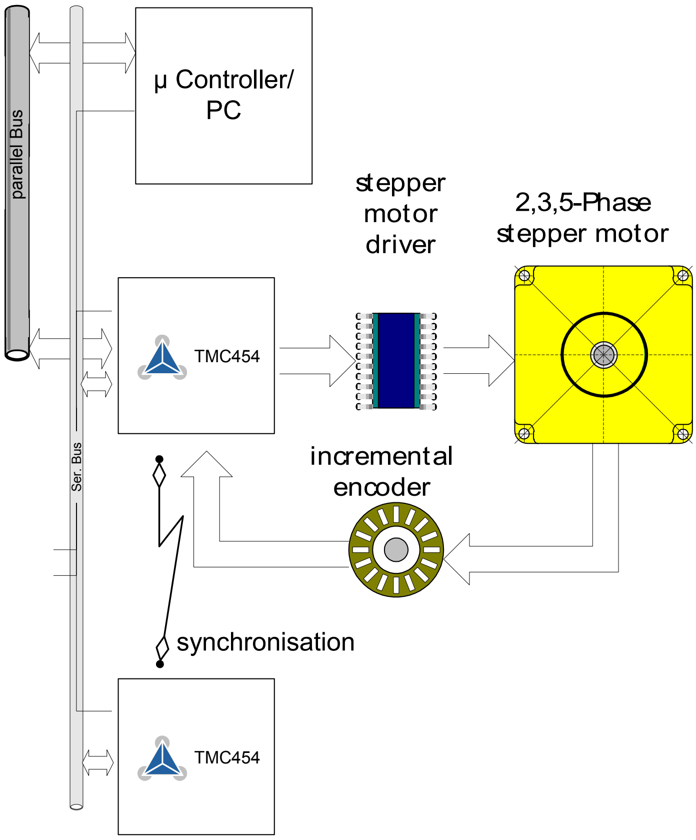


Figure 2.1: System environment of the TMC454

Depending on the requirements the TMC454 can be programmed via a serial or a parallel interface. It provides an efficient synchronization mechanism to control multiple motors synchronously.

This large set of features is controlled via 84 integrated registers. While this may seem to be a large number of function, control and status registers, only 4 registers have to be programmed to start an automatic ramp function.

3 Block diagram

The TMC454 is built in a modular way. Every module realizes a defined set of functions. Only the modules which are needed for an application have to be programmed.

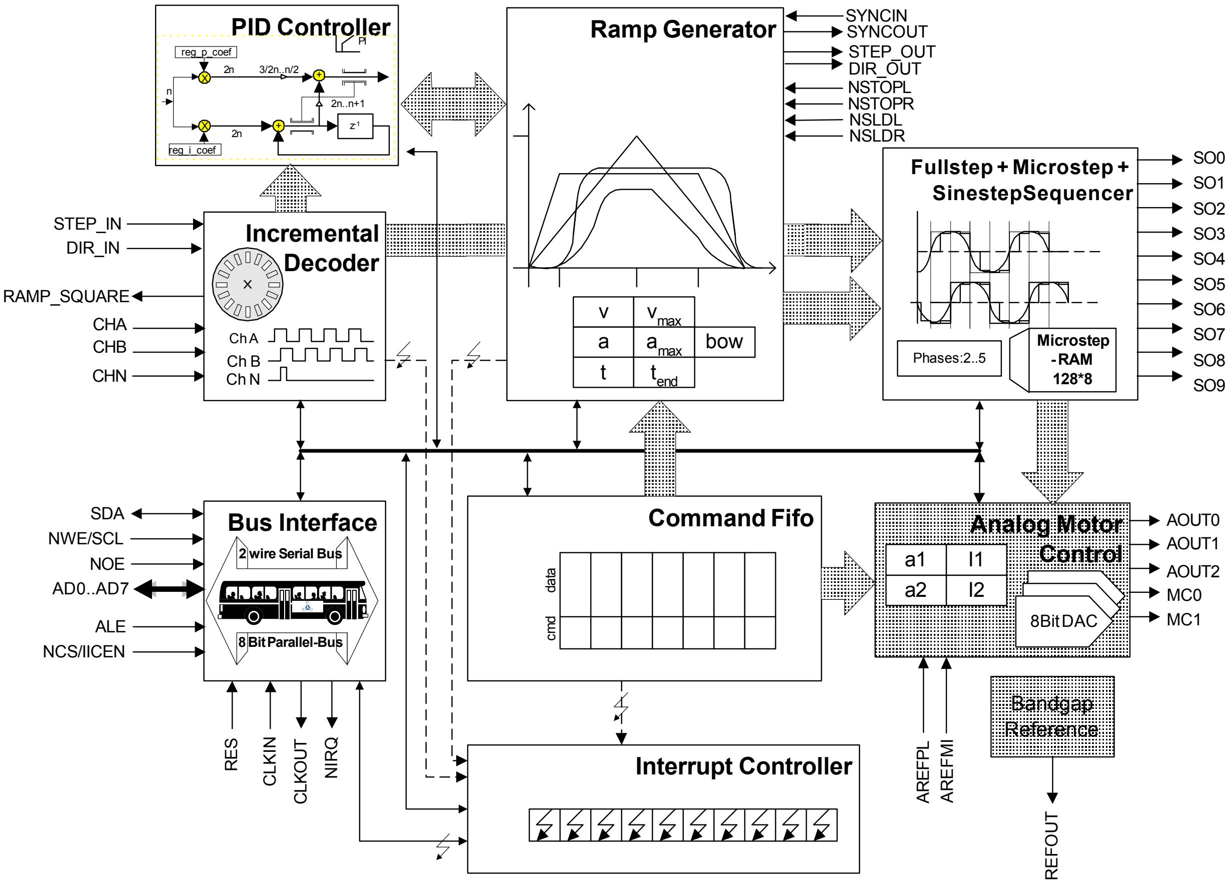


Figure 3.1: Block diagram of the TMC454's modules

4 Pinning, Package, and Electrical data of the TMC454

4.1 Pinning of TMC454

The TMC454 is available within a 144 ball fine pitch (1 mm) BGA package.

Important Hint:

All pins specified as (n.c. = not connect) *must* be left unconnected (open). All power supply pins (named V33 for +3.3V and named V15 for +1.5V) and all ground pins *must* be connected.

All inputs and all outputs have 3.3V CMOS level. Level shifter are required when using 5V devices (e.g. micro controller with 5V IO, incremental encoder with 5V ABN outputs, …)

#	BGA Ball	Signal Name	In/Out	Description / Comment
1	**A1**	**GND**		**ground**
2	**A2**	**V33**		**+3.3V supply voltage**
3	A3	NCS	I	low active chip select input
4	A4	STEP_IN	I	ext. step Input: Pulse (edge triggered, min. pulse length 1 clock)
5	A5	DIR_IN	I	ext. step Input: direction input
6	**A6**	**GND**		**ground**
7	A7	NSTOPL	I	End Switch Left
8	**A8**	**V15**		**+1.5V supply voltage**
9	A9	TMCDRV_SHAFT	I	TRINAMIC stepper motor driver interface, shaft input for fixed selection of direction of motion, this is to set up compatibility with an existing application based on the TMC453 if the motor turns into the opposite direction with the TMC454 together with a different driver configuration (do not use it as a "direction" input") – internal pull-down R
10	A10	CHB	I	channel B of incremental encoder
11	A11	CHA	I	channel A of incremental encoder
12	**A12**	**GND**		**ground**
13	B1	CLKIN	I	system clock input, up to 16 MHz
14	**B2**	**GND**		**ground**
15	B3	NCS_OUT	O	low active chip select output for nRD and nWR micro controller interface (instead of nCS with nWR/RD)
16	B4	STEP_OUT	O	step pulse output (1 clock high on change of position)
17	B5	DIR_OUT	O	direction output
18	B6	NSLDL	I	slowdown switch left
19	B7	NSTOPR	I	end switch right
20	**B8**	**GND**		
21	B9	TMCDRV_ERROR	O	TRINAMIC internal driver diagnosis signals, logical or of driver diagnosis signals
22	B10	CHN	I	null-signal (N) of incremental encoder
23	**B11**	**GND**		**ground**
24	**B12**	**V33**		**+3.3V supply voltage**
25	C1	AD[0]	I / O	AD[0] of bi-directional address/data bus AD[7] … AD[0]
26	C2	AD[1]	I / O	AD[1] of bi-directional address/data bus AD[7] … AD[0]
27	C3	CLKDIV2OUT	O	half clock frequency clock output, e.g. for micro controller with 8 MHz when TMC454 running with 16 MHz

28	C4	V15		+1.5V
29	C5	RAMP_SQUARE	O	symmetrical step pulse output
30	C6	NSLDR	I	slowdown switch right
31	C7	n.c. (open)	n.c.	
32	C8	n.c. (open)	n.c.	
33	C9	n.c. (open)	n.c.	
34	C10	n.c. (open)	n.c.	
35	C11	STO[1]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[1] of STO[0] … STO[9])
36	C12	STO[0]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[1] of STO[0] … STO[9])
37	D1	AD[2]	I /O	AD[2] of bi-directional address/data bus AD[7] … AD[0]
38	D2	AD[3]	I /O	AD[3] of bi-directional address/data bus AD[7] … AD[0]
39	D3	n.c. (open)	n.c.	
40	D4	SYNCOUT	O	synchronization output for command FIFO
41	D5	SYNCIN	I	synchronization input for command FIFO
42	D6	n.c. (open)	n.c.	
43	D7	n.c. (open)	n.c.	
44	D8	n.c. (open)	n.c.	
45	D9	n.c. (open)	n.c.	
46	D10	n.c. (open)	n.c.	
47	D11	STO[3]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[3] of STO[0] … STO[9])
48	D12	STO[2]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[2] of STO[0] … STO[9])
49	E1	V15		**+1.5V supply voltage**
50	E2	AD[4]	I /O	AD[4] of bi-directional address/data bus AD[7] … AD[0]
51	E3	NRES	I	reset input (active low), internal pull-up R. **Important Hint**: Due to internal synchronisation, the active low reset (NRES) needs to be low for one rising edge of the clock (CLKIN). In other words, the clock needs to run before the NRES changes from active low to inactive high
52	E4	V33		**+3.3V supply voltage**
53	E5	n.c. (open)	n.c.	
54	E6	V33		**+3.3V supply voltage**
55	E7	V33		**+3.3V supply voltage**
56	E8	n.c. (open)	n.c.	
57	E9	V33		**+3.3V supply voltage**
58	E10	V15		**+1.5V supply voltage**
59	E11	STO[5]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[5] of STO[0] … STO[9])
60	E12	STO[4]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[4] of STO[0] … STO[9])
61	F1	AD[5]	I / O	AD[5] of bi-directional address/data bus AD[7] … AD[0]
62	F2	GND		**ground**
63	F3	n.c. (open)	n.c.	
64	F4	GND	n.c.	**ground**

65	**F5**	**GND**		**ground**
66	**F6**	**GND**		**ground**
67	**F7**	**GND**		**ground**
68	F8	n.c. (open)	n.c.	
69	F9	n.c. (open)	n.c.	
70	**F10**	**GND**		**ground**
71	F11	STO[7]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[7] of STO[0] … STO[9])
72	F12	STO[6]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[6] of STO[0] … STO[9])
73	G1	AD[6]	I / O	AD[6] of bi-directional address/data bus AD[7] … AD[0]
74	**G2**	**GND**		**ground**
75	**G3**	n.c. (open)	n.c.	
76	G4	n.c. (open)	n.c.	
77	**G5**	**GND**		**ground**
78	**G6**	**GND**		**ground**
79	**G7**	**GND**		**ground**
80	G8	n.c. (open)	n.c.	
81	G9	n.c. (open)	n.c.	
82	G10	n.c. (open)	n.c.	
83	G11	STO[9]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[9] of STO[0] … STO[9])
84	G12	STO[8]	O	Digital Motor Control Outputs for full-/half step patterns sine waves and velocity values (STO[8] of STO[0] … STO[9])
85	**H1**	**V15**		**+1.5V supply voltage**
86	H2	AD[7]	I / O	AD[7] of bi-directional address/data bus AD[7] … AD[0]
87	H3	n.c. (open)		
88	H4	SERIAL_EN	I	serial enable (power-on reset-option)
89	**H5**	**V15**		**+1.5V supply voltage**
90	H6	n.c. (open)		
91	H7	n.c. (open)		
92	H8	n.c. (open)		
93	H9	n.c. (open)		
94	**H10**	**V33**		**+3.3V supply voltage**
95	H11	TMCDRV_OTPW	O	TRINAMIC driver diagnosis “over temperature warning”
96	**H12**	**V15**		**+1.5V supply voltage**
97	J1	ALE	O	address latch enable (1=address valid)
98	J2	SDA_OD	I/O	serial data (3.3V open drain output)
99	**J3**	**V33**		**+3.3V supply voltage**
100	J4	n.c. (open)	n.c.	
101	J5	DAC2_LSB[3]	O	DAC2 bit # 3 digital output
102	J6	DAC2_LSB[2]	O	DAC2 bit # 2 digital output
103	**J7**	**V15**		**+1.5V supply voltage**
104	J8	internal (TCK)	I	1K pull down R required (can be left open with VJTAG tied to GND)
105	J9	DAC2_LSB[1]	O	DAC2 bit # 1 digital output
106	J10	n.c. (open) / (TDO)	n.c. / (O)	(can be left open with VJTAG tied to GND)
107	J11	MC1	O	digital control of Motor Current (two signals MC0, MC1)
108	J12	MC0	O	digital control of Motor Current (two signals MC0, MC1)

109	K1	NOE	I	output enable (active low)
110	K2	SDA_NOE	O	serial data enable (active low, 0 = open drain active)
111	K3	ADDIR	O	ADDIR to DIR of 74LVC4245 with A @ 5V side (µC) / B @ 3v3 side (TMC454)
112	K4	PCM_DAC2	O	PCM_DAC2, not available (reserved for later version)
113	K5	PCM_DAC1	O	PCM_DAC1, not available (reserved for later version)
114	K6	PCM_DAC0	O	PCM_DAC0, not available (reserved for later version)
115	**K7**	**GND**		**ground**
116	K8	LTC1665_NCLR	O	connect to nCLR input of LTC1665 DAC
117	K9	DAC2_LSB[0]	O	DAC2 bit # 0 digital output
118	**K10**	**GND**		**ground**
119	K11	TMCDRV_nSTALL	O	Switch Mixed Decay on (when using TRINAMIC driver), this signal can be controller by the MC signals
120	K12	TMCDRV_MD_EN	I	Mixed Decay Enable control signal for TRINAMIC driver, internal pull-down R
121	**L1**	**GND**		**ground**
122	**L2**	**V33**		**+3.3V supply voltage**
123	L3	ADNOE	O	ADNOE to nOE of 3.3V / 5V level shifter
124	L4	PWM_DAC2	O	PWM output (DAC2 with R-C pass voltage divider)
125	**L5**	**V33**		**+3.3V supply voltage**
126	L6	TMCDRV_SDO	I	connect to SDO output of TRINAMIC driver
127	L7	LTC1665_NCS_012	O	connect to nCS of external LTC1665 DACs
128	L8	LTC1665_DIN_01	O	connect to DIN of external LTC1665 DAC 01
129	L9	n.c. (open) / (TMS)	(I)	(can be left open with VJTAG tied to GND)
130	**L10**	**V33 / (VJTAG dis.)**		**+3.3V (VJTAG supply) / GND (ground to disable JTAG)**
131	**L11**	**V33**		**+3.3V supply voltage**
132	L12	reserved (TRST)	I	1K pull down R required **(ground to disable JTAG)**
133	**M1**	**GND**		**ground**
134	M2	INT	O	interrupt output (polarity programmable)
135	M3	NWE_SCL	I	write enable (active low) / serial clock SCL (for IIC operation)
136	M4	TMCDRV_CSN	O	connect to CSN input of TRINAMIC driver
137	M5	TMCDRV_SCK	O	connect to SCK input of TRINAMIC driver
138	M6	TMCDRV_SDI	O	connect to SDI input of TRINAMIC driver
139	M7	LTC1665_SCK_012	O	connect to SCK input(s) of LTC1665 DACs
140	M8	LTC1665_DIN_2	O	connect to DIN of LTC1665 DAC that emulates DAC2
141	M9	n.c. (open) / (TDI)	n.c.	(can be left open with VJTAG tied to GND)
142	**M10**	**V33**		**+3.3V supply voltage**
143	M11		I	1K pull-down R required
144	**M12**	**GND**		**ground**

Table 4.1 : TMC454-BC pinout

4.1.1 Pull-Up / Pull-Down Resistances

Some inputs have weak on-chip pull-up resp. weak pull-down resistors. The resistance of these is within the range of 10 kOhm (min.) … 45 kOhm (max.)

4.1.2 Blocking Capacitors

Ceramic capacitors of 100 nF capacitance should be connected as close as possible at each 1V5 power supply pin and at each 3V3 power supply pin. Alternatively, two power planes – one power plane for 1V5 and one power plane for 3V3 can be used.

4.2 Package Outlines and Dimensions

4.2.1 Fine Pitch BGA Package with 144 Balls (FBGA144) of TMC454-BC

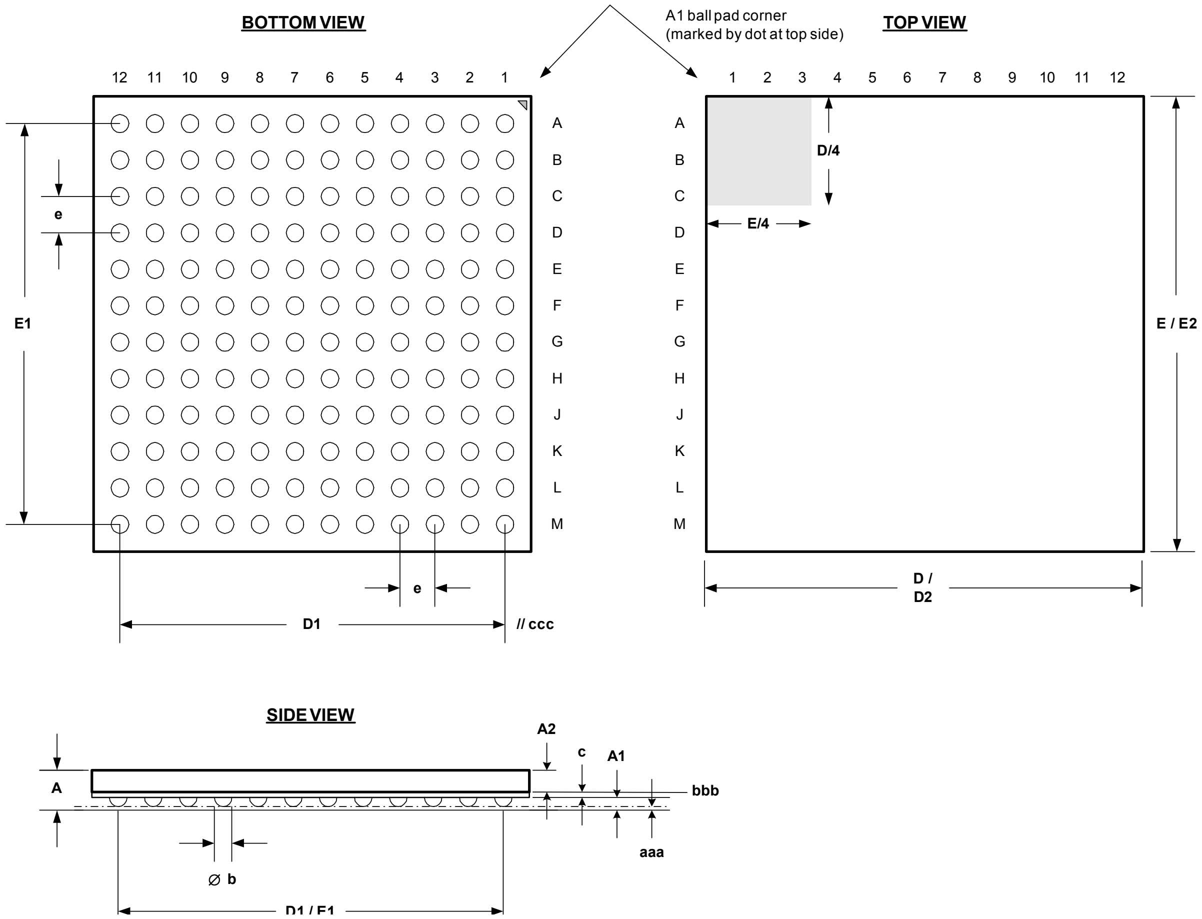


Figure 4.1 – Package Outline Drawing FBGA144 – MO-192 VAR DAD-1

Symbol	Dimensions in MILLIMETERS			*Dimensions in INCHES*		
	Min	Typ	Max	*Min*	*Typ*	*Max*
A	1.35	1.45	1.55			
A1	0.35	0.40	0.45			
A2	0.65	0.70	0.75			
aaa		0.12				
b	0.45	0.50	0.55			
bbb		0.25				
c	-	0.35	-			
ccc		0.35				
D	12.80	13.00	13.20			
D1		11.00 BSC				
D2	12.80	13.00	13.20			
E	12.80	13.00	13.20			
E1		11.00 BSC				
E2	12.80	13.00	13.20			
e		1.00				

Table 4.2 : Dimensions of FBGA144 (Note: BSC = Basis Spacing Between Centers)

4.3 Absolute Maximum Ratings

The maximum ratings may not be exceeded under any circumstances. Permanent operating close to the absolute maximum ratings can cause permanent damage to the device. Please refer to Table 4.3 for permanent operation conditions.

DC Supply IO Voltage V33	-0.3V ≤ V33 ≤ 3.75V
DC Supply Core Voltage V15	-0.3V ≤ V15 ≤ 1.65V
DC Voltage on any Pin	VSS - 0.3V ≤ Vin ≤ V33 + 0.3V
Output Current	≤ 12 mA
ESD Voltage on any pin HBM (Human Body Model)	2000V
ESD Voltage on any pin MM (Machine Model)	250V
ESD Voltage CDM (Charge Device Model)	200V
Max. Junction Temperature	≤ 125°C
Storage Temperature	-65°C ≤ 150°C

	Min	Typ	Max	Units
Digital IO Supply Voltage V33	3.0	3.3	3.6	V
Digital Core Supply Voltage V15	1.425	1.5	1.575	V
$T_{ambient}$ (commercial range)	0	+25	+70	°C
Operating Frequency	0		16	MHz
V33 Supply Current @ f=16 MHz (depends on static and dynamic IO load)		10		mA
V15 Supply Current @ f=16 MHz		10		mA

Table 4.3 : Recommended Operating Conditions / Typical Characteristics

Hint: Linear regulators (e.g. LD1117, ST Microelectronics) are recommended for 3.3V and 1.5V from a 5V supply). Only one additional linear regulator (LD1117, ST Microelectronics) is required for the 1.5V supply if a 3.3V supply is already available.

4.4 Analog Functions - External Low Cost DACs

In contrast to the discontinued TMC453 motion controller, the TMC454 is not equipped with analog parts as integrated DACs, buffered by operational amplifiers or reference voltage source. A low cost DAC can be realized based on the digital PWM_DAC2 output signal together with a RC voltage divider low pass filter. Alternatively, LTC1665 DACs can be directly connected to the TMC454 for emulation of analog functionality of the TMC453.

Parameter	Description	Max	Units
fUPDT_DAC	number of TMC454 clock cycles for DAC data update, the DACs are updated with a frequency of fCLKIN[Hz] / 71	71	clock cycles
fUPDT_DAC_8	update frequency at 8 MHz clock frequency of TMC454	112.5	kHz
fUPDT_DAC_16	update frequency at 16 MHz clock frequency of TMC454	225	kHz

Table 4.4 : Update Frequency of DACs

4.5 Using TRINAMIC Driver (TMC236, TMC239, TMC246, TMC249) via SPI

Hint:

TMCDRV_SDI is an output and has to be connected with the pin named SDI of the TRINAMIC driver (TMC236, TMC239, TMC246, TMC249). TMCDRV_SDO is an input and has to be connected with the output pin named SDO of the TRINAMIC driver.

Please refer the TRINAMIC stepper motor driver datasheets for a detailed description of the control signals resp. control bits.

4.5.1 Control Signals

There are two control signals for the TRINAMIC drivers.

4.5.1.1 TMCDRV_MD_EN

The signal named TMCDRV_MD_EN controls the Mixed Decay Feature of the TRINAMIC stepper motor drivers TMC236, TMC239, TMC246, TMC249.

4.5.1.2 TMCDRV_SHAFT

The shaft signal is intended to be used for changing the direction of revolution of a stepper motor. The shaft signal is not intended to be used as a direction signal. It is intended to be used for compatibility with existing applications based on the TMC453 with an other driver configuration.

4.5.2 Diagnosis Signals

There are three diagnosis signals directly available.

4.5.2.1 TMCDRV_OTPW

The TMCDRV_OTPW is the “over Temperature Pre-Warning” signal sent back from the TRINAMIC driver via SPI.

4.5.2.2 TMCDRV_ERROR

The TMCDRV_ERROR is the logical of error condition signals (short, open load, over temperature)

4.5.3 Using StallGuard™ with TMC246 or TMC249

The TMC454 is equipped with a low active StallGuard™ output named TMCDRV_nSTALL. The output is low active to used it to drive directly TMC454 reference switch inputs. The StallGuard™ threshold is set by the three LSB of the DAC2. These three LSB are mapped internally to the TRINAMIC stepper motor SPI interface. With this, the DAC2 can still be used for analog current scaling.

4.5.4 PWM DAC for Current Scaling

Analog current scaling can be done by using DAC2 without an additional external DAC. A simple resistor voltage divider with a capacitor driven by the PWM_DAC2 output of the TMC454. The internal resistance of the both INA and INB analog control inputs of the TRINAMIC stepper motor drivers have to be taken into account. Please refer to the TMC236 / TMC239 / TMC246 / TMC249 data sheets for details.

The PWM clock frequency fPWM is fCLKIN / 256. So, for a clock frequency fCLKIN = 16 MHz one gets a PWM frequency fPWM = 16 MHz / 256 = 62.5 kHz for the PWM_DAC2 output.

Parameter	Description	Max	Units
fUPDT_TMC	number of TMC454 clock cycles for TRINAMIC stepper motor driver data update, the drivers data are updated with a frequency of fCLKIN[Hz] / 52	52	clock cycles
fUPDT_TMC_8	update frequency at 8 MHz clock frequency of TMC454	154	kHz
fUPDT_TMC_16	update frequency at 16 MHz clock frequency of TMC454	308	kHz

Table 4.5 : Update Frequency of TRINAMIC Drivers

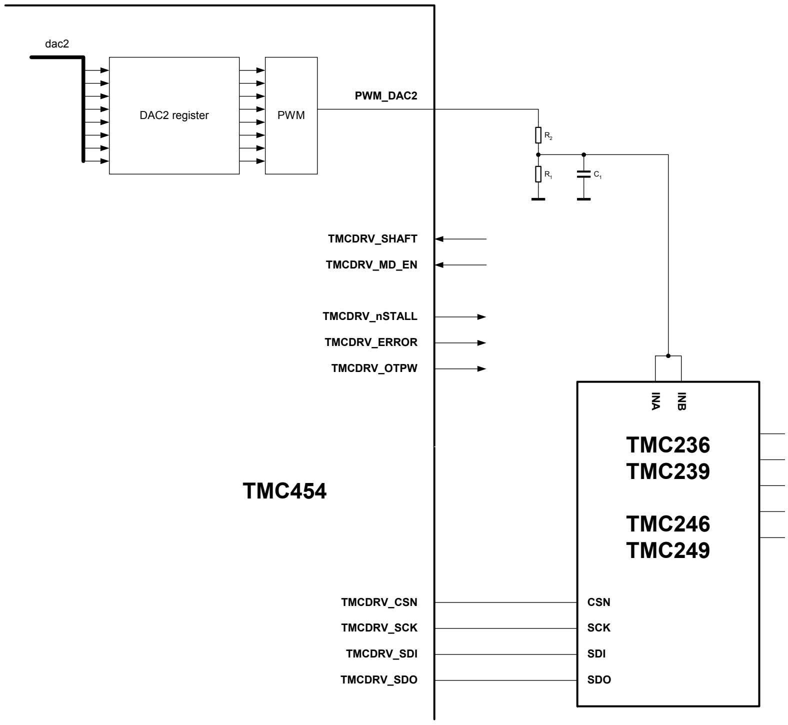


Figure 4.2 : Connection of a TRINAMIC driver with the TMC454

4.6 External DACs - LTC1665

The LTC1665 does not support negative reference voltages. It is equipped with one single reference voltage input named REF for positive reference voltages within range from 0V .. VCC (pls. LTC1665 datasheet for details).

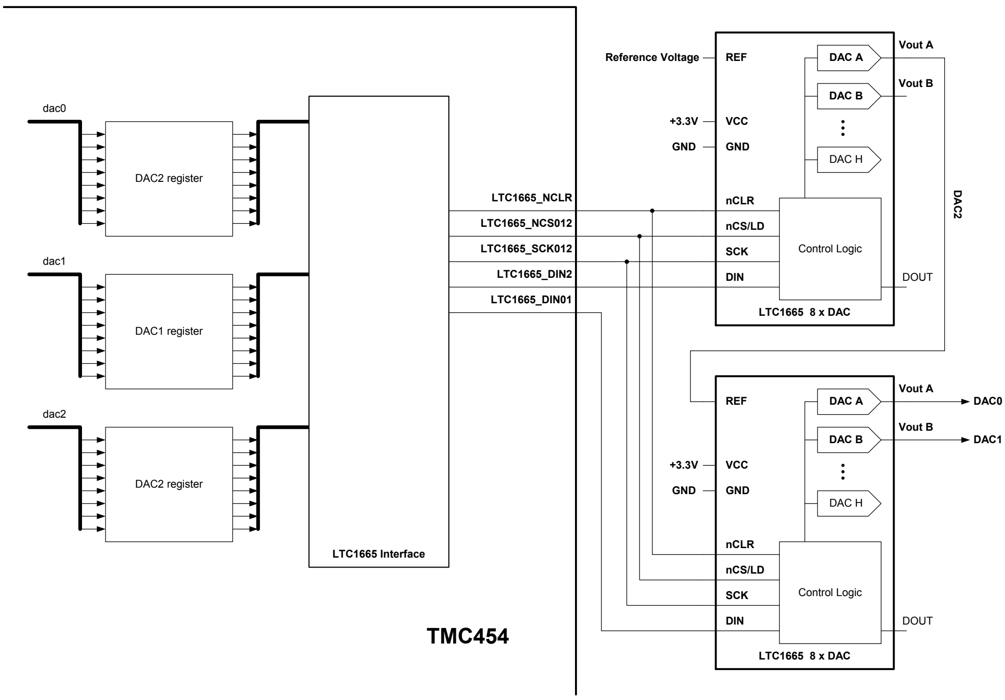


Figure 4.3: Connection of external DACs LTC1665 with the TMC454

4.7 Digital part

	Min	Typ	Max	Units
Input Low Level	0		0.8	V
Input High Level	2.0		3.6	V
Output Low Level @ I= 12mA			0.4	V
Output High Level @ I=-12mA	2.4			V
Input Current			± 10	µA
Output Short Circuit Current to High (V33 =3.3V)			109	mA
Output Short Circuit Current to Low (V33 =3.3V)			103	mA
Input Capacitance			8	pF

Table 4.6 : CMOS DC Characteristics (V33 = 3.3V ± 10 %; 0°C < $T_{ambient}$ < 70°C)

5 The Bus interface

The TMC454 has got both a parallel interface with multiplexed address/data bus and a serial 2-wire serial interface to allow communication with a host processor. The parallel interface can either be directly connected to a host processor or can be driven via port lines. The serial interface also allows operation of other circuits on the same bus. Which interface is enabled is decided by the polarity of the SERIAL_EN pin during a chip reset. The polarity of SERIAL_EN is latched with the rising edge at the NRES pin.

5.1 Parallel Interface

The parallel interface allows direct connection to processor systems with multiplexed address/data bus like the 8051 series. It is important to make sure that the TMC454 is clocked fast enough to satisfy the access time requirements of the processor. For non-multiplexed processor systems it is possible to decode the TMC454s ALE signal on one address of the CPUs address space and the TMC454 NCS on the following address. This allows write accesses to the TMC454 using two subsequent writes and read accesses using a write access to the address register and a subsequent read access to the data register. Another possibility is control of the TMC454 using 11 I/O ports. This is especially recommended when many TMC454s are controlled by one processor or the distance between host CPU and TMC454 is large.

The TMC454 is equipped with an additional nCS_OUT output signal for microcontrollers with separate nRD / nWR signals. This saves an additional and gate. For micro controllers with nCE and nWR/RD the nCS_OUT of the TMC454 can be ignored.

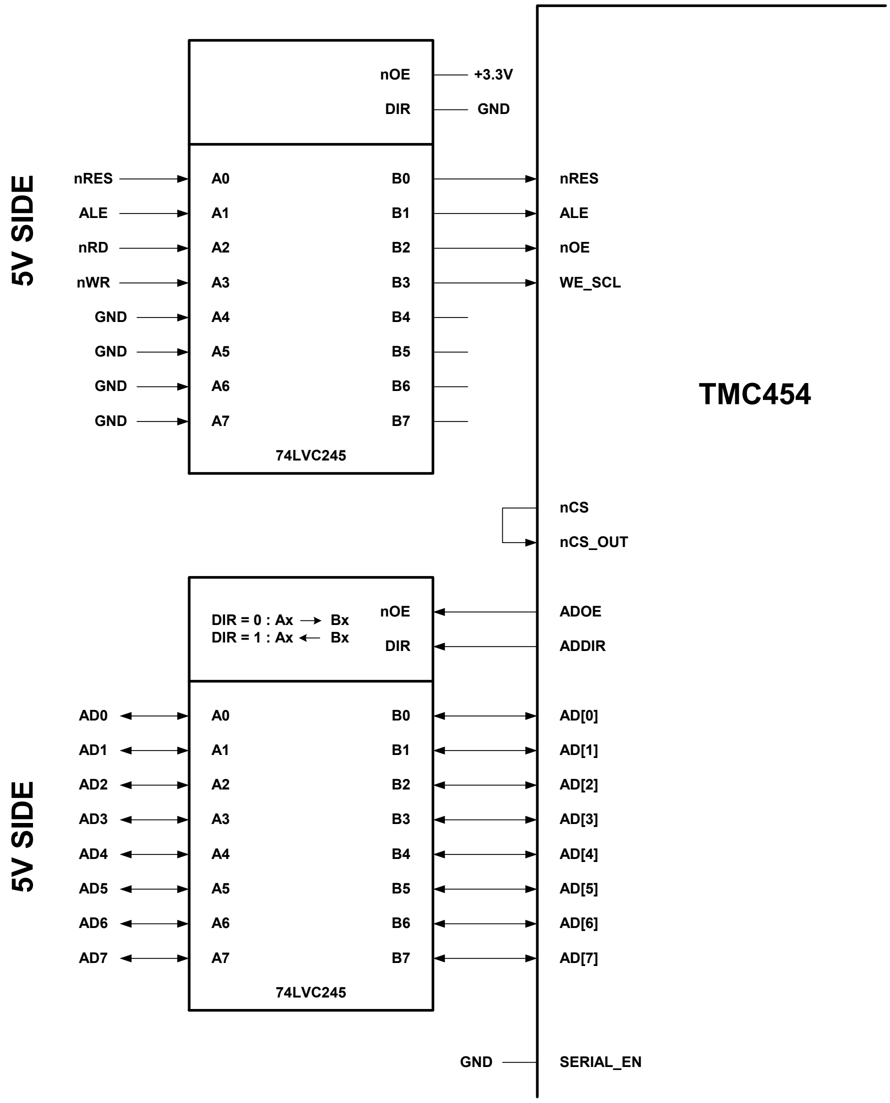
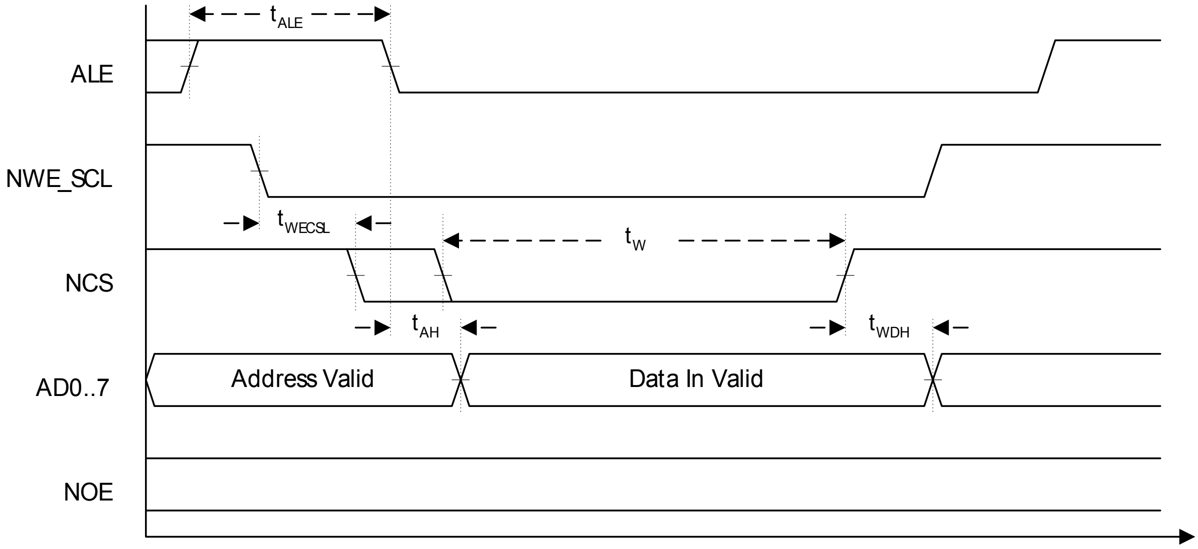


Figure 5.1 : Parallel Interface with 5V to 3V3 Level Shifters

Write Access Timing



Read Access Timing

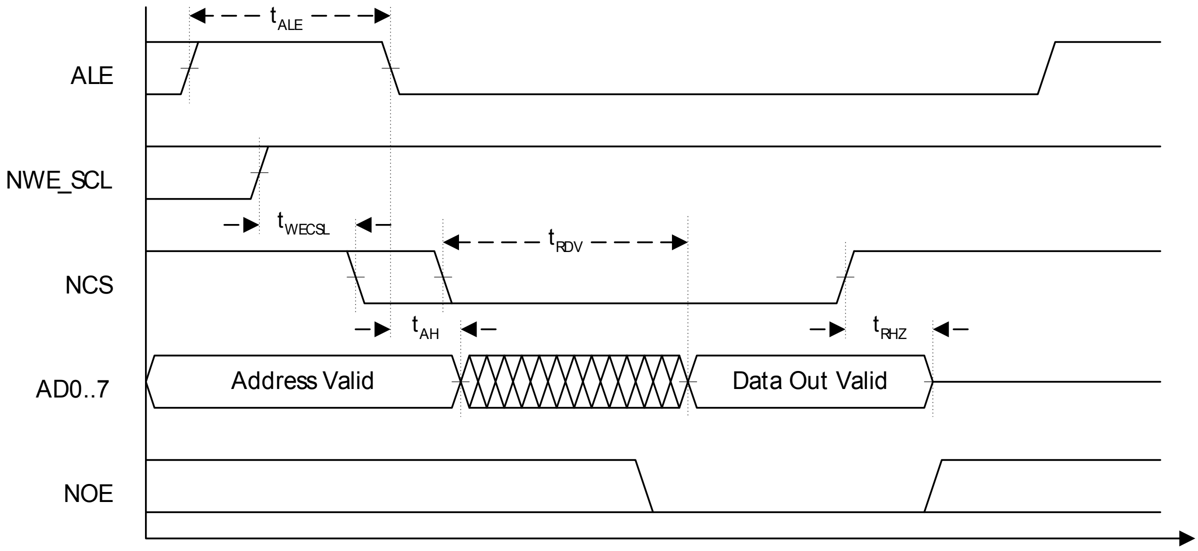


Figure 5.2: Timing of the parallel interface

Symbol	Meaning	Min	Max
t_{ALE}	ALE active time	30 ns	-
t_{WECSL}	Write enable valid to CS active or ALE inactive (whichever comes last)	10 ns	-
t_{AH}	Address hold time after ALE inactive	10 ns	-
t_{W}	Write time. A write occurs during the overlap of an active CS, active WE and inactive ALE.	1 t_{clk} + 10 ns	-
t_{WDH}	Write data hold time after Write end	10 ns	-
t_{WREC}	Write recovery time. Time from write access end to next Write / Read access	2 t_{clk}	-
t_{RDV}	Read access time. Time from read start to valid data out. A read occurs during the overlap of an active CS, inactive WE and inactive ALE.	-	3 t_{clk} +20 ns
t_{RREC}	Read recovery time. Time from read access end to next write / read access	1 t_{clk}	-
t_{RHZ}	Output tristate time (from CS, WE, OE or ALE)	0 ns	10 ns

1) t_{clk} is the length of one clock period at the clock input of the TMC454, t_{clk}>= 80 ns (t.b.d.)

5.2 Serial Interface

The serial 2-wire bus allows data transfer rates of several 100 kbit/s and addressing of up to 128 TMC454s using a simple protocol. It uses the lines SDA and SCL (pin NWE_SCL). The protocol can be simply implemented in software while hardware interfaces are available also. In this mode the address/data pins AD1 to AD7 select the serial address of the TMC454. The remaining pins of the parallel interface are not used and should be connected to a defined voltage (/CS inactive). The serial lines SCL and SDA are internally filtered over three system clocks.

The internal address counter is incremented after every access.

Symbol	Meaning	Min	Max
t_{SCL}	SCL high / low time	5 t_{clk}	-
t_{SUSS}	Start / Stop condition data setup time	5 t_{clk}	-
t_{HSS}	Start / Stop condition data hold time	5 t_{clk}	-

1) t_{clk} is the length of one clock period at the clock input of the TMC454

Serial Bus Data Transfer Sequence

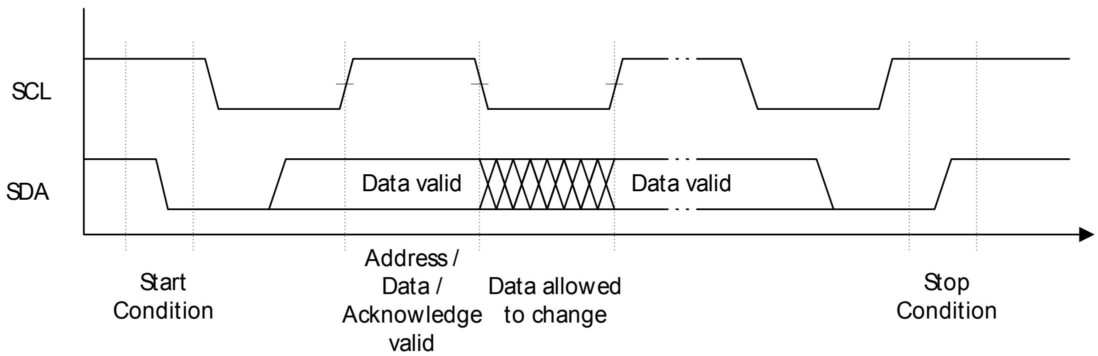


Figure 5.3: Data transfer on the serial bus

The data line SDA is a bi-directional open-drain port. An external pull-up resistor with a value of some kilo ohms is required for operation. During data transmission the logic level on SDA is only allowed to changed, when the level on SCL is low. Changes during the high-phase of SCL are reserved for start and stop conditions.

The clock line SCL is only used as input for the synchronization of the data bits.

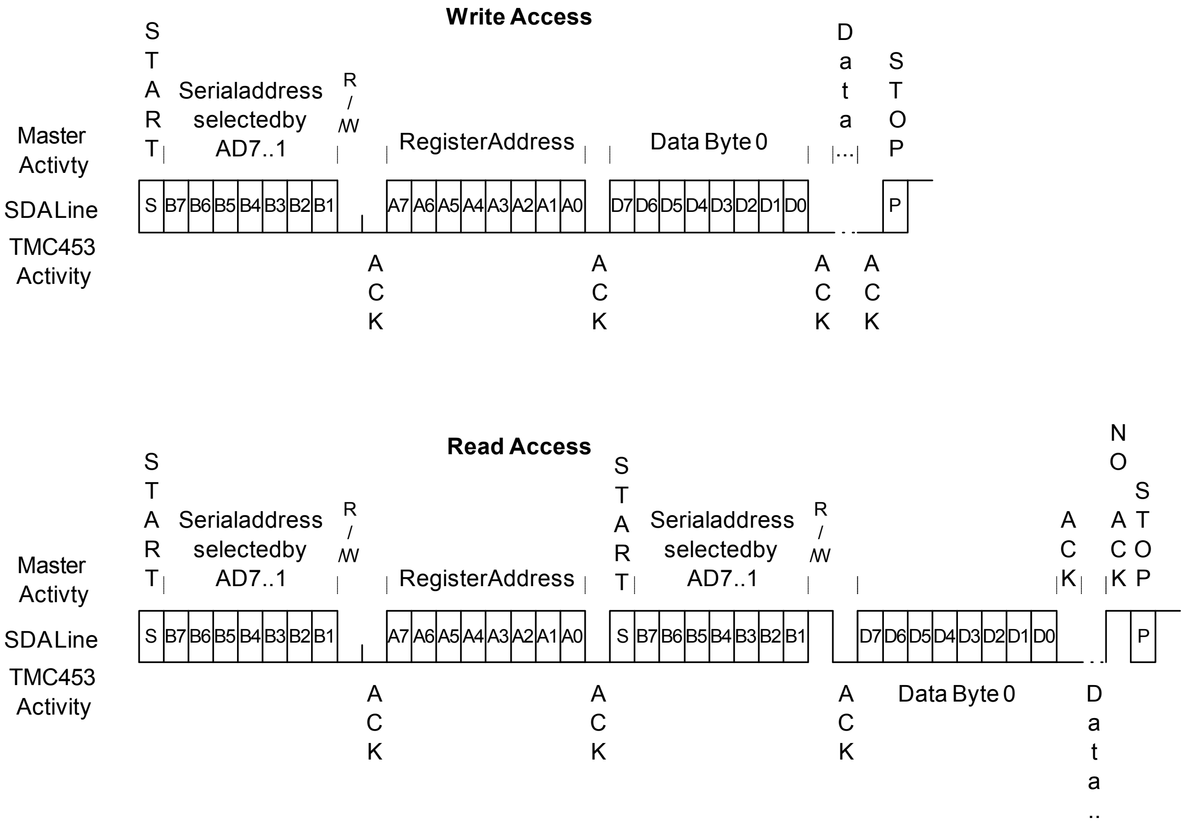


Figure 5.4: Telegram on the serial bus

The I/O signals of the TMC454 have 3.3V CMOS level. So, the serial interface of the TMC454 has 3.3V CMOS level. The SDA is realized as an open drain output. To use the TMC454 within a 5V environment, the TMC454 provides a open drain control signal (SDA_nOE). The IIC clock SCL and the SDA for the TMC454 have to be converted with by level shifters.

6 Description of the COMMAND FIFO

6.1 Accessing the TMC454

The TMC454 contains a set of registers with a width of up to 32 bits. Its bus interfaces transfer data in portions of 8 bits. To allow the access to 32 bit values, the TMC454 internally uses a 32 bit bus architecture. When accessing registers, which are wider than one byte, the new value is transferred into the internal register whenever the most significant byte is written. Table 4.1 : TMC454-BC pinoutTherefore, the host processor has to write the bytes in ascending order, e.g. the least significant byte (lowest address) first. When using the serial bus interface this is already implied by the automatic address increment. Accordingly, when reading registers with a width of more than one byte, the least significant byte has to be read first. Whenever the first byte of a register is read, the complete register is copied into an internal latch, which is read on the subsequent accesses to the higher bits of the register. Thus it is guaranteed that the values are consistent.

Reset Values

All registers contain “all bits cleared” after reset, unless stated differently. All functions which can be enabled by register bits are enabled by a “one-“ bit.

6.2 General functionality

This module controls all real time-critical functions in the TMC454. These are especially the functions for the generation of velocity ramps, as well as the control of motor parameters, which could have to be altered in different segments of a ramp. The system's microprocessor writes commands into the command FIFO which are executed sequentially whenever the preceding command is finished. The commands consist each of a number of actions and a condition for the termination. When the condition is met, the TMC454 continues with the execution of the next command at once. In fact the execution time is only three clock cycles, so that typically a number of values can be changed between each two motor steps.

6.3 Description of the registers of the COMMAND FIFO

fifo_command: 32 Bit (address: 0x00h)		
Bit	**Term**	**description**
0..31 W	FIFO_COMMAND	Entry of new commands into the FIFO
0..31 R	ACTIVE_COMMAND	Readout of the command currently in execution

6.3.1 FIFO Commands

The Command FIFO interprets a 32 bit wide command word in each execution step. The command can either consist of a 16 bit wide operation code and up to 16 data bits, or of an 8 bit wide operation code and up to 24 data bits. 24 data bits are used for position registers and time limit. The code in byte 0 of the command determines the type (width of the argument and op-code). Commands can only write to registers. Reading registers is possible via separate read-registers.

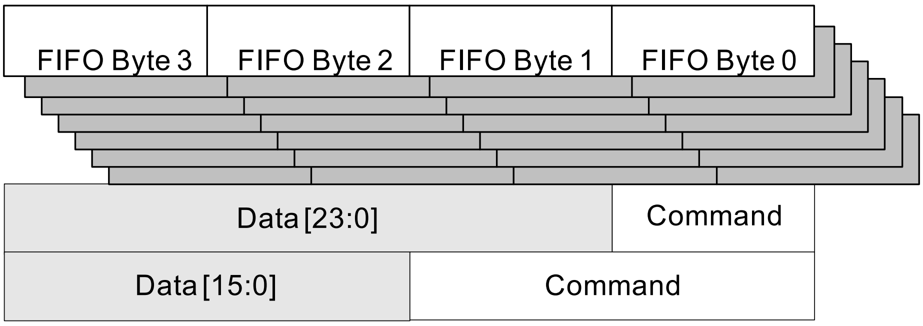


Figure 6.1: FIFO-Command types

Structure of the two command types

Command Type	FIFO Byte 3 (Bit 31..24)	FIFO Byte 2 (Bit 23..16)	FIFO Byte 1 (Bit 15..8)	FIFO Byte 0 (Bit 7..0)
24 Bit value	Value3 [23..16]	Value3 [15..8]	Value3 [7..0]	Control [7..0]
16 Bit value	Value2 [15..8]	Value2 [7..0]	Extended Control [15..8]	Control [7..0]

The parameters associated with Value2 / Value3 are adjusted with the LSB at the least significant bit position. It is always necessary to write the complete 32 bit entry, the most significant byte as last value.

Components of the Command code, lower byte

Control [0..7]		
Bit	**Term**	**Description**
0..3	Command Opcode	Operation code (s. table)
4	WAIT_ON_SYNC	When set: This command is not executed before SYNC_IN has got the polarity which is defined by Bit 5 (SYNC_POLARITY)
5	SYNC_POLARITY	Polarity of SYNC_IN for the WAIT_ON_SYNC function
6	COPY_REG_INT	When this flag is set, the following actions are done, when execution of the command is started: 1. The actual values of the ramp registers are copied to a set of holding registers 2. A FIFO-Interrupt is generated
7	FIFO_OVERRIDE	When this bit is set in a command, all previous FIFO entries which write values are executed at once, while all ramp commands are skipped and the waiting conditions are ignored. Then the new command is executed. This bit is interpreted at once when writing to the FIFO

Command Opcodes for bits 0..3

Comman d	Opcod e	Description
PRP	0000	**Start parabolic ramp**: In this type of ramp the acceleration RAMP_ACT_ACCEL is increased continuously by the value FIFO_BOW, while the velocity RAMP_ACTVEL is increased continuously by the resulting acceleration value. *Necessary precondition:* FIFO_BOW, FIFO_A_NOM and FIFO_V_NOM are set *Termination conditions:* FIFO_A_NOM reached, FIFO_V_NOM reached, FIFO_POS_END reached or FIFO_T_LIM reached
LRP	0001	**Start linear ramp**: The acceleration ACT_ACCEL is constant and is added continuously to the velocity ACTVEL. *Necessary precondition:* ACT_ACCEL and FIFO_V_NOM are set *Termination conditions:* FIFO_V_NOM reached, FIFO_POS_END reached or FIFO_T_LIM reached
CRP	0010	**Constant ramp** (constant velocity): The velocity is not changed. *Necessary preconditions:* The desired velocity value is set *Termination conditions:* FIFO_POS_END reached or FIFO_T_LIM reached
ARP	0011	**Start automatic ramp**: The motor is driven to its target position using and S-shaped ramp. *Necessary preconditions:* ACTVEL=0; FIFO_BOW, FIFO_A_NOM and FIFO_V_NOM are set *Termination conditions:* FIFO_POS_END reached (always enabled) or FIFO_T_LIM reached *Note:* When a different ramp function was active before, you should set the mode CRP and set ACTVEL=0 before starting an automatic ramp.
SET_ANA	1000	**Setting analog registers:** The register is specified by the Extended Control-ANA command code.
SET_RMP	1010	**Setting parameters of the ramp generator:** The register is specified by the Extended Control command code.
SET_TLIM	1100	**Setting a time limit:** (8 bit command with 24 bit parameters) FIFO_T_LIM = Value3 The time limit register is loaded with the specified value.
SET_POS_ ACT	1101	**Setting the position counter:** (8 bit command with 24 bit parameters) POS_ACT = Value3 POS_ACT is loaded with the new value
SET_POS_ END	1110	**Setting the target position:** (8 bit command with 24 bit parameters) POS_END = Value3 POS_END is loaded with the new value.
NOP	1111	**No operation resp. no change of the ramp mode:** This opcode does not change the operation mode. The other command bits are

		evaluated as in a 16 bit wide opcode.

Structure of the Command code, upper byte, only with 16 bit command code, except with SET_ANA command

With 8 bit command code the previous settings of the bits 12..15 are maintained.

Extended Control [8..15]		
Bit	**Term**	**Description**
8..11	Register Write Command Code	The parameters selected with these bits are loaded with the value which is given as Value2: 0000: FIFO_A_NOM = Value2 (14 Bit signed) 0001: FIFO_V_NOM = Value2 (14 Bit signed) 0010: ACTACCEL = Value2 (14 Bit signed) 0011: ACTVEL = Value2 (14 Bit signed) 0100: FIFO_A_SLD = Value2 (13 Bit unsigned) 0101: FIFO_V_SLD = Value2 (13 Bit unsigned) 0110: FIFO_BOW = Value2 (14 Bit signed) 0111: FIFO_PRE_DIV4 = Value2 (4 Bit) 1110: FIFO_MISC_CTRL = Value2 (2 Bit) (see Ramp Generator description) 1111: Do not change any parameters The values have to be right adjusted with the width given in parentheses.
12	END_BY_POS	Only when this flag is set, the current command can be terminated by reaching the target position. When cleared, this termination condition is switched off. This flag has no influence on the automatic ramp.
13	END_BY_TIME	Only when this flag is set, the current command can be terminated when the time limit is reached.
14	SYNC_OUT_VAL	This bit controls the polarity of the synchronization output.
15	CLEAR_TIMER	When this bit is set, the time counter is set to zero upon command begin.

Structure of the command code, upper byte, only with SET_ANA command

Extended Control [8..15]		
Bit	**Term**	**Description**
8..11	SET_ANA Register Write Command Code	The parameters of the analog controller selected by these bits are loaded with the value which is given as Value2: 0000: FIFO_A_COMP1 = Value2 (13 Bit unsigned) 0001: FIFO_V_COMP1 = Value2 (13 Bit unsigned) 0010: FIFO_A_COMP2 = Value2 (13 Bit unsigned) 0011: FIFO_V_COMP2 = Value2 (13 Bit unsigned) 0100: FIFO_IMOT0 = Value2 (8 Bit unsigned) 0101: FIFO_IMOT1 = Value2 (8 Bit unsigned) 0110: FIFO_IMOT2 = Value2 (8 Bit unsigned) 0111: FIFO_IMOT3 = Value2 (8 Bit unsigned)

6.3.2 Description of the status bits

fifo_status: 14 bit r (address: 0x04h)		
Bit	**Term**	**Description**
0..2	FIFO_ENTRIES	actual number of FIFO entries (0..7), 0=FIFO empty
8	A_NOM_REACHED	nominal acceleration reached
9	V_NOM_REACHED	nominal velocity reached
10	POS_END_REACHED	target position reached
11	AUTO_ACTIVE	automatic ramp active
12	STOP_CONDITION	Stop condition has occurred (Flag is cleared upon read of this register)
13	SLD_CONDITION	Slowdown function is active (caused by a SLD pin) (Flag is cleared upon read of this register)

fifo_input_status: 5 Bit r (address: 0x06h)		
Bit	**Term**	**Description**
0	SLD_RIGHT	right slowdown switch is active
1	SLD_LEFT	left slowdown switch is active
2	STOP_RIGHT	right brake switch is active
3	STOP_LEFT	left brake switch is active
4	SYNC_IN	polarity of external synchronization pin

fifo_port_func: 2 Bit r/w (address: 0x07h)		
Bit	**Term**	**Description**
0	ENABLE_STOP	Enable stop function via stop switches (Default: 1)
1	ENABLE_SLD	Enable function of the slowdown switches

6.3.3 STOP and SLOWDOWN-functions

The TMC454 supports stop switches (pins NSTOPL, NSTOPR) and a slowdown function (pins NSLDL, NSLDR). When an impulse occurs on the stop input which corresponds to the motor direction (increasing position value corresponds to right switch), the motor is stopped at once. This is done by setting the velocity value to zero and stopping the ramp function. The stop flag in the FIFO-Status register is set.

While an SLD input is active, the ramp generator is switched to linear ramp and the velocity is continuously decreased by the pre-programmed slowdown acceleration *fifo_a_sld* until the velocity *fifo_v_sld* is reached. The SLD function can not be used with automatic ramps. When an automatic ramp function was active, it is terminated and the motor stops at once. The application environment of the TMC454 should be designed in a way, that the SLD input stays active for at least the time that the external host processor needs to switch off the previously active ramp function, e.g. by switching to a constant ramp. Short impulses on the SLD functions could lead to uncontrolled reactions - they have to be filtered, e.g. using an RC-network. It is advised to completely control the slowdown function via software, because interaction of the processor is required in most cases. This can be efficiently realized using the interrupt ability of the SLD pins.

Stop and SLD flag are automatically reset upon read of the FIFO status register.

6.3.4 Finding the Reference Position

The FIFO synchronization input SYNC_IN can be efficiently used to perform as reference input for null position finding. This is possible using the synchronization condition: The motor drives a linear or constant ramp segment in the direction of the reference switch. The next command has got the WAIT_ON_SYNC and the COPY_REG_INT bits set together with FIFO_OVERRIDE. The reference switch then triggers this command via SYNC_IN. The latched ramp position now describes the exact position of the reference switch.

The reference position can also be found using the stop switches and the automatic stop function: First do a rough search for the stop switch using a high velocity. Then, drive the motor out of the switch again and go back to the switch using a constant ramp with the motors start-/ stop velocity and with stop function enabled. As soon as the motor is stopped, you can set the actual position to zero.

6.3.5 Programming example for the FIFO

The following example shows how to program the TMC454 to start an automatic ramp in positive direction driving a 2 phase stepper motor in sine step mode. The codes for the control registers are shown in binary form (%).

1. Switch off the sequencer:
   `seq_config Byte0 = %00000000, Byte1 = %00000000`
2. Program sinestep and switch on the sequencer:
   `seq_config Byte0 = %10001100, Byte1 = %00000101`
3. Program the pre-divider for the desired step frequency range, e.g. FIFO_PRE_DIV4 = 8
   using the FIFO command SET_RMP:
   `FIFO-Command Byte0 = %00001010, Byte1 = %00000111, Byte2 = 8, Byte3 = 0`
4. Program the bow parameter, e.g. FIFO_BOW = 5:
   `FIFO-Command Byte0 = %00001010, Byte1 = %00000110, Byte2 = 5, Byte3 = 0`
5. Program the acceleration parameter, e.g. FIFO_A_NOM = 100:
   `FIFO-Command Byte0 = %00001010, Byte1 = %00000000, Byte2 = 100, Byte3 = 0`
6. Program the velocity parameter, e.g. FIFO_V_NOM = 6000 = 0x1770:
   `FIFO-Command Byte0 = %00001010, Byte1 = %00000001, Byte2 = 0x70, Byte3 = 0x17`
7. Program the target position, e.g. POS_END = 70000 = 0x011170
   using the FIFO command SET_POS_END:
   `FIFO-Command Byte0 = %00001110, Byte1 = 0x70, Byte2 = 0x11, Byte3 = 0x01`
8. Start the automatic ramp via FIFO command ARP without modification of further parameters:
   `FIFO-Command Byte0 = %00000011, Byte1 = %00001111, Byte2 = 0, Byte3 = 0`

7 The Ramp Generator

7.1 General Description

The ramp generator can generate velocity ramps (velocity over time) with constant velocity, constant acceleration and linear rising acceleration. This allows the generation of step-less and smooth ramps to avoid the motor loosing steps at the joints of each two curve segments. The so called S-curve can be generated automatically. It has got several advantages compared to the trapezoidal ramps which are commonly used for positioning tasks.

7.2 Principle of Operation

In principle the ramp generator is a cascade of three integrators (Figure 7.1). All integrators work with the frequency f_{ramp}, which is divided from the clock frequency via a divider (*fifo_pre_div*) which can be programmed in powers of two. The same frequency is used to derive the step frequency of the motor. The first integrator generates a linear change of the acceleration for parabolic curves. It adds up the 14 bit wide parameter *fifo_bow* in every time step to the 22 bit wide acceleration register (*ramp_actaccel*). The integrator is stopped as soon as the pre-programmed value for the nominal acceleration (*fifo_a_nom*) is reached. The velocity value is generated using the same mechanism: The 22 bit wide velocity integrator adds up the upper 14 bits of the acceleration value in every time step until the nominal velocity (*fifo_v_nom*) is reached.
The TMC454 avoids overflows of the integrators by stopping the integration if the next addition step would exceed the preprogrammed nominal values. Thus it is important to always choose correct nominal values to stop integration.
To yield a finer step resolution, the input values of the integrators are shifted to the right by 8 bits. This corresponds to a division by 256.

The velocity value (*ramp_actvel*) calculated by the integrator chain feeds a programmable pulse generator, which generates one impulse for every motor step. The frequency of this pulse generator can be calculated as follows:

Step frequency of the ramp generator (Microsteps or fullsteps depending on the settings of the sequencer)

$$f_{step} = f_{clk} \cdot v / 2^{14+fifo\_pre\_div+1}$$

full step frequency = micro step frequency (f_{step}) / microstep count
f_{clk}: external clock frequency of the TMC454
v: actual 14 bit velocity value, signed (-8191..8191)

Operation frequency of the ramp generator
The calculation of the actual velocity and acceleration values as well as the time counter work with a frequency which is controlled by the pre-divider (f_{ramp}):

$$f_{ramp} = f_{clk} / 2^{fifo\_pre\_div+1}$$

All settings of the ramp generator are controlled via the command FIFO. The corresponding registers and commands are listed in the chapter on the command FIFO.

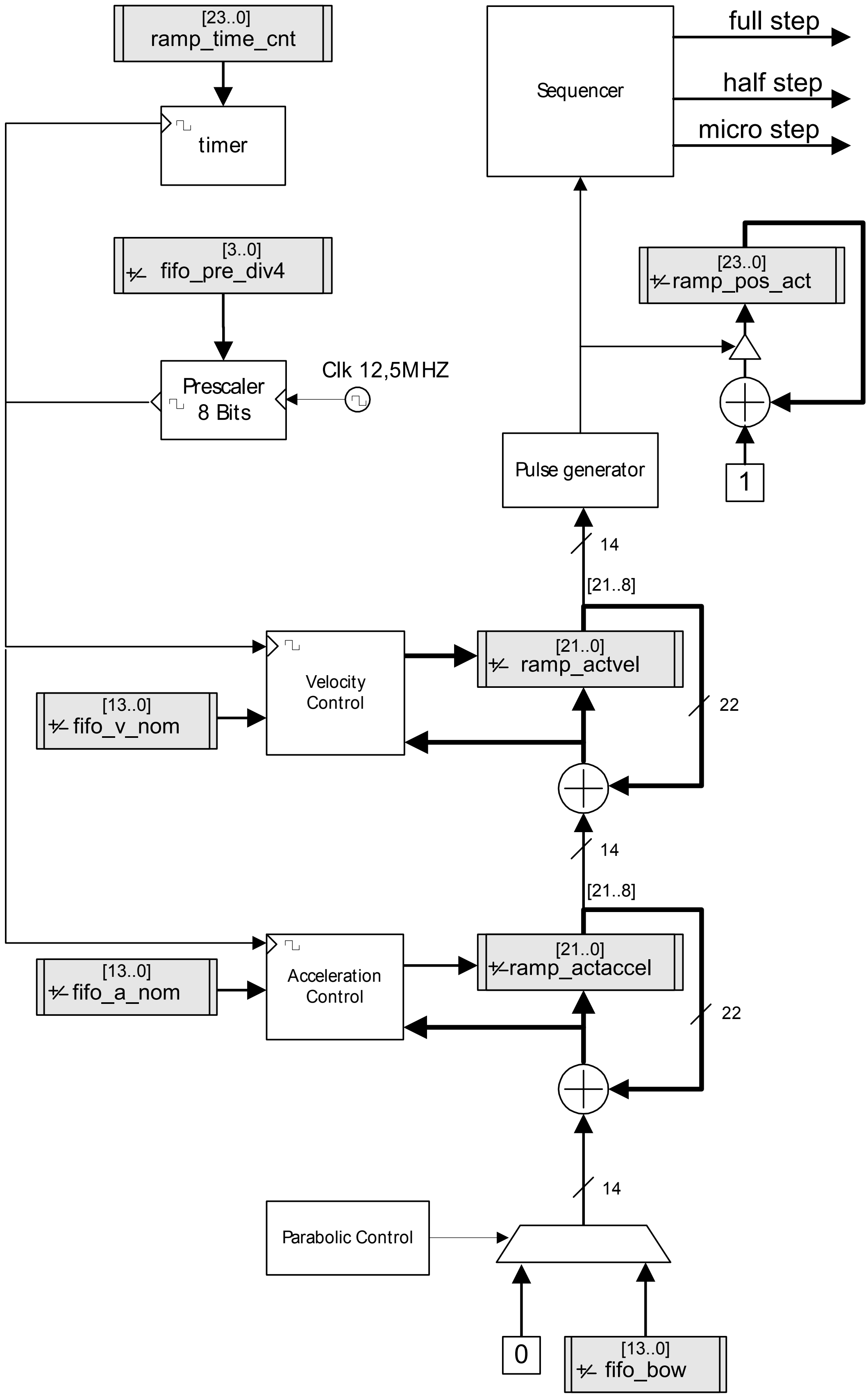


Figure 7.1: Schematic of the ramp generator

Registers of the ramp generator

Address	Term	Width	Description
0x40	latch_ramp_-params	0 (w)	A write access to this register causes all relevant data of the ramp generator to be copied to the ramp-holding registers. (This is the same as the execution of a command with the bit COPY_REG_INT set)
0x41	ramp_status	8 (r)	Holding register for the state of the ramp generator
0x44	ramp_time_cnt	24 (rw)	Internal ramp time reference (counts ramp generator clocks) (On read access: value of the holding register)
0x48	ramp_actvel	22 (r)	Holding register for velocity value (s. rounding *)
0x4C	ramp_actaccel	22 (r)	Holding register for acceleration value (s. rounding *)
0x50	ramp_pos_act	24 (r)	Holding register for position counter (actual value)
0x60	fifo_a_nom	14 (r)	Nominal acceleration for ramp generation
0x62	fifo_v_nom	14 (r)	Nominal velocity for ramp generation
0x64	fifo_a_sld	13 (r)	Acceleration for slowdown operation
0x66	fifo_v_sld	13 (r)	Final velocity for slowdown operation
0x68	fifo_bow14	14 (r)	Ascent of acceleration (bow parameter) for ramp generation
0x6A	fifo_a_comp1	13 (r)	Acceleration compare value for automatic motor current control (MC0, MC1,AOUT2)
0x6C	fifo_v_comp1	13 (r)	Velocity compare value for automatic motor current control (MC0, MC1, AOUT2)
0x6E	fifo_a_comp2	13 (r)	Acceleration compare value for automatic motor current control (MC0, MC1, AOUT2)
0x70	fifo_v_comp2	13 (r)	Velocity compare value for automatic motor current control (MC0, MC1, AOUT2)
0x74	fifo_pre_div4	4 (r)	Pre-divider for ramp generator (division in powers of two)
0x75	fifo_imot0	8 (r)	Output value for analog output AOUT2 for automatic ramp-dependant current control with 0 exceeded compare values
0x76	fifo_imot1	8 (r)	Output value for analog output AOUT2 for automatic ramp-dependant current control with 1 exceeded compare value
0x77	fifo_imot2	8 (r)	Output value for analog output AOUT2 for automatic ramp-dependant current control with 2 exceeded compare values
0x78	fifo_imot3	8 (r)	Output value for analog output AOUT2 for automatic ramp-dependant current control with 3 or 4 exceeded compare values
0x79	fifo_misc_ctrl	2 (r)	Flag for PID controller (bit 1) and flag for control based on the measured velocity (bit 0)
0x7C	fifo_t_lim	24 (r)	Time limit for execution of the actual FIFO-command
0x80	fifo_pos_end	24 (r)	End position for ramp segment

(*) Rounding of the 22 bit wide values in *ramp_actvel* and *ramp_actaccel*:
For all internal calculations the values in the 22 bit wide velocity and acceleration registers are rounded to 14 bit signed values. The internal rounding algorithm rounds into the direction of the next number with a higher absolute value when the least significant 8 bits have got a value between 0x80 and 0xFF for positive numbers, respectively between 0x7F and 0x00 for negative numbers.

Hint:
When reading the two's-complement signed numbers from the TMC454 into the host processor, please note, that they have to be sign-extended before using in 16 or 32 bit arithmetic.

Ramp_status: 7 Bit r (address: 0x41h)		
Bit	**Term**	**Description**
0	SLD_RIGHT	Right slowdown switch active
1	SLD_LEFT	Left slowdown switch active
2	STOP_RIGHT	Right stop switch active
3	STOP_LEFT	Left stop switch active
4	A_NOM_REACHED	Nominal acceleration reached
5	V_NOM_REACHED	Nominal velocity reached
6	POS_NOM_REACHED	Target position reached
7	AUTO_ACTIVE	Automatic ramp generation active

7.3 Programming the Ramp generator

7.3.1 Automatic Ramp generation

Before starting a ramp, the necessary parameters of the ramp generator have to be programmed. For an automatic ramp (Figure 7.2) for example, the parameters *fifo_bow*, *fifo_a_nom*, *fifo_v_nom*, and the target position *fifo_pos_end* have to be programmed. It is very important to select the signs of all parameters correctly: For example consider the motor driving in negative direction (decreasing position) – then all ramp parameters have to be set to negative values. Further the automatic ramp generator does not start if the actual velocity is different from zero.

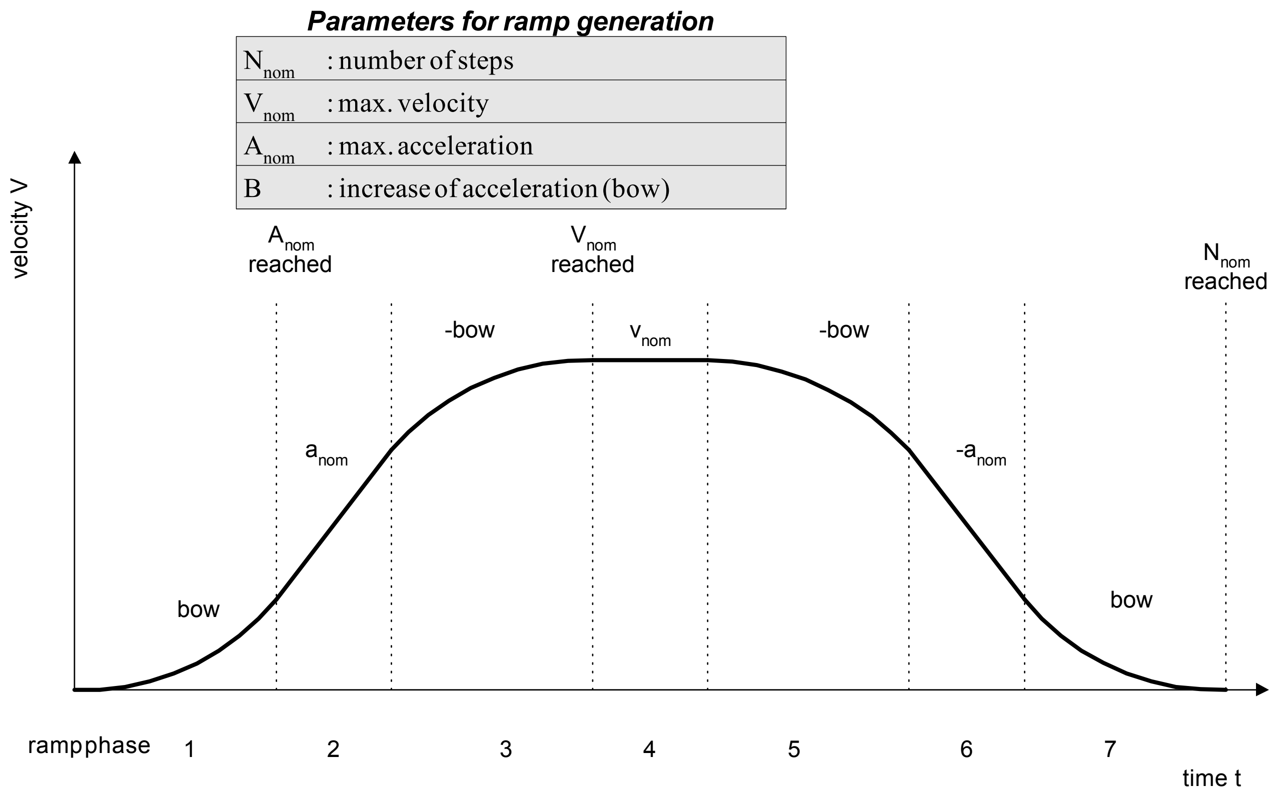


Figure 7.2: Automatically generated S-ramp

Note: Terminating an automatic ramp

Should a mistake occur when programming the automatic ramp, e.g. the programmed velocity value is zero, it can only be terminated by writing a FIFO-command with the override-bit set (FIFO_OVERRIDE) which sets the actual position to the preprogrammed target position, or by activating one of the other ramp types. In every case termination of the automatic ramp clears the values for actual velocity and actual acceleration to zero leading to an abrupt stop of the motor.

To terminate an automatic ramp and slow down to zero in a controlled way, use the following programming sequence:

1. Write a command for a linear ramp (LRP) into the FIFO. (This command is not yet executed!)

2. Program an acceleration for slowing down to zero using a write to ACTACCEL command (mind the sign!).
3. Set the new nominal velocity (that is 0) using a write to FIFO_V_NOM command.
4. Now read the actual velocity and, (if it is not 0 anyway,) at once write it back to the actual velocity with the override bit (FIFO_OVERRIDE) set using a linear ramp command (LRP) combined with write to ACTVEL.
5. Program an acceleration of zero using a write to ACTACCEL command, to signal the automatic current control, that current can be reduced to standstill value after motor stop.

Now the TMC454 slows down to zero with the programmed acceleration. The command FIFO is empty as soon as the motor has stopped.

7.3.2 Programmed / Interactive Ramp generation

A number of applications requires the precise calculation of ramps, to reach certain coordinates on given points of time. An example is a plotter, where two axes have to be synchronized. In these cases the host CPU can program user defined ramps by constructing the desired curve shapes from bits of parabolic and linear ramps. Then it will be necessary to exactly control the number of steps driven in every ramp segment and to reprogram the ramp generator at the calculated positions before the next motor step is done. Because of the limited time between each two steps the TMC454 supports a programming method capable of real time reprogramming using its command FIFO: As soon as a preprogrammed condition is satisfied, the next command is fetched from the FIFO and executed. Some of the possible conditions are: Nominal velocity or nominal acceleration reached, position reached or time limit reached. To inform the host CPU, when the next command starts, it can issue an interrupt after each command. Further on a snapshot of all ramp registers can be triggered at the same moment.

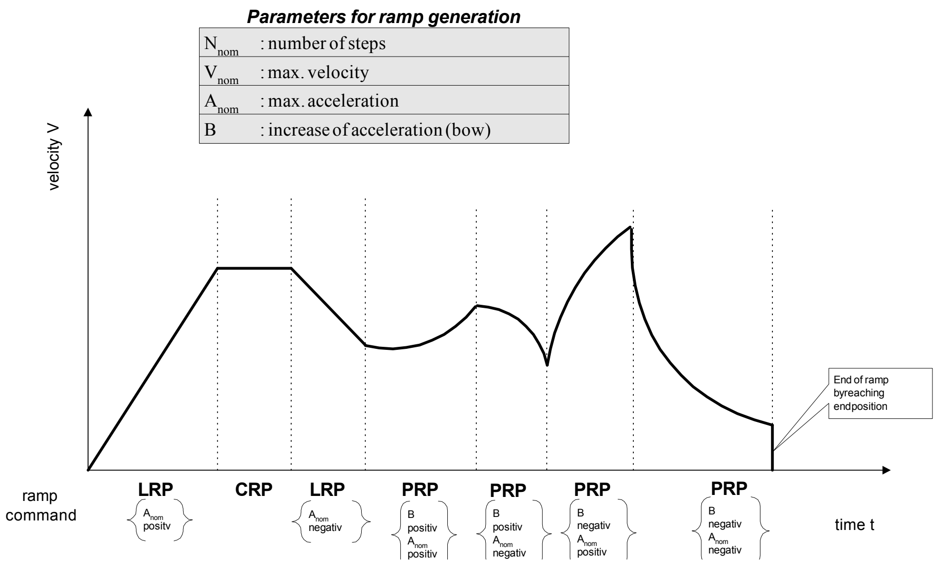


Figure 7.3: User defined ramp

7.3.3 Synchronization of multiple TMC454s

The command FIFO allows the synchronization of multiple TMC454s on a ramp segment basis, e.g. to start all motors in the same moment. In a typical application the host processor would program all TMC454s with the necessary commands and then start execution of the commands via the synchronization inputs (SYNCIN). Each FIFO command can specify the polarity of the SYNCIN pin as pre-condition for its execution. To start the TMC454s without further interaction of the host processor, each command can also control the polarity of the SYNCOUT pin of the TMC454. Using an external AND-operation between all SYNCOUT pins to control all SYNCIN pins in a system, allows functions like all axes waiting for the slowest one, before the next ramp is started. A step wise synchronization can be achieved by clocking multiple TMC454s with the same clock or by interconnecting the STEP_IN and STEP_OUT pins in the desired way.

Example for the programming of a ramp and FIFO usage:

No.	No. of entries	Command
1.	0	set *fifo_a_nom* (positive)
2.	0	set *fifo_bow* (positive)

3.	0	set *fifo_v_nom*
4.	0	set *fifo_pos_end*
5.	1	start parabolic curve with SYNCIN condition
6.	2	set *fifo_bow* again (negative), Start linear movement
7.	3	set *fifo_a_nom* to 0, start parabolic curve
8.	4	set *fifo_a_nom* (negative), start constant phase

Now set sync signal to start execution.

This example shows the programming of the first half of an S-curve.

7.4 Ramp adaptive motor current control

In many applications the stepper motor drives dynamic loads. Dynamic loads require a low static torque but a high torque when accelerating. To meet all requirements, the TMC454 is equipped with two compare registers for velocity control and two compare registers for acceleration control (fifo_a_comp1/2, fifo_v_comp1/2). The sum of the values exceeded in each moment is available as binary number at the outputs MC0/MC1 (limited to 0 to 3). At the same time this sum is used as a pointer to one of four IMOT registers. The value of the selected IMOT-Registers can be output to DAC2. This DAC can be used for control of the motor current by using the DAC2 output voltage as reference voltage for DAC0 and DAC1 in microstepping applications. A time dependant current control can be achieved using the timer commands of the command FIFO.

8 The Incremental Encoder Interface

1.1 General Description

This module decodes the signals of a digital incremental encoder (CHA,CHB,CHN) and provides a position register. Additional function registers allow position comparison of ramp position (ramp generator) and actual position (encoder) to monitor the stepper motor, or to enable regulation via the integrated PID controller. An interrupt can be issued when the difference exceeds a user programmable maximum value. Further the encoder signals can be converted to pulse and direction signals.

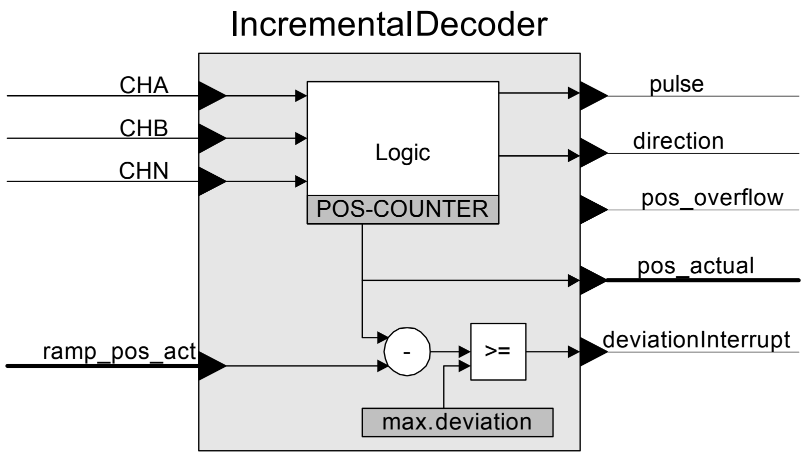


Figure 8.1: Schematic of the Incremental Encoder Interface

1.2 Registers of the Incremental Encoder Interface

enc_control: 7 bit rw (address: 0x10h)		
Bit	**Term**	**Description**
0	CHN_POL	Selects the polarity of the CHN input. [1] : positive, [0]: negative
1	CLR_POS_CNT_CHN	Set: The next CHN signal sets *ENC_COUNT* to zero
2	LTH_POS_CNT_CHN	Set: The next CHN signal copies *ENC_COUNT* to *LTH_POS*
3	LTH_POS_CNT_IMD	*ENC_COUNT* is copied to *LTH_POS* at once (bit resets automatically)
4	ACT_POS_IS_NOM_POS	*ENC_COUNT* is loaded with the ramp position *RAMP_POS_ACT* at once (bit resets automatically)
5	LTH_POS_CNT_RAMP_PAR	*ENC_COUNT* will be copied to *LTH_POS* together with the ramp parameters (when bit COPY_REG_INT is set in a FIFO command).
6	DIR_POL	Defines the direction of the encoder signals. [1]: CHA->CHB, [0]: CHB->CHA

enc_portstat: 3 bit r (address: 0x11h)		
Bit	**Term**	**Description**
0	CHN	State of the input port CHN
1	CHB	State of the input port CHB
2	CHA	State of the input port CHA

enc_count: 24 bit rw (address: 0x14h)		
Bit	**Term**	**Description**
0-23	ENC_COUNT	Actual value of the encoder counter

enc_holdreg: 24 bit r (address: 0x18h)		
Bit	**Term**	**Description**
0-23	LTH_POS	Holding register for encoder counter *ENC_COUNT*

enc_prediv_cnt: 8 bit rw (address: 0x1Ch)		
Bit	**Term**	**Description**
0-7	ENC_PRE_DIV_CNT	Pre-divider for incremental encoder (counts encoder signal changes)

enc_prediv_ratio: 8 bit rw (address: 0x1Dh)		
Bit	**Term**	**Description**
0-7	ENC_PRE_DIV_RATIO	Pre-divider ratio for incremental encoder (maximum value of counter) Ratio: 1/1..1/256

enc_deviation: 12 bit rw (address: 0x1Eh)		
Bit	**Term**	**Description**
0-11	ENC_DEVIATION	Maximum deviation between ramp position counter and encoder position counter: ABS(*ENC_COUNT* – *RAMP_POS_ACT*) ≤ *ENC_DEVIATION*. If the deviation is exceeded an interrupt is issued.

9 The Sequencer

The sequencer generates the control signals required by the different types of stepper motors. Motor type and stepping type can be programmed freely. For the ease of use the typical control patterns for 2-, 3-, and 5-phase motors are hardwired. Additionally the user can define any required step pattern with a length of up to 128 patterns. In microstep mode two different operation modes are distinguished. Microstep operation can either use the built in sine generator (sinestep) to control the coil currents or the integrated 128x8 RAM for user defined waves. The sine generator generates sine and cosine functions with programmable frequency and amplitude. The microstep RAM (MSR) can store either user defined waves or user defined control patterns of both.

The following chapters detail the registers required for the control of each stepping mode and their usage.

9.1 Registers for Sinestep operation

seq_sig_sin: 16 bit (address: 0x24)		
Bit	**Term**	**Description**
0-15	SINUS_ AMPLITUDE	Controls the amplitude of the sine wave. The signed value is given as a two's complement (7FFFh-8001h). The possible range is 3FFFh-C001h. Amplitudes larger than 3FFFh are clipped to 3FFFh. **Reset value: 1FFFh**

seq_sig_cos: 16 bit (address: 0x26)		
Bit	**Term**	**Description**
0-15	COSINUS_ AMPLITUDE	Controls the amplitude of the cosine wave. The signed value is given as a two's complement (7FFFh-8001h). The possible range is 3FFFh-C001h. Amplitudes larger than 3FFFh are clipped to 3FFFh. **Reset value: 2FFFh**

seq_reg_shift: 4 bit (address: 0x2E)		
Bit	**Term**	**Description**
0-3	SINE_RESOLUTION	Controls the resolution of the sine wave. **Reset value: 4h**

seq_sine_offset: 8 bit (address: 0x3A)		
Bit	**Term**	**Description**
0-7	SINE_OFFSET	Defines an optional DC-offset added to the sine and cosine wave.

Sine and cosine amplitude are signed values in two's complement notation. The valid range is 16384 to –16384. Internally the sine calculation uses a range of 32767 to –32768 to check for overflows. When the result is an overflow, the value is clipped to 16384 resp. -16384. If the result is under maximum value again, the sine wave is continued. To generate a full sine wave, the square-sum of sine register (*seq_sig_sin*) and cosine register (*seq_sig_cos*) has to be below 16384 squared (SINUS_ AMPLITUDE + COSINUS_ AMPLITUDE ≤ 16384). Figure 9.1 shows the valid parameter range of both registers.
The register *seq_sine_offset* allows the addition of a constant to both waves. When using the offset, the sum of the squares of the registers *seq_sig_sin, seq_sig_cos* and *seq_sine_offset* has to be less or equal to the squared maximum value (SINUS_ AMPLITUDE² + COSINUS_ AMPLITUDE² + SINE_OFFSET ≤ 16384²).
The register *seq_reg_shift* controls the resolution of the sine wave.

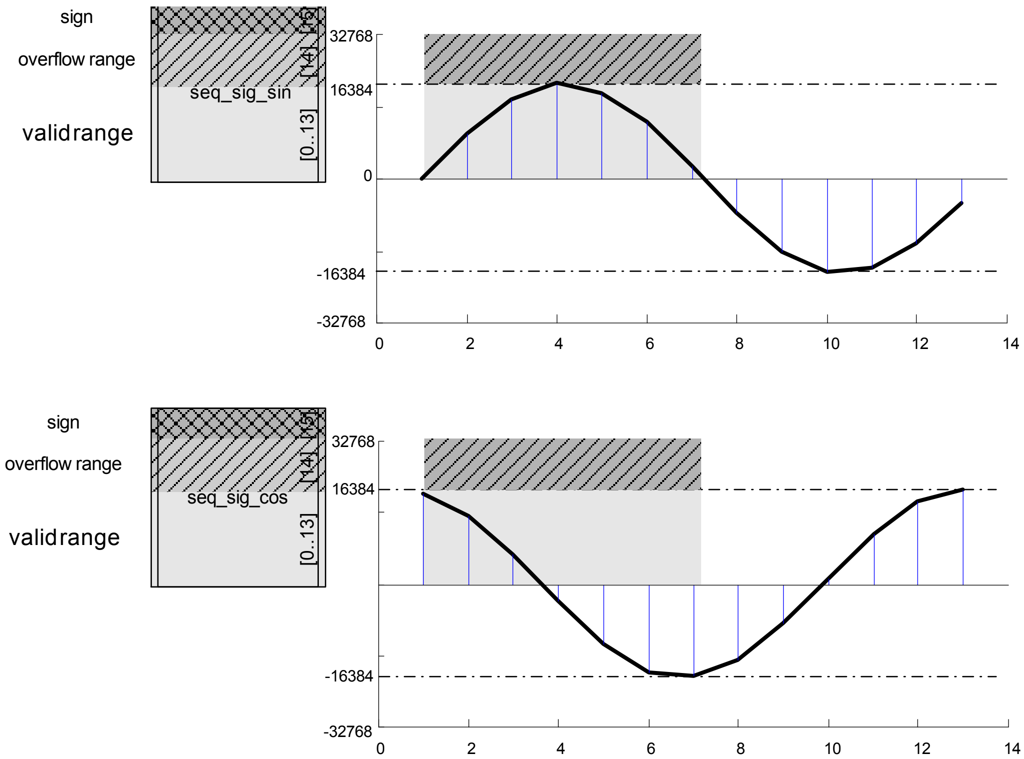


Figure 9.1: Range for sine wave and cosine wave.

9.1.1 Programming the Sine Generator

The sine steps are calculated using the following equations:

$$S_{n+1} = S_n + \frac{C_n}{2^k}$$

$$C_{n+1} = C_n - \frac{S_{n+1}}{2^k}$$

S_n and C_n are sine- and cosine amplitude of the sine wave and n the sequence of microsteps with n=1..Smax. Smax is the number of sine steps to be generated. k is the resolution control parameter, given by the register *seq_reg_shift*.

These equations are built from a digital differential equation. The three parameters S_n, C_n and k have an influence on this equation. With a fixed register width one has to calculate using a restricted resolution, so that different numbers of steps per wave result from rounding errors for different amplitudes.

Therefore the following rules should be obeyed when programming the sine generator:

The higher the amplitude is chosen, the better the sine resolution and the smoother the sine waves become. Resulting from the differential equation three different behavioral patterns of the sine generator can occur:

With the amplitude difference fulfilling [*maximum value*] $\geq S_n - C_n \geq 2^{k+1}$, sine waves result.

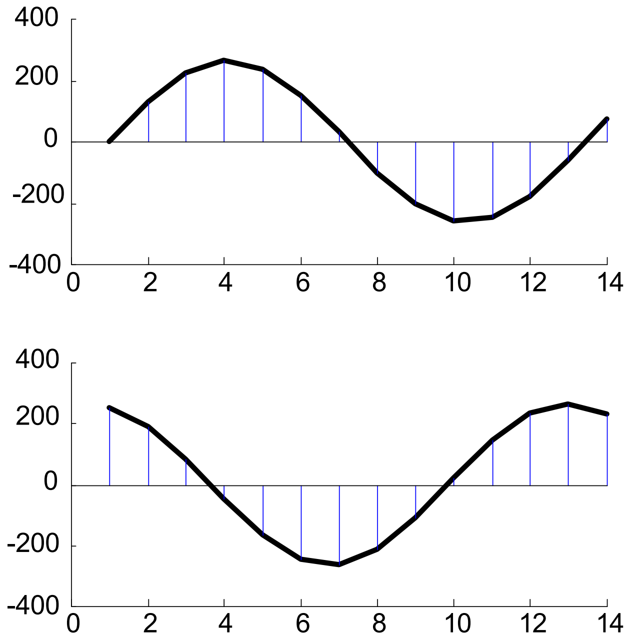


Figure 9.2: Amplitude vs. step number. Sinusoidal curve with k=1 and S_n-C_n = 255

The smaller the amplitude difference is, the coarser the wave becomes. Figure 9.3 shows clearly that the curves get coarser and that the number of steps for the full wave has increased slightly.

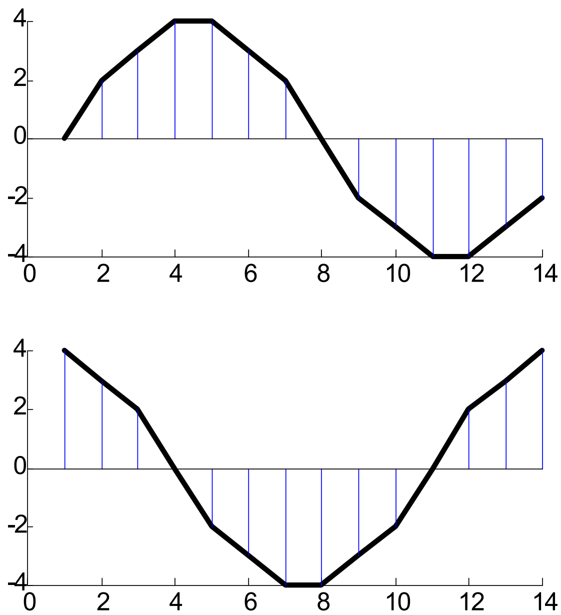


Figure 9.3: Amplitude vs. step number. Near sine curve with k=1 and S_n-C_n = 4

When the amplitude difference becomes too small (with $\mathbf{2^{k+1} > S_n\text{-}C_n \geq 2^k}$), a linear rising resp. falling curve results instead of the sine wave. Figure 9.4 shows this case.

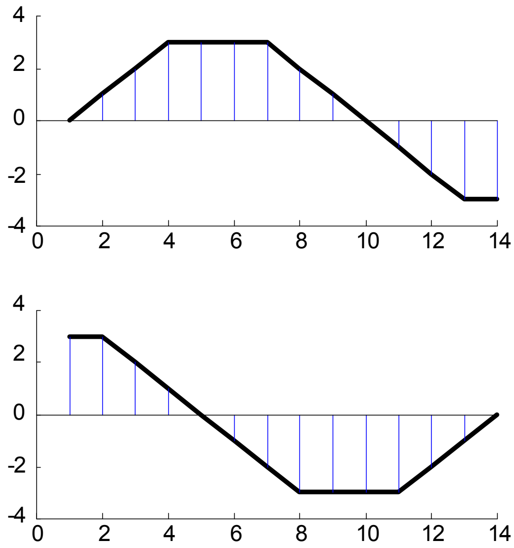


Figure 9.4: Amplitude vs. step number. Linear Wave with k=1 and S_n-C_n = 3

If the amplitude difference becomes smaller than 2^k a constant wave results.

For a sinusoidal curve the following heuristical formula shows the calculation of the number of steps per wave:

$N = 2^{(k+2)} + 2 \cdot 2^k + k$

N is the number of steps in a full sine wave.
The full wave shown in Figure 9.2 thus has N = 2^3+4+1 =13 steps.

register value **seq_reg_shift**	number of steps per full wave
1	13
2	26
3	51
4	100
5	197
6	406

Since the internal DACs have a resolution of only 8 bit, values of more than 5 for the s*eq_reg_shift* register do not lead to more sine steps, but, transferred to the motor, to a slower and smoother movement.

Output of the sine steps via the port:
In sine step operation the full steps for the control of the phase polarity are directly taken from the sine waves.

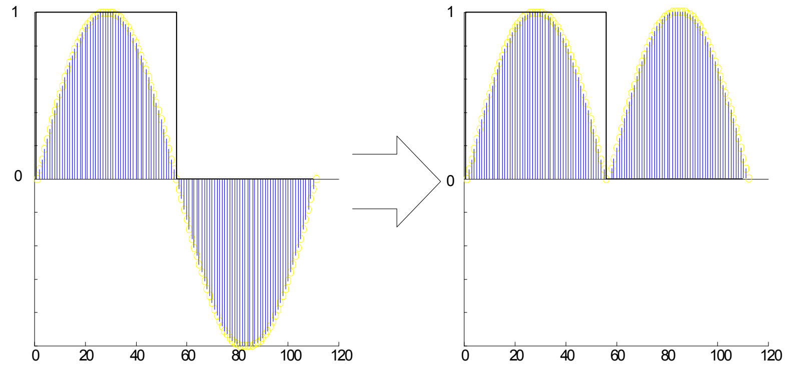


Figure 9.5: Derivation of the phase polarity in sine step mode

Figure 9.6 illustrates the mapping of the outputs. The analog outputs DAC0OUT and DAC1OUT output the absolute value of the respective wave (s. Figure 9.5). STO0 and STO2 are controlled by the respective signs.

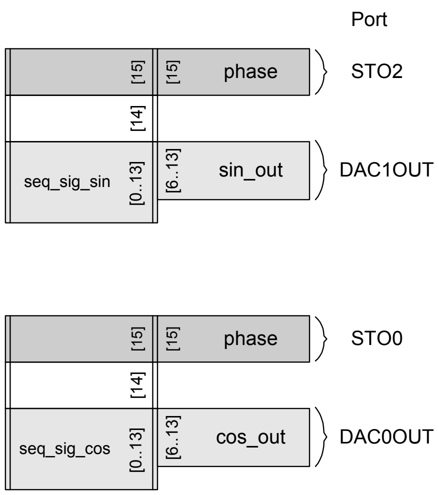


Figure 9.6: Output mapping in sine step mode

For the activation of sine step mode the following setup has to be done in the configuration register *seq_config*: `[SO_PHASE_OR_SIN = 0; SINUS_GEN_ON = 1; PHASE_GEN_ON = 0; SET_AUTO_PHASE_PATTERN = 0; SET_AUTO_PHASE_INDEX = 0; SINE_OUTPUT_TYPE = 0; TRIGGER_TYPE = 00; STEP_TYPE = 11; MOTOR_TYPE = 00]`

The bit **SO_PHASE_OR_SIN** allows to output the internally generated sine wave on the outputs STO0-STO9.

9.2 Full Step and Half Step Operation

9.2.1 Automatic Phase Pattern Setup

The sequencer contains freely programmable registers for the generation of halfstep and fullstep patterns. For the common motors with a bipolar drive, the phase patterns are stored in the TMC454 and can be accessed via the register *seq_config*.

For the automatic phase pattern setup, the motor type has to be identified with the bits MOTOR_TYPE and the mode of operation via the bits STEP_TYPE. Before programming the phase patterns, the ramp generator has to be disabled by clearing the bit PHASE_GEN_ON. Only in inactive state the automatic phase patterns can be recalled. By setting the bits SET_AUTO_PHASE_PATTERN and SET_AUTO_PHASE_INDEX the corresponding phase patterns are selected and the phase pointers are loaded into the registers *seq_phase1_index, seq_phase2_index, seq_phase3_index, seq_phase4_index* and *seq_phase5_index***.**

9.2.2 Manual Phase Pattern Setup

For manual programming of the phase patterns the following registers have to be programmed:

seq_phase1_index: 4 bit (address: 0x28)
seq_phase2_index: 4 bit (address: 0x29)
seq_phase3_index: 4 bit (address: 0x2A)
seq_phase4_index: 4 bit (address: 0x2B)
seq_phase5_index: 4 bit (address: 0x2C)

Bit	Term	Description
0-4	PHASE1_INDEX PHASE2_INDEX PHASE3_INDEX PHASE4_INDEX PHASE5_INDEX	Sets the corresponding index pointer for the phase pattern. The index pointer addresses a bit in the register **seq_phase_pattern**. The valid range is 0 to 19. **These registers can only be programmed when the phase generator is disabled by clearing the bit PHASE_GEN_ON (Register *seq_config*).**

seq_phase_pattern: 20 bit (address: 0x30)

Bit	Term	Description
0-19	PHASE_PATTERN	Defines the phase pattern for fullstep generation. Used for 2-,3- and 5-phase motors.

seq_dis_current: 20 bit (address: 0x34)

Bit	Term	Description
0-19	CURRENT_PATTERN	Defines the additional phase patterns for halfstep generation. (Disable of single coils) Used for 2-,3- and 5-phase motors.

Principle of phase pattern generation

The 20 bit wide registers *seq_phase_pattern* and *seq_dis_current* are used for the basic phase patterns. The registers *seq_phase1_index, seq_phase2_index, seq_phase3_index, seq_phase4_index* and *seq_phase5_index* are used as pointers. The addressed bits in the basic phase patterns (*seq_phase_pattern, seq_dis_current*) are directly output at STO0-STO9.
The register *seq_phase_pattern* controls the phase polarity in fullstep and halfstep operation while the register *seq_dis_current* controls the current disable in halfstep operation. Figure 9.7 shows a motor coil driver for halfstep operation. The **phase** input is controlled by the corresponding index pointer pointing to one bit in the register *seq_phase_pattern*. The **disable** input is controlled by the same index pointing to one bit in the register *seq_dis_current*.

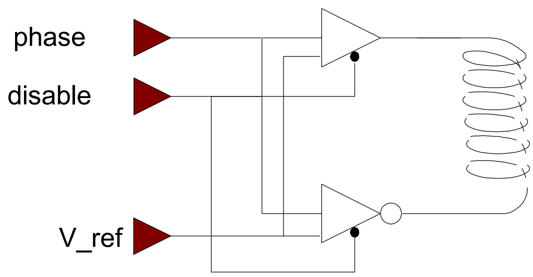


Figure 9.7: Driving a coil

Figure 9.8 shows an example for a pattern and pointer configuration. In every step the pointers are incremented respectively decremented depending on the direction. The increment resp. decrement is automatically calculated as modulus of the length of the phase pattern.

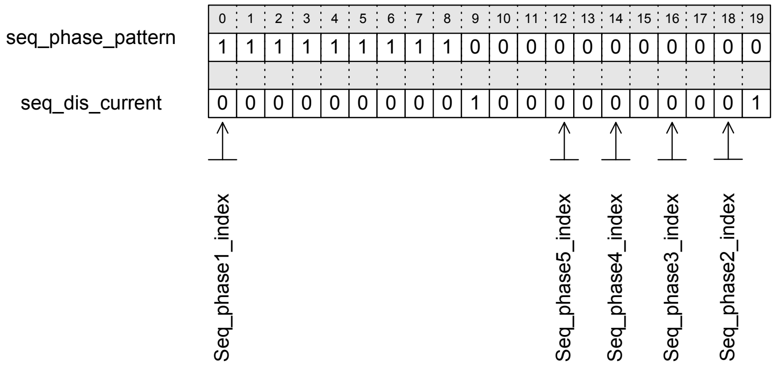


Figure 9.8: Example for user programmed phase pattern (5-phase motor in half step op.)

The following table shows the phase patterns stored in the TMC454 for the different operation modes:

basic phase pattern seq_phase_pattern	**disable pattern seq_dis_current**	pointer (Z = 1..5) **seq_phase(Z)_index**	maximum length of phase pattern
2 phase / fullstep			
00000000000000000011	**00000000000000000000**	Z1: 0 Z2: 3	4
2 phase / halfstep			
00000000000000000111	**00000000000010001000**	Z1: 0 Z2: 6	8
3 phase / fullstep			
00000000000000000011 (*)	**00000000000000100100**	Z1: 0 Z2: 2 Z3: 4	6
3 phase / halfstep			
00000000000000011111	**00000000100000100000**	Z1: 0 Z2: 4 Z3: 8	12
5 phase / fullstep			
00000000000000001111	**00000000001000010000**	Z1: 0 Z2: 9 Z3: 8 Z4: 7 Z5: 6	10
5 phase / halfstep			
00000000000111111111	**10000000001000000000**	Z1: 0 Z2: 18 Z3: 16 Z4: 14 Z5: 12	20

(*)The three phase motor fullstep pattern in the TMC454 is a modified halfstep pattern. For microstep operation the user should program a phase pattern of “0...0111”

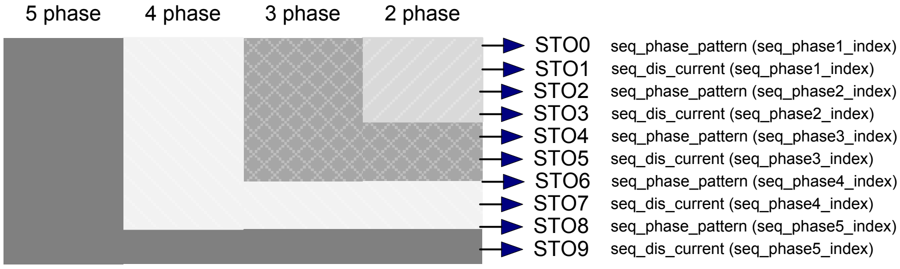


Figure 9.9: Port mapping in the different modes of operation

Figure 9.9 shows how the ports are controlled depending on phase count and step mode. In 2-, 3- and 4-phase operation not all STO port bits are used. The free outputs can be user programmed via the register *seq_phase_pattern*.

The following table shows the STO port bits available in dependence of the operation mode:

MOTOR_TYPE	Free STO ports / corresponding register bits
2-phase motor	STO4 = *seq_phase_pattern(14)* STO5 = *seq_phase_pattern(15)* STO6 = *seq_phase_pattern(16)* STO7 = *seq_phase_pattern(17)* STO8 = *seq_phase_pattern(18)* STO9 = *seq_phase_pattern(19)*
3-phase motor	STO6 = *seq_phase_pattern(16)* STO7 = *seq_phase_pattern(17)* STO8 = *seq_phase_pattern(18)* STO9 = *seq_phase_pattern(19)*
4-phase motor	STO8 = *seq_phase_pattern(18)* STO9 = *seq_phase_pattern(19)*

The register *seq_config* configures the sequencer for the desired mode, step type and port mapping. In normal operation the internal pulse generator is used to generate the stepping clock. Additionally, it is possible to select different external sources for the stepping clock. The bits TRIGGER_TYPE allow to select other sources, like the incremental encoder.

seq_config: 12 bit (address: 0x38)		
Bit	**Term**	**Description**
0-1	MOTOR_TYPE	Sets the motor type for phase pattern generation: 00: 2 phase motor 01: 3 phase motor 10: 4 phase motor 11: 5 phase motor **Reset value: 00**
2-3	STEP_TYPE	Controls the step type for the selected motor: 00: fullstep operation 01: halfstep operation 10: microstep operation 11: sinestep operation **Reset value: 00**
4-5	TRIGGER_TYPE	Controls the pulse source for the stepping clock: 00: internal pulse generator (controlled by ramp generator) 01: incremental encoder (divided by the encoder predivider) 10: external signals STEP_IN and DIR_IN 11: external signals CHA (step) and CHB (direction) **Reset value: 00**
6	SINE_OUTPUT_TYPE	Controls the sine wave output: 0: Absolute value and sign 1: Signed sine wave **Reset value: 0**
7	SET_AUTO_PHASE_INDEX	Controls the automatic loading of the index pointers: 1 : Load automatic pointer values 0 : No operation **Condition: Can only be used, with PHASE_GEN_ON = 0.** **This register resets automatically.**
8	SET_AUTO_PHASE_PATTERN	Controls the phase pattern source (automatic / user programmed): 1: Use automatic phase patterns 0: Use user programmed phase patterns **Reset value: 0**
9	PHASE_GEN_ON	Activates generation of the phase patterns: 1: On 0: Off **Reset value: 0**
10	SINUS_GEN_ON	Activates the sinestep mode 1: On 0: Off **Reset value: 0**
11	SO_PHASE_OR_SIN	Sets the outputs STO0-STO9 to output the phase signals or the sinewave 0: Output of phase signals 1: Output of sinewave **Reset value: 0**

9.3 Registers for Microstep Operation

User defined microsteps for 2- and 3-phase motors can be generated using the internal RAM. This allows programming of a microstep pattern adapted to the characteristics of the motor. Up to 128 microstep values can be stored in the internal RAM with a resolution of 8 bits. Only one half wave is stored in the RAM. Thus up to 64 user defined microsteps are possible between each two fullsteps. It is also possible to store multiple half-waves or up to three different half-waves for the different coils of the motor. Further the microstep RAM can store user defined digital patterns.
For each of the three DACs in the TMC454 one pointer (mstep_phase0_cnt, mstep_phase1_cnt, mstep_phase2_cnt) points to a location in the microstep RAM. The pointers are increased resp. decreased with every motor step.

mstep_ram_adr: 7 bit w (address: 0xE0)		
Bit	**Term**	**Description**
0-6	RAM_ADR	Address register for the access to the internal RAM. Write directly before write / read access to the RAM.

mstep_ram_data: 8 bit rw (address: 0xE1)		
Bit	**Term**	**Description**
0-6	RAM_DATA	Data register for RAM access.

mstep_table_end: 7 bit rw (address: 0xE4)		
Bit	**Term**	**Description**
0-6	TABLE_END	Defines the end of the user programmed microstep table. The RAM pointers count with a modulus of MSTEP_TABLE_END+1.

mstep_phase0_cnt: 7 bit rw (address: 0xE5) **mstep_phase1_cnt: 7 bit rw (address: 0xE6)** **mstep_phase2_cnt: 7 bit rw (address: 0xE7)**		
Bit	**Term**	**Description**
0-6	PHASE0_CNT PHASE1_CNT PHASE2_CNT	Pointer to the microstep table. The addressed RAM byte can be output to the corresponding DAC. The pointers increase / decrease with every step.

mstep_full_step_dist: 7 bit rw (address: 0xE8)		
Bit	**Term**	**Description**
0-6	FULL_STEP_DIST	Fullstep distance for microstep generation. Defines the microstep count (-1) after which the fullstep sequencer has to be clocked.

mstep_cnt_full_step_dist: 7 bit rw (address: 0xE9)		
Bit	**Term**	**Description**
0-6	CNT_FULL_STEP_DIST	Counts the number of microsteps. On underflow or exceeding the fullstep distance (FULL_STEP_DIST), the fullstep sequencer is clocked.

mstep_conf: 1bit (address: 0xEA)		
Bit	**Term**	**Description**
1	TABLE_DIV	Sets the configuration for the microstep table: 0: The RAM is addressed in its full length. The register mstep_table_end defines the length of the table. The pointers phase0_cnt, phase1_cnt and phase2_cnt address the whole table. 1: The RAM is addressed as 3 separate areas. Each area can contain a table. The first area begins at address 0 and ends at the address defined by the register mstep_table_end. The second area begins at the address given by the register mstep_table_end + 1 and ends at the address 2* mstep_table_end +1. The third area begins at the address 2* mstep_table_end +2 and ends at the address 3* mstep_table_end +2. The pointers phase0_cnt, phase1_cnt and phase2_cnt count cyclic in their areas. When the third area is not needed, the mstep_table_end can be set at up to address 63 to use half of the RAM for each of two tables.

Note:

All registers for microstep operation (except for the RAM) can only be programmed while the sequencer is off (PHASE_GEN_ON=0). The microstep generation is enabled by the bit PHASE_GEN_ON in register *seq_config*.

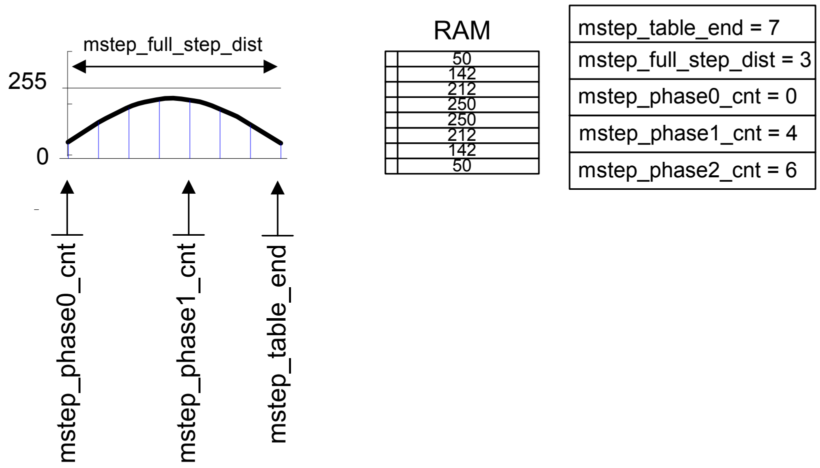


Figure 9.10: Using the microstep RAM (Example for a 2 phase motor)

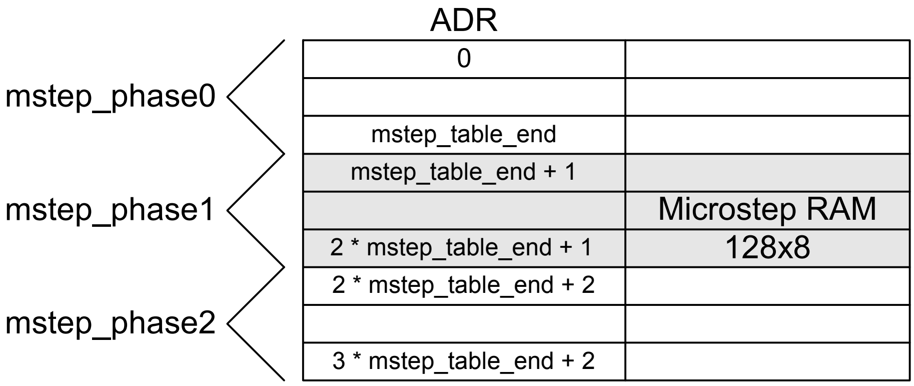


Figure 9.11: Organization of the microstep RAM when using three separate tables.

For microstep operation via the microstep RAM, the fullstep distance has to be programmed into the register MSTEP_FULL_STEP_DIST. The fullstep distance controls after how many microsteps a new fullstep is generated. After each fullstep the sequencer then switches to the next phase pattern. Thus the corresponding phase pattern has to be programmed into the sequencer before the microstep mode can be used (see chapter 9.2Full Step and Half Step Operation).

<u>Note</u>:
In TMC454 writing to the microstep RAM is only reliable when the motor stands still (sequencer off or no pulses). When the microstep RAM has to be changed during operation of the motor, multiple write accesses could be necessary to change the contents of RAM locations.

9.4 Administration of the different modes of operation and output control

The TMC454 supports all kinds off stepper motors and thus has a number of different modes of operation. The outputs have to be configured differently depending on the mode. Some operation modes need both the digital and the analog outputs. Unused outputs can be used as general purpose output.

The analog outputs are controlled via the module analog motor control.
The registers *pmap_dac0*, *pmap_dac1* and *pmap_dac2* allow direct control of the output voltages for all three DACs.

pmap_dac0: 8 bit rw (address: 0x90)

Bit	Term	Description
0-7	DAC0_REG_DIRECT	Controls the output value for DAC0OUT in direct mode.

pmap_dac1: 8 bit rw (address: 0x91)

Bit	Term	Description
0-7	DAC1_REG_DIRECT	Controls the output value for DAC1OUT in direct mode.

pmap_dac2: 8 bit rw (address: 0x92)

Bit	Term	Description
0-7	DAC2_REG_DIRECT	Controls the output value for DAC2OUT in direct mode.

pmap_mc: 2 bit rw (address: 0x93)

Bit	Term	Description
0,1	MC_REG_DIRECT	Controls the outputs MC0 and MC1 in direct mode.

The register pmap_conf controls, which unit operates the analog outputs (DAC0OUT, DAC1OUT, DAC2OUT).

pmap_conf: 6 bit rw (address: 0x94)		
Bit	**Term**	**Description**
0	ENABLE_DAC0_REG_DIRECT	Controls the analog output DAC0OUT. When sinestep operation is active, (s.register *seq_config*) the cosine wave is output. In microstep mode, the value of the microstep RAM addressed by *mstep_phase0_cnt* (s.Register *seq_config*) is output. If neither mode is active this flag selects: 1: The output is controlled by the value in *pmap_dac0*. 0: The output gives the microstep table contents.
1	ENABLE_DAC1_REG_DIRECT	Controls the analog output DAC1OUT. When sinestep operation is active, (s.register *seq_config*) the sine wave is output. In microstep mode, the value of the microstep RAM addressed by *mstep_phase1_cnt* (s.Register *seq_config*) is output. If neither mode is active this flag selects: 1: The output is controlled by the value in *pmap_dac1*. 0: The output gives the microstep table contents.
2	ENABLE_DAC2_REG_DIRECT	Controls the analog output DAC2OUT. When the ENABLE_STO_VOUT_DIRECT is set, the actual velocity value is output via DAC3. It is output as absolute value (MSB-bound: bits 12-5). In microstep mode, the value of the microstep RAM addressed by *mstep_phase2_cnt* (s.Register *seq_config*) is output, when a 3-phase motor is selected. If neither mode is active this flag selects: 1: The output is controlled by the value in *pmap_dac2*. 0: The output is controlled by the automatic motor current control unit (described in the chapter on the FIFO).
3	ENABLE_MC0/MC1_REG_DIRECT	0: The digital outputs MC0 and MC1 are controlled by the current control unit. 1: The digital outputs MC0 and MC1 are controlled by the register *pmap_mc*.
4	ENABLE_DAC2_VOUT_DIRECT	0: DAC2OUT is controlled by the standard functions. 1: The actual absolute velocity value is output via DAC2OUT **This function is useful for servo motor control!**
5	ENABLE_STO_VOUT_DIRECT	0: STO0-STO9 is controlled by the standard functions. 1: The actual velocity value is output via STO0-STO9 (MSB-bound: bits 12-3). **This function is useful for servo motor control!**

The digital outputs STO0-STO9 are controlled according to the chosen function:
They are influenced by the following registers:
- seq_sot_output_select selects the microstep RAM or the sequencer as source.
- ENABLE_STO_VOUT_DIRECT in register *pmap_conf* switches the velocity value to the outputs.

The different combinations are shown in the following table:

seq_sto_output_select: 2 bit rw (address: 0x2D)		
Bit	**Term**	**Description**
0-1	STO_OUTPUT_SELECT	Configures the digital outputs STO0-STO9 With ENABLE_STO_VOUT_DIRECT = 0: 01: 10 bit value from the microstep RAM (bits 9-0) 10: 10 bit value from the microstep RAM (bits 17-8) 11: 10 bit value from the microstep RAM (bits 23-14) 00: Control via sequencer When ENABLE_STO_VOUT_DIRECT = 1, the actual velocity is output as 10 bit signed integer.

The selection of bit patterns from the microstep RAM is done by the RAM pointers (mstep_phase0_cnt, mstep_phase1_cnt, mstep_phase2_cnt). Each pointer addresses an 8 bit value from the RAM. The combination of all three values result in a 24 bit wide value. This value can be output at the digital outputs (STO0-STO9) as chosen via the register *seq_sto_output_select*.

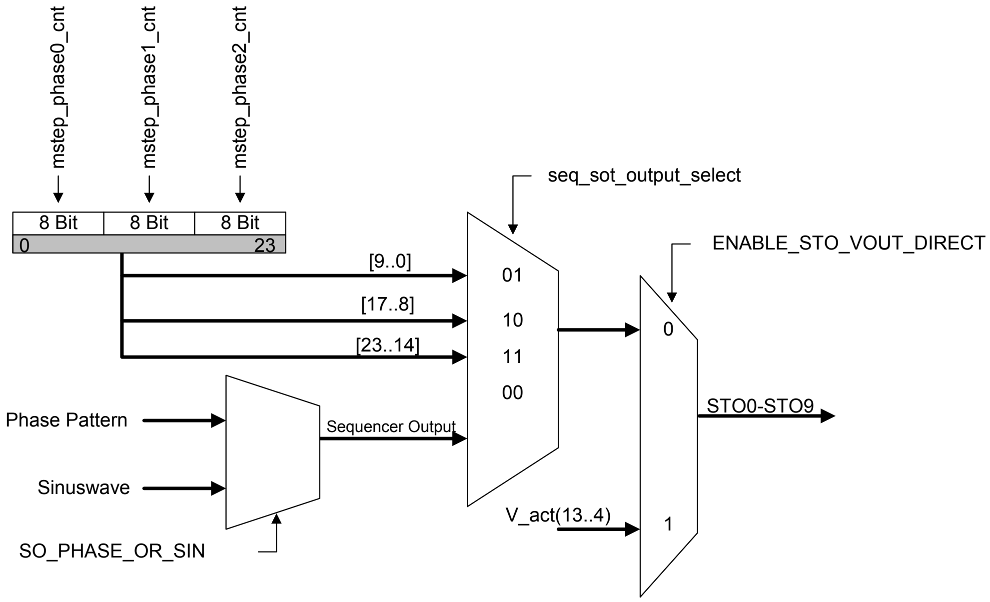


Figure 9.12: Control of the digital outputs STO0-STO9

10 The PID Controller

10.1 General introduction

This module allows feedback control for position stabilization. This is only possible in connection with an incremental encoder as feedback for the actual position. The PID controller allows positioning in encoder steps, provided that the motor resolution is set high enough. The idea of the regulator is a generalization of the micro step function. The regulator will try to correct the position of the motor, if a difference between the generated desired ramp position and the actual, measured position occurs. Because of the inherent time delays in the loop TMC454-motor-encoder-TMC454, this type of regulation can cause oscillations of the system. The delay time of this loop inside the TMC454 has to be minimized. However a careful adaptation of the regulation characteristic to the mechanical characteristics of the application is necessary to optimize the regulation stability and speed.

The main function of the regulator is to enforce the actual position calculated by the ramp generator at every point of time as precisely as possible using the position indicated by the incremental encoder. This includes compensation of movements caused by varying loads. The TMC454 additionally calculates the actual motor speed from the encoder signal.

In dependence of the calculated position difference and a set of parameters the PID controller calculates a correction velocity which is added to the actual velocity of the ramp generator and then used to clock the sequencer. As an option the velocity calculated from the incremental encoder can be used as velocity base instead of the ramp generator velocity. This mode allows load adaptive motor driving.

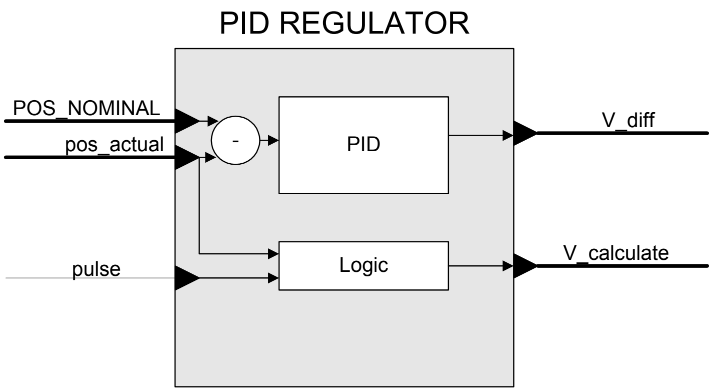


Figure 10.1: Simplified schematic the PID controller

10.1.1 Increasing stepping accuracy and stabilizing the position

Basically the regulator is intended to increase the stepping accuracy and to stabilize the motor against varying mechanical loads. A proportional filter (p-filter) weighs the difference between the actual (encoder) position and the desired (ramp) position. The result is used to influence the motor via the PID speed correction input (v_diff) of the pulse generator. To limit the effect of this speed correction to a relatively small value in the case of a failure, the difference is symmetrically clipped to a programmable bound.

10.2 Description of the registers of the PID controller

pid_control: 9 bit (address: 0xA0h)

Bit	Term	Description
0-2	PID_CLK	Controls the pre-scaler for PID controller operating frequency 000 : Ramp generator frequency / 1 001 : ''/2 010 : ''/3 011 : ''/4 100 : ''/8 101 : ''/16 110 : ''/32 111 : ''/64 **Note:** *At least 10 system clocks are needed for internal calculations of the PID controller.*
3-5	DIFF_PID_CLK	Controls the clock divider for the calculation of the differential part. 000: PID clock /1 100: PID clock /32 001: PID clock /4 101: PID clock /64 010: PID clock /8 110: PID clock /128 011: PID clock /16 111: PID clock /256 The position error is sampled with the divided clock to get the difference.
6-8	V_CALC_PID	Controls the clock divider for the calculation of the velocity from the encoder position. 000 : PID clock /1 100 : PID clock /32 001 : PID clock /4 101 : PID clock /64 010 : PID clock /8 110 : PID clock /128 011 : PID clock /16 111 : PID clock /256 **Note:** *At least 48 system clocks are needed for internal calculations*.

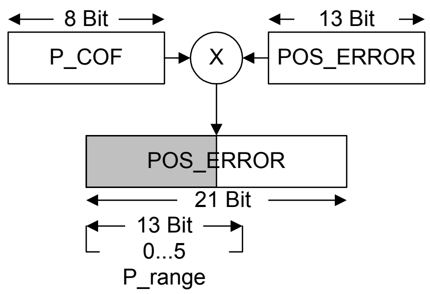
pid_vcalc_factor: 12 bit (address: 0xA2h)

Bit	Term	Description
0-11	V_CALCULATE_FACTOR	Defines the factor for the calculated velocity value. The factor can be calculated as follows: V_CALCULATE_FACTOR = V_NOM / V_ACT The factor always has to match the system configuration, because it is dependant on the resolution of the incremental encoder and the clock frequency settings. V_calculate = V_calc * V_CALCULATE_FACTOR

pid_pcof: 8 bit (address: 0xA4h)

Bit	Term	Description
0-7	PROPORTIONAL_COEFFICIENT	Controls the influence of the proportional part of the PID controller. The position error is multiplied with this coefficient.

pid_p_range: 3 bit (address: 0xA5h)

Bit	Term	Description
0-2	PROPORTIONAL_RANGE	Scales the 21 bit weighed proportional part by selecting a 13 bit range. Allowed values: 0..5. Divides by 2^(x+3). 0=no division. 

pid_icof: 8 bit (address: 0xA8h)

Bit	Term	Description
0-8	INTEGRAL_COEFFICIENT	Controls the influence of the integral part of the PID controller. The contents of the scaled integral register is multiplied with this coefficient and then divided by 8.

pid_i_range: 4 bit (address: 0xA9h)

Bit	Term	Description
0-3	INTEGRAL_ RANGE	Scales the 20 bit integral part by selecting an 8 bit range. Allowed values: 0..11. Divides by 2^x. 0=no division, select bits 0..7. (compare schematic for proportional part)

pid_dcof: 8 bit (address: 0xACh)

Bit	Term	Description
0-7	DIFFERENTIAL_COEFFICIENT	Controls the influence of the differential part of the PID controller. The scaled diffe-rence between the last two errors is multiplied with this value and then divided by 8.

pid_d_range: 3 bit (address: 0xADh)

Bit	Term	Description
0-2	DIFFERENTIAL_ RANGE	Scales the 16 bit weighed differential part by selecting an 8 bit range. Allowed values: 0..5. Divides by 2^x. 0=no division (advised value is 0). (compare schematic for proportional part)

pid_clip_i: 8 bit (address: 0xB0h)

Bit	Term	Description
0-7	CLIPPING_ INTEGRATION_VALUE	Sets the clipping for the integration part (upper 8 bits of absolute value).

pid_clip_p: 8 bit (address: 0xB4h)

Bit	Term	Description
0-7	CLIPPING_ PROPORTIONAL_VALUE	Sets the clipping for the proportional part (upper 8 bits of absolute value).

pid_clip_d: 8 bit (address: 0xB8h)

Bit	Term	Description
0-7	CLIPPING_ DIFFERENTIAL_VALUE	Sets the clipping for the differential part (upper 8 bits of absolute value).

pid_clip_int_sum: 8 bit (address: 0xB9h)		
Bit	**Term**	**Description**
0-7	CLIPPING_ INTEGRAL _SUM	Sets the clipping for the 20 bit signed integration register (upper 8 bits of absolute value).

pid_clip_int_input: 8 bit (address: 0xBAh)		
Bit	**Term**	**Description**
0-7	CLIPPING_ INTEGRAL_INPUT	Sets the clipping for the integrator input (upper 8 bits of absolute value).

pid_clip_sum: 13 bit (address: 0xBCh)		
Bit	**Term**	**Description**
0-12	CLIPPING_ REGULATOR_SUM	Sets the clipping for the PID controller output (maximum influence on velocity).

pid_int_sum_reg: 20 bit (address: 0xC0h)		
Bit	**Term**	**Description**
0-19	INTEGRATION_SUM	Integration register, 20 bit signed

pid_error_n: 14 bit r (address: 0xC4h)		
Bit	**Term**	**Description**
0-13	ERROR_ACT	Actual input value of the PID controller (Error: Difference between *ENC_COUNT* and *RAMP_POS_ACT*), 14 bit signed

pid_error_pre: 14 bit r (address: 0xC6h)		
Bit	**Term**	**Description**
0-13	ERROR_PRE	Previous position error for calculation of differential part, 14 bit signed

pid_v_diff_out: 14 bit r (address: 0xC8h)		
Bit	**Term**	**Description**
0-13	VELOCITY_DIFFERENCE	Velocity difference calculated by PID controller, 14 bit signed

pid_v_calc: 14 bit r (address: 0xCAh)		
Bit	**Term**	**Description**
0-13	VELOCITY _CALCULATED	Velocity calculated from the incremental encoder, 14 bit signed

The following schematic shows the functions and the control registers of the PID module.

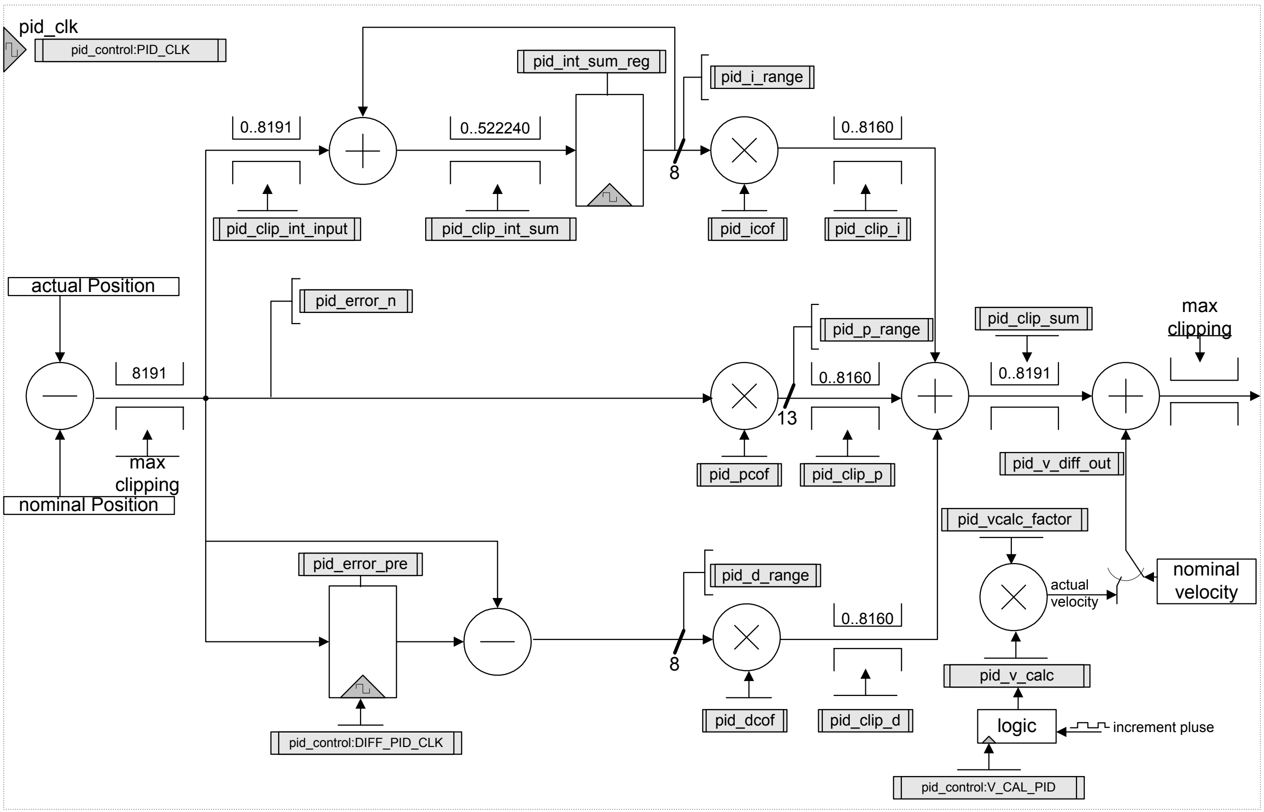


Figure 10.2: PID controller and registers

Note on programming the PID coefficients:
When programming the PID proportional coefficient register and range register, always set the coefficient between 128 and 255 to achieve the highest possible resolution. A coefficient below 128 should be programmed with the double value for the coefficient register and the corresponding range register increased by one. This is necessary, because the multiplication is done before the division.
When programming the PID integral and differential registers, always minimize the values in the range registers, because division here is done before multiplication!

11 Interrupt control and Interrupt Sources

The interrupt controller supports 11 interrupt sources. The interrupt is edge controlled and the polarity of the interrupt signal can be programmed. The interrupt output polarity is positive active by default. Setting the IRQ_POLARITY bit, makes the output negative active.

Table of interrupt signals

Interrupt	Source	Description
Left/right stop switch	external signal	Activation/deactivation of left/right stop switch. (The motor can be stopped on switch activation when the direction corresponds to the switch.)
Left/right slowdown switch	external signal	Activation/deactivation of left/right slowdown switch. (The motor can be slowed down on switch activation when the direction corresponds to the switch.)
Channel N	external signal	Activation/deactivation of the incremental encoder null signal.
FIFO empty	internal signal	FIFO is empty.
Stop condition	internal signal	The TMC454 has accepted a stop condition.
FIFO interrupt set	internal signal	Execution of a FIFO command with a set interrupt bit has started.
Overflow of encoder position counter	internal signal	The incremental encoder counter has had an overflow.
Time out	internal signal	The programmed time limit has been reached. (The timer has exceeded the programmed compare value.)
Position deviation exceeded	internal signal	The deviation between encoder counter and ramp position has exceeded the programmed maximum.

irq_polarity 2 bit rw (address: 0xD0h)		
Bit	**Term**	**Description**
0	STOPL_POLARITY	Polarity of NSTOPL input for interrupt generation
1	STOPR_POLARITY	Polarity of NSTOPR input for interrupt generation
2	SLDL_POLARITY	Polarity of NSLDL input for interrupt generation
3	SLDR_POLARITY	Polarity of NSLDR input for interrupt generation
4	CHN_POLARITY	Polarity of CHN input for interrupt generation
5	IRQ_POLARITY	Polarity of the INT output (Interrupt signal to host CPU)

irq_enable 11 bit rw (address: 0xD2h)		
Bit	**Term**	**Description**
0	EN_STOPL_INTERRUPT	Enable NSTOPL interrupt
1	EN_STOPR_INTERRUPT	Enable NSTOPR interrupt
2	EN_SLDL_INTERRUPT	Enable NSLDL interrupt
3	EN_SLDR_INTERRUPT	Enable NSLDR interrupt
4	EN_CHN_INTERRUPT	Enable CHN interrupt
5	EN_FIFO_EMPTY_INTERRUPT	Enable FIFO empty interrupt
6	EN_STOP_INTERRUPT	Enable stop condition interrupt
7	EN_FIFO_INTERRUPT	Enable FIFO command interrupt
8	EN_INCENC_OVERFLOW_INTERRUPT	Enable incremental encoder overflow interrupt
9	EN_TIMER_INTERRUPT	Enable timeout interrupt
10	EN_DEVIATION_INTERRUPT	Enable position deviation interrupt for incremental encoder

irq_status 11 bit rw(*) (address: 0xD4h)		
Bit	**Term**	**Description**
0	STATUS_STOPL_INTERRUPT	Interrupt status for NSTOPL
1	STATUS _STOPR_INTERRUPT	Interrupt status for NSTOPR
2	STATUS _SLDL_INTERRUPT	Interrupt status for SLDL
3	STATUS _SLDR_INTERRUPT	Interrupt status for SLDR
4	STATUS _CHN_INTERRUPT	Interrupt status for CHN
5	STATUS _FIFO_EMPTY_INTERRUPT	Interrupt status for FIFO empty
6	STATUS _STOP_INTERRUPT	Interrupt status for stop condition
7	STATUS _FIFO_INTERRUPT	Interrupt status for FIFO command interrupt
8	STATUS_INCENC_OVERFLOW_INTERRUPT	Interrupt status for incremental encoder overflow
9	STATUS _TIMER_INTERRUPT	Interrupt status for timeout
10	STATUS _DEVIATION_INTERRUPT	Interrupt status for position deviation

(*) The interrupt status flags can be cleared selectively by writing a 1-bit to the desired bit positions.

12 TMC454 Register Overview

The TMC454 has an extensive set of functions controlled by different kinds of registers. There are status registers (usually read only), command registers (usually write only), control registers, configuration registers (usually unchanged during operation) and registers for values. Each register belongs to a specific module in the TMC454. The term for the register begins with an abbreviation for the register name as detailed in the following table.

Prefix	Module name
fifo	Command FIFO
enc	Incremental Encoder Interface
seq	Sequencer
ramp	Ramp Generator
pmap	Port Mapper
pid	PID Controller
iiq	Interrupt Controller
mstep	Micro Step RAM

ADR (hex)	Term	width (read / write)	Description
0x0	fifo_command	32 (rw)	Command register of the command FIFO Write: Set new command Read: Read command in execution
0x4	fifo_status	14 (r)	Status register of the command FIFO
0x6	fifo_input_status	5 (r)	Status register for the command FIFO input signals State of the digital inputs (NSLDR, NSLDL,NSTOPL,NSTOPR,SYNCIN)
0x7	fifo_port_func	2 (rw)	Control register Activates the digital inputs for slowdown and stop switches
0x10	enc_control	7 (rw)	Configuration register of the incremental encoder
0x11	enc_portstat	3 (r)	Status register Actual status of the digital inputs (CHA,CHB,CHC)
0x14	enc_count	24 (rw)	Status register Actual incremental encoder position
0x18	enc_holdreg	24 (r)	Holding register for *enc_count*
0x1C	enc_prediv_cnt	8 (rw)	Status register Encoder pre-divider counter
0x1D	enc_prediv_ratio	8 (rw)	Encoder pre-divider ratio
0x1E	enc_deviation	12 (rw)	Maximum deviation of encoder position from the actual ramp generator position for interrupt generation
0x24	seq_sig_sin	16 (rw)	Actual sine value of the sine generator / Amplitude control
0x26	seq_sig_cos	16 (rw)	Actual cosine value of the sine generator / Amplitude control
0x28	seq_phase1_index	5 (rw)	Configuration register for motor type Index pointer 1 for the programmable phase pattern Used for 2-,3- and 5-phase motors
0x29	seq_phase2_index	5 (rw)	Configuration register for motor type Index pointer 2 for the programmable phase pattern Used for 2-,3- and 5-phase motors
0x2A	seq_phase3_index	5 (rw)	Configuration register for motor type Index pointer 3 for the programmable phase pattern Used for 3- and 5-phase motors
0X2B	seq_phase4_index	5 (rw)	Configuration register for motor type Index pointer 4 for the programmable phase pattern

			Used for 5-phase motors
0x2C	seq_phase5_index	5 (rw)	Configuration register for motor type Index pointer 5 for the programmable phase pattern Used for 5-phase motors
0x2D	Seq_sot_output_selec t	2(rw)	Configuration register for motor type Source select for digital outputs SO0-SO9
0x2E	seq_reg_shift	4 (rw)	Configuration register for step resolution Controls the resolution of the sine wave
0x30	seq_phase_pattern	20 (rw)	Configuration register for motor type Defines the phase pattern for fullstep generation
0x34	seq_dis_current	20 (rw)	Configuration register for motor type Defines the phase-disable pattern for halfstep generation
0x38	seq_config	12 (rw)	Configuration register of the sequencer Defines the operation mode
0x3A	seq_sine_offset	8 (rw)	Optional offset for the sine wave in microstep operation
0x40	latch_ramp_params	0 (w)	Control register Any write access latches all ramp generator parameters into the holding registers
0x41	ramp_status	8 (r)	Status holding register of the ramp generator
0x44	ramp_time_cnt	24 (rw)	Holding register Internal timer (Read Access: Value of the holding register)
0x48	ramp_actvel	22 (r)	Holding register for the actual velocity
0x4C	ramp_actaccel	22 (r)	Holding register for the actual acceleration
0x50	ramp_pos_act	24 (r)	Holding register for the position counter
0x60	fifo_a_nom	14 (r)	Status register Nominal acceleration
0x62	fifo_v_nom	14 (r)	Status register Nominal velocity
0x64	fifo_a_sld	13 (r)	Status register Acceleration for slowdown
0x66	fifo_v_sld	13 (r)	Status register Maximum velocity after slowdown
0x68	fifo_bow14	14 (r)	Status register Bow parameter
0x6A	fifo_a_comp1	13 (r)	Status register Acceleration compare value for current control
0x6C	fifo_v_comp1	13 (r)	Status register Velocity compare value for current control
0x6E	fifo_a_comp2	13 (r)	Status register Acceleration compare value for current control
0x70	fifo_v_comp2	13 (r)	Status register Velocity compare value for current control
0x74	fifo_pre_div4	4 (r)	Status register Configuration of clock prescaler
0x75	fifo_imot0	8 (r)	Status register Output value 0 for AOUT2 automatic current control
0x76	fifo_imot1	8 (r)	Status register Output value 1 for AOUT2 automatic current control
0x77	fifo_imot2	8 (r)	Status register Output value 2 for AOUT2 automatic current control
0x78	fifo_imot3	8 (r)	Status register Output value 3 for AOUT2 automatic current control
0x79	fifo_misc_ctrl	2 (r)	Status register Flag for PID unit and evaluation of measured velocity
0x7C	fifo_t_lim	24 (r)	Status register Time limit for command execution / interrupt generation
0x80	fifo_pos_end	24 (r)	Status register End position for ramp segment
0x90	pmap_dac0	8 (rw)	Direct output value for DAC0
0x91	pmap_dac1	8 (rw)	Direct output value for DAC1

0x92	pmap_dac2	8 (rw)	Direct output value for DAC2
0x93	pmap_mc_2	2 (rw)	Direct output value for MC0 and MC1 pin
0x94	pmap_conf	6 (rw)	Configuration register for the analog motor controller Source selection for analog and digital outputs (AOUT0, AOUT1, AOUT2,MC0,MC1)
0xA0	pid_control	9 (rw)	Configuration register for the PID controller Control of the PID regulator operating frequencies
0xA2	pid_vcalc_factor	12 (rw)	Factor for internal velocity calculation
0xA4	pid_pcof	8 (rw)	Proportional part coefficient for PID regulator
0xA5	pid_p_range	3 (rw)	Proportional part scaling ($1/2^n$)
0xA8	pid_icof	8 (rw)	Integral part coefficient for PID regulator
0xA9	pid_i_range	4 (rw)	Integral part scaling ($1/2^n$)
0xAC	pid_dcof	8 (rw)	Differential part coefficient for PID regulator
0xAD	pid_d_range	3 (rw)	Differential part scaling ($1/2^n$)
0xB0	pid_clip_i	8 (rw)	Integral part clipping
0xB4	pid_clip_p	8 (rw)	Proportional part clipping
0xB8	pid_clip_d	8 (rw)	Differential part clipping
0xB9	pid_clip_int_sum	8 (rw)	Clipping for integration register
0xBA	pid_clip_int_input	8 (rw)	Clipping for input value of integrator
0xBC	pid_clip_sum	13 (rw)	Clipping for total sum of PID regulator
0xC0	pid_int_sum_reg	20 (rw)	Actual integration sum of the PID regulator
0xC4	pid_error_n	14 (r)	Status register Actual input value of the PID regulator (Error: Difference between *ENC_COUNT* and *RAMP_POS_ACT*)
0xC6	pid_error_pre	14 (r)	Status register Previous error for calculation of differential part
0xC8	pid_v_diff_out	14 (r)	Status register Velocity difference calculated by PID regulator
0xCA	pid_v_calc	14 (r)	Status register Velocity calculated from incremental encoder
0xD0	irq_polarity	6 (rw)	Configuration register of the interrupt controller Programmable polarity of the inputs (STOPL, STOPR, SLDL, SLDR, CHN) and interrupt output
0xD2	irq_enable	11 (rw)	Configuration register Interrupt enable
0xD4	irq_status	11 (rw)	Status register Status / Reset of the interrupt sources
0xE0	mstep_ram_adr	7 (w)	Microstep RAM address register
0xE1	mstep_ram_data	8 (rw)	Microstep RAM data register
0xE4	mstep_table_end	7 (rw)	Configuration register End of microstep table
0xE5	mstep_phase0_cnt	7 (rw)	Pointer 0 for microstep RAM
0xE6	mstep_phase1_cnt	7 (rw)	Pointer 1 for microstep RAM
0xE7	mstep_phase2_cnt	7 (rw)	Pointer 2 for microstep RAM
0xE8	mstep_full_step_dist	7 (rw)	Configuration register Fullstep distance in microstep RAM for clocking of the sequencer
0xE9	mstep_cnt_full_step_-dist	7 (rw)	Counter for fullstep distance
0xEA	mstep_conf	1 (rw)	Configuration of microstep RAM for one or two resp. three tables

13 Literature & Links

- 74LVC245 - Data Sheet, Texas Instruments,
  (online http://www.ti.com/)
- LTC1665 - Data Sheet, Linear Technology,
  (online http://www.linear.com/)
- TMC453 – Data Sheet Version 2.3, TRINAMIC Motion Control GmbH & Co. KG,
  (on-line www.trinamic.com -> Discontinued Products)
- TMC236 Datasheet, TRINAMIC Motion Control GmbH & Co. KG,
  (on-line www.trinamic.com)
- TMC239 Datasheet, TRINAMIC Motion Control GmbH & Co. KG,
  (on-line www.trinamic.com)
- TMC246 Datasheet, TRINAMIC Motion Control GmbH & Co. KG,
  (on-line www.trinamic.com)
- TMC249 Datasheet, TRINAMIC Motion Control GmbH & Co. KG,
  (on-line www.trinamic.com)
- Links to Distributors of TRINAMIC products,
  (on-line http://www.trinamic.com/ → Sales → Distributors)

14 Revision History

Version	Date (Initials)	Comment
0.99	July 24, 2007 (LL)	Initial version based on TMC453 Datasheet v. 2.3 / October 1st, 2004 (BD) section 4.2 Package Outlines and Dimensions added, section 4.5 TRINAMIC stepper motor driver interface for TMC236/TMC239 / TMC246/TMC249, 4.5.3 StallGuard^{TM} output signal of TMC246 / TMC249, section 4.5.4 PWM DAC output for current control, section 4.6 external DAC LTC1665 added, description of analog functions of the TMC453 removed - analog functions are no longer integrated within the TMC454; TMC453 Errata of TMC453 datasheet v. 2.3 (October 1, 2004) corrected within the TMC454 data sheet : Figure 7.1, page 30 : [23..0] ramp_pos_act, Figure 9.6, p. 41 : output mapping in sine step mode / phase output bit #15 (not #14), Figure 9.10, p. 47 - example shows the configuration for 4 micro steps now with 8 corrected table entries;
1.00	September 10, 2007 (LL)	First published version
1.01	January, 19, 2008 (LL)	hint added in table Table 4.1 : TMC454-BC pinout, that the clock must run before the NRES becomes inactive
1.02	October 5, 2009 (LL)	JTAG pins added to Table 4.1 : TMC454-BC pinout with hint how to disable the JTAG by pulling L10 (VJTAG) to GND (ground) instead of powering with +3V3; Note "In the 08 Version of the TMC454 writing to the microstep …" at the end of *section 9.3 Registers for Microstep Operation* changed to "In the TMC454 writing to the microstep …"

Please refer to www.trinamic.com for updated data sheets and application notes on this product and on other products.

The TMCtechLIB CD-ROM including data sheets, application notes, schematics of evaluation boards, software of evaluation boards, source code examples, parameter calculation spreadsheets, tools, and more is available from TRINAMIC Motion Control GmbH & Co. KG.