TMC332 – DATA SHEET

Micro step sequencer with integrated PWM units for 2-phase and 3-phase motors with closed loop current regulator

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

www.trinamic.com

1 Features

The TMC332 is a high resolution micro step sequencer with integrated sine wave look-up table and integrated PWM units. The TMC332 can control both, 2-phase stepper motors and 3-phase stepper motors. The TMC332 has two alternative step direction interfaces - a classical two wire step/direction interface and a SPI step direction interface for the TMC428 driver chain. An Internal micro step counter with programmable step width makes the handling of physical units comfortable. Parameterizing the TMC428 and the TMC332 takes place over a common SPI micro controller interface.

- micro step sequencer for smooth motion applications
- 2-phase stepper motor control
- 3-phase stepper motor control
- 9 bit PWM units
- 9 bit sine wave look-up table
- 8 bit PWM scaler
- 4 bit PWM sub-modulation for scaling quantization compensation
- programmable break-before-make timing
- classical two wire step direction interface
- driver chain step direction interface for TMC428
- common SPI interface for parameterizing of TMC332 and TMC428
- Integrated Current Measurement ADC (just two TLC374, one LM324 and some RC required)
- Integrated Current Regulation
- Manual or automatic micro-/full step switching for higher speeds
- 16 MHz clock frequency
- operation with 3.3V compatible IOs / 1.5V core voltage
- available in FBGA144 package

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.

2 Table of Contents

3 Table of Figures

4 Table of Tables

5 Pinning

A single package variant (BGA FG144, 0.5mm pitch) is available for the TMC332. For proper operation, it is strongly recommended to connect all power and ground pins (3.3V, 1.5V, GND).

Pin	FG144	I/O	Description
NRST	F1	I	low active reset input
CLK	G1	I	clock input
nEN	H2	I	low active enable input, nEN='1' disables all gate control signals equivalent to OVC
nSCS	M3	I	low active SPI chip select input of the TMC332 driven from µC
nSCS_428	L4	I	low active SPI chip select input driven from µC to select the TMC428
SCK	M4	I	serial data clock input driven from µC
SDI	M5	I	serial data input driven from µC
SDO_OF_428	L6	I	input driven by the SDO_C of the TMC428
SDO	M6	O / Z	serial data output to µC (from TMC332 or TMC428, depending on nSCS / nSCS_428)
STP	J1	I	step pulse input (logical ored with internal STEP pulse via TMC428 SPI driver chain)
DIR	K1	I	direction input (logical ored with internal STEP pulse via TMC428 SPI driver chain)
nSCS_DRV	M8	I	low active SPI chip select input driven from TMC428 (driver by nSCS_S)
SCK_DRV	L8	I	serial data clock input driven from TMC428 (driven by SCK_S)
SDI_DRV	M7	I	serial data input driven from TMC428 (driven by SDO_S)
SDO_DRV	L7	O	serial data output to TMC428 SDI_S or to next SPI driver within chain (SDI)
PH	A4	I	gate control high side, 0 : low active gate control / 1 : high active gate control
PL	B4	I	gate control low side, 0 : low active gate control / 1 : high active gate control
OVC_PH	K12	I	high side gate control outputs T1H, T2H, T3H, T4H on OVC condition
OVC_PL	K11	I	low side gate control outputs T1L, T2L, T3L, T4L on OVC condition
OVC_DISABLE	J11	I	high active OVC input (from comparator output that is high on OVC condition)
OVC_NDISABLE	J12	I	low active OVC input (from comparator output that is low on OVC condition)
OVC	H11	O	over current status (0 : no over current condition / 1 : over current condition)
SHAFT	A3	I	direction of motion clockwise / counter clockwise, direction depends on motor
nXY_UVW	B3	I	'0' : selects two phase stepper scheme / '1' : selects three phase stepper scheme
T1H	A5	O	T1H_Y2 / Y2 / deactivated (level depending on PL)
T1L	B5	O	T1L_Y2 / Y2 / deactivated (level depending on PH)
T2H	B6	O	T2H_Y1 / Y1 / T3H_W / W
T2L	C6	O	T2L_Y1 / Y1 / T3L_W / W
T3H	A7	O	T3H_X2 / X2 / T2H_V / V
T3L	B7	O	T3L_X2 / X2 / T2L_V / V
T4H	B8	O	T4H_X1 / X1 / T1H_U / U
T4L	A9	O	T4L_X1 / X1 / T1L_U / U
ENC_A	C1	I	3.3V input for incremental encoder signal A (a pull up-resistor might be required)
ENC_B	D1	I	3.3V input for incremental encoder signal B (a pull up-resistor might be required)
ENC_N	D2	I	3.3V input for incremental encoder signal N (a pull up-resistor might be required)
ENC_O_1	M2	O	Encoder compare output 1
ENC_O_2	L3	O	Encoder compare output 2
ENC_O_3	K4	O	Encoder compare output 3
RRC	A11	O	RC control for current measurement
COMP_ADC	B10	O	RC control for current measurement/limiting
COMP_PWM	A10	O	RC control for current limiting
CMP1P	C11	I	Positive comparator input for phase Y2
CMP1N	C12	I	Negative comparator input for phase Y2
CMP2P	D11	I	Positive comparator input for phase W/Y1
CMP2N	D12	I	Negative comparator input for phase W/Y1
CMP3P	E11	I	Positive comparator input for phase V/X2
CMP3N	E12	I	Negative comparator input for phase V/X2
CMP4P	F11	I	Positive comparator input for phase U/X1
CMP4N	F12	I	Negative comparator input for phase U/X1
STDBY	F3	I	Standby current enable (power down mode)
STANDALONE_EN	H6	I	Enable standalone mode (settings via SC_* inputs)
SC_FS_CUR_0	F9	I	Standalone mode configuration, Fullstep mode current, bit 0
SC_FS_CUR_1	G9	I	Standalone mode configuration, Fullstep mode current, bit 1
SC_FS_THRS_0	E8	I	Standalone mode configuration, Fullstep mode threshold, bit 0
SC_FS_THRS_1	F8	I	Standalone mode configuration, Fullstep mode threshold, bit 1
SC_FS_THRS_2	G8	I	Standalone mode configuration, Fullstep mode threshold, bit 2
SC_P_REG_CUR_0	J9	I	Standalone mode configuration, Current regulator target, bit 0
SC_P_REG_CUR_1	K9	I	Standalone mode configuration, Current regulator target, bit 1

SC_P_REG_P_0	H7	I	Standalone mode configuration, Current regulator P parameter, bit 0
SC_P_REG_P_1	H8	I	Standalone mode configuration, Current regulator P parameter, bit 1
SC_PD_AMOUNT_0	E2	I	Standalone mode configuration, power down mode, bit 0
SC_PD_AMOUNT_1	E3	I	Standalone mode configuration, power down mode, bit 1
SC_PHI_0	D9	I	Standalone mode configuration, phi setting (microstepping), bit 0
SC_PHI_1	D10	I	Standalone mode configuration, phi setting (microstepping), bit 1
SC_PWM_AMPL_0	J5	I	Standalone mode configuration, pwm amplitude, bit 0
SC_PWM_AMPL_1	J6	I	Standalone mode configuration, pwm amplitude, bit 1
SC_PWM_AMPL_2	K5	I	Standalone mode configuration, pwm amplitude, bit 2
SC_PWM_AMPL_3	K6	I	Standalone mode configuration, pwm amplitude, bit 3
SC_SD_SEL_0	G11	I	Standalone mode configuration, slow decay mode setting, bit 0
SC_SD_SEL_1	G12	I	Standalone mode configuration, slow decay mode setting, bit 1
1V5	A8, C4, E1, E10, G3, H1, H5, H12, J7		1.5V core supply voltage (all pins have to be connected proper operation)
3V3	A2, B12, E4, E6, E7, E9, H10, J3, L2, L5, L11, M10		3.3V IO supply voltage (all pin have to be connected for proper operation)
GND	A1, A6, A12, B2, B11, F2, F5, F6, F7, F10, G2, G5, G6, G7, K7, K10, L1, L10, L12, M1, M11, M12		ground (all pins have to be connected for proper operation)
reserved	B1, B9, C2, C3, C5, C7, C8, C9, C10, D3, D4, D5, D6, D7, D8, E5, F4, G4, G10, H3, H4, H9, J2, J4, J8, J10, K2, K3, K8, L9, M9		not connected, may be used in the future do not connect

Table 5-1: TMC332-FG144 Pinning

5.1 Package Outlines and Dimensions

5.1.1 TMC332-BC - Fine Pitch BGA Package with 144 Balls (FBGA144)

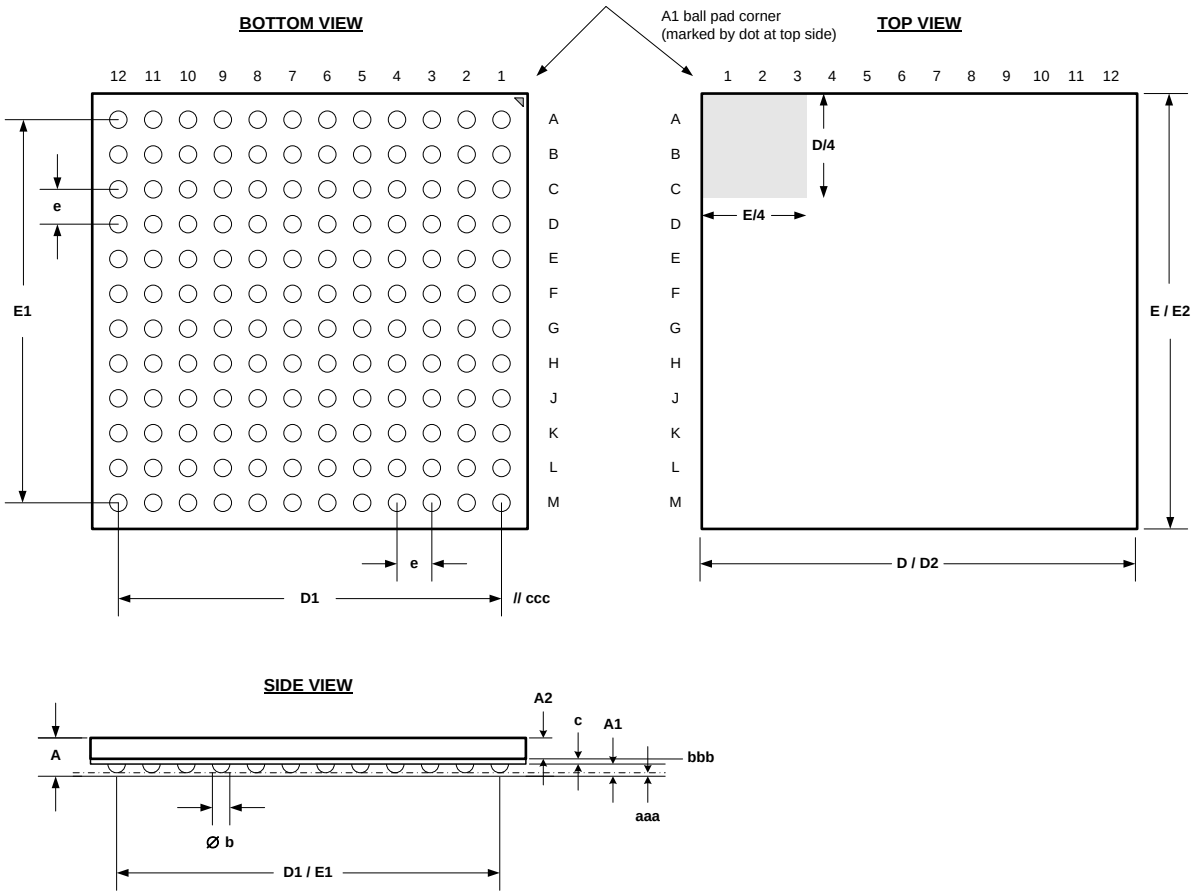

Figure 1: Package Outline Drawing FBGA144 – (JEDEC MO-192 VAR DAD-1)

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 5-2: Dimensions of FBGA144 (Note: BSC = Basis Spacing Between Centers)

6 General Description

The functional block structure of the TMC332 is given by Figure 2. The TMC332 is equipped with two step direction interfaces processed by the micro step sequencer. The micro step width can be programmed in a wide range with a high resolution. Together with the additional 32 bit position counter, one can easily realize stepping within physical units. The TMC428 step direction interface of the TMC332 is directly compatible to the driver chain interface of the TMC428. The other step direction interface is a classical two wire step direction interface. The parameterizing of the TMC332 takes place via a separate four wire serial interface (SPI) for microcontroller access to the internal register (Control Logic / Register Bank). The integrated sine wave look up table (SIN LUT) can output either two sine waves with a phase shift of 90° for two phase stepper motors or three sine waves with +/- 120° phase shift for three phase stepper motors. The gate control block provides the gate control signals for the power stage half bridges, including programmable break-before-make timing and over current protection. PWM frequency and brake-before-make timing are programmable.
Incremental encoders with ABN output can be connected to the TMC332 to evaluate the physical motor position. Three output lines are available to trigger external actions when previously set encoder positions are reached.
The TMC332 also contains a current measurement unit and a current regulator, requiring only a few inexpensive external components.

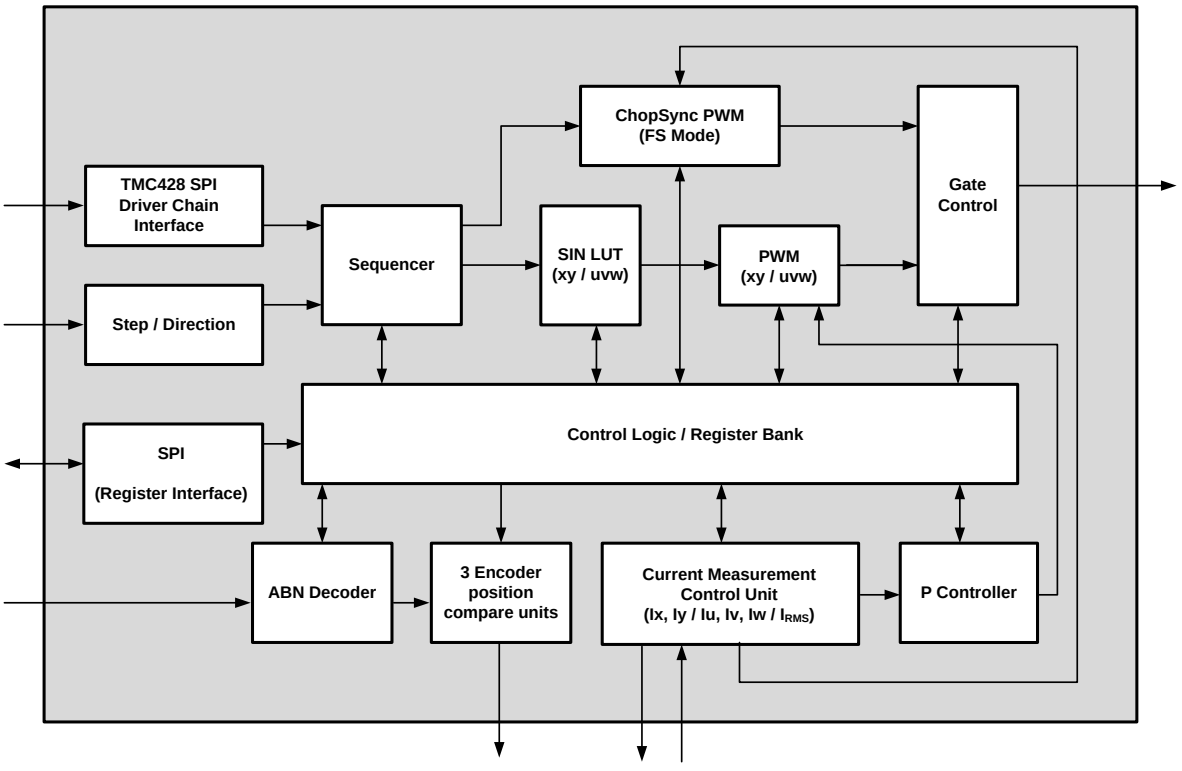

Figure 2: TMC332 Functional Block Diagram

6.1 Chipset Configuration Outlines

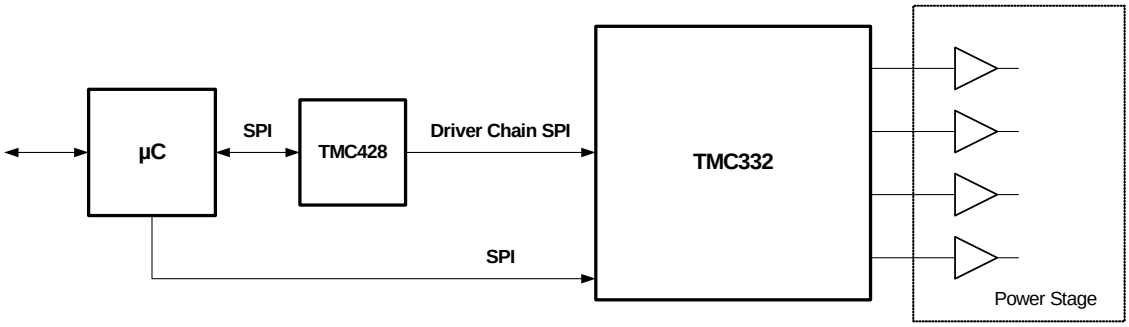

Figure 3: Outline of a typical TMC332 setup with TMC428 and µC

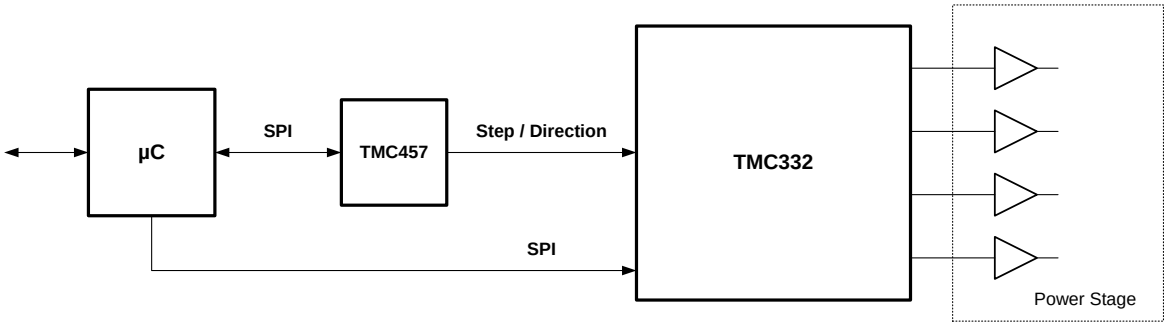

Figure 4: Outline of a TMC332 Setup with TMC457 and µC

7 Application circuit notes

7.1 SPI chain details

When using the TMC428 to drive the TMC332, the SDO line of the SPI bus, leading to the microcontroller is shared between the TMC332 and the TMC428. Usually the TMC428 requires external logic if the SPI bus has to be shared with other ICs. To avoid another IC, the TMC332 includes logic which switches the SDO signal of the TMC428 through as long as the nSCS428 input is low (active). In all other conditions, the line uses the SDO signal of the TMC332. This means that both chips are connected parallel with distinct chip select lines. That makes it possible to send small datagrams either to the TMC428 or to the TMC332 instead of having to send a large datagram through a daisy chain of both chips.

Since it makes no sense to write the same data to both chips at once, only one chip select line should be active. If both chip select lines are active and data is transferred, both chips will evaluate the data but only the output of the TMC428 will be returned.

If the step/direction input is used, the nSCS_428 pin should be connected to 3.3V and the SDO_OF_428 pin should be connected to GND. When using a TMC428, the STP and DIR pins should be connected to GND.

The configuration of the TMC428 must be chosen so that the last transmitted bits represent the direction and step information. For the fastest possible communication, only these two bits should be transmitted and the configuration should be $12, $33. This represents the direction bit selection ($12) and the step bit selection with the “next motor” bit set ($33).

7.2 Step/Direction interface

The STP and DIR inputs represent a direct Step/Direction interface that can be used instead of the TMC428 interface. The minimum step pulse with must be larger than one clock cycle of the TMC332 to ensure the recognition of every step pulse.

7.3 Gate control output polarity (PH, PL pins)

The PH and PL pins define the polarity of the high- respectively low-side-gates of the bridges. When using non inverting gate drivers and N-MOSFETS on the high side and the low side, both pins would be tied to 3.3V.

If one of the pins sets the wrong polarity, both transistors in the bridge would be switched at once and most probably trigger the over current protection. If both polarities are inverted, the short circuit state would only last as long as the break-before-make delay and if this time is too short to trigger the over current protection, this would lead to a high power consumption in the bridge transistors.

7.4 Encoder input pins

If no encoder is used in the application, the encoder input pins ENC_A, ENC_B and ENC_N pins should be connected to either GND or 3.3V.

7.5 Current regulation pins

If no current measurement/regulation and no full step mode is needed in the application circuit (when the motor is only operated at low speeds), the external current sensing parts can be omitted. In this case, the comparator inputs (CMPnP and CMPnN, with n = 1, 2, 3, 4) should be connected to GND.

If full step mode is needed, all inputs should be connected together and should be high, as long as the motor current is too high (The bridges are turned on while no input is high, where CMP1P/N are not evaluated in three phase stepper mode and the phases are evaluated separately in two phase stepper mode).

7.6 Over current protection

An external signal, either created by another integrated circuit or by some discrete components can be used to detect over current conditions in the motor power path. The TMC332 provides two complementary inputs, which disable the bridges immediately. This over current condition is not reset automatically but needs a SPI datagram to be sent to the TMC332. The datagram contains the set *O* (clear over current condition) bit: **$88 00 00 01**

Usually only one of the two inputs is used and the other is tied to GND or 3.3V, to disable its function:

Over current signal active level	OVC_DISABLE	OVC_NDISABLE
low	GND	signal
high	signal	3.3V

The state of the high and low side outputs during an over current condition is defined by the OVC_PH and OVC_PL pins, which work equivalent to the PH and PL pins.
The pin OVC is a representation of the over current flag in the register bank and can be used to trigger an interrupt in a microcontroller.

7.7 Standalone pins

When using the standalone mode (external configuration), the **STANDALONE_EN** pin has to be connected to 3.3V and the SC_* inputs have to be connected to 3.3V or GND to set the needed values. The exact function of these pins is explained below.
When using the classic configuration mode via the SPI bus, the **STANDALONE_EN** pin has to be connected to GND.
In standalone mode, the SC_* inputs override some settings in the register bank.

7.7.1 Configuration input description

The numbers of the inputs with the same name are the bit numbers, where bit 0 is the LSB. The binary value 0 represents 0V/GND and 1 represents 3.3V.

SC_PWM_AMPL_*
The 4 bit value is used as the upper 4 bits of the 8 bit pwm_ampl value. The lower 4 bits are set to 1, resulting in 16 possible values from 15 to 255 in steps of 16.

SC_PHI_*
The four possible settings for PHI are:

SC_PHI_		phi	microstepping with 2 phase stepper
1	0		
0	0	1	256x
0	1	4	64x
1	0	16	16x
1	1	64	4x

SC_PD_AMOUNT_*
The power down amount bits are the same as the stdby value in the register bank but the reduced power setting is used as long as one of the inputs is set high.

SC_PD_AMOUNT_		pwm_ampl / current regulator target
1	0	
0	0	* 1
0	1	* 0.5
1	0	* 0.25
1	1	* 0.125

SC_P_REG_CUR_* and **SC_P_REG_P_***
The current regulator is disabled as long as **SC_P_REG_CUR_0** and **SC_P_REG_CUR_1** are both 0. In other cases the current regulator is enabled and the P parameter selected with SC_P_REG_P_* is used. The regulator tolerance is set to 12.5%.

SC_P_REG_CUR_		current regulator target
1	0	
0	0	disabled
0	1	40
1	0	60
1	1	80

The P parameter is configured via the SC_P_REG_P inputs

SC_P_REG_P_		P parameter	equivalent setting in register bank
1	0		

0	0	1/512	$0008
0	1	1/256	$0010
1	0	1/128	$0020
1	1	1/64	$0040

SC_FS_THRS_* and **SC_FS_CUR _***
The manual/automatic full step switching is configured with the SC_FS_THRS_* inputs.

SC_FS_THRS_			Full step switching (threshold equivalent to register bank)
2	1	0	
0	0	0	always off
0	0	1	$10
0	1	0	$20
0	1	1	$30
1	0	0	$40
1	0	1	$80
1	1	0	$C0
1	1	1	always on

Note that in “always on” mode the phi setting from the **SC_PHI_*** inputs is still active, meaning that more than one step pulse (4/16/64/256) is needed to make the motor do one full step.

The SC_FS_CUR_* inputs configure the current setting in full step mode.

SC_FS_CUR_		Setting equivalent to register bank
1	0	
0	0	2
0	1	4
1	0	6
1	1	8

SC_SD_SEL_*
The slow decay configuration inputs have the same functionality as the sl_d bits in the register bank.

SC_SD_SEL_		Setting equivalent to register bank
1	0	
0	0	no slow decay phase
0	1	50% slow decay phase
1	0	75% slow decay phase
1	1	87.5% slow decay phase

7.8 Schematics

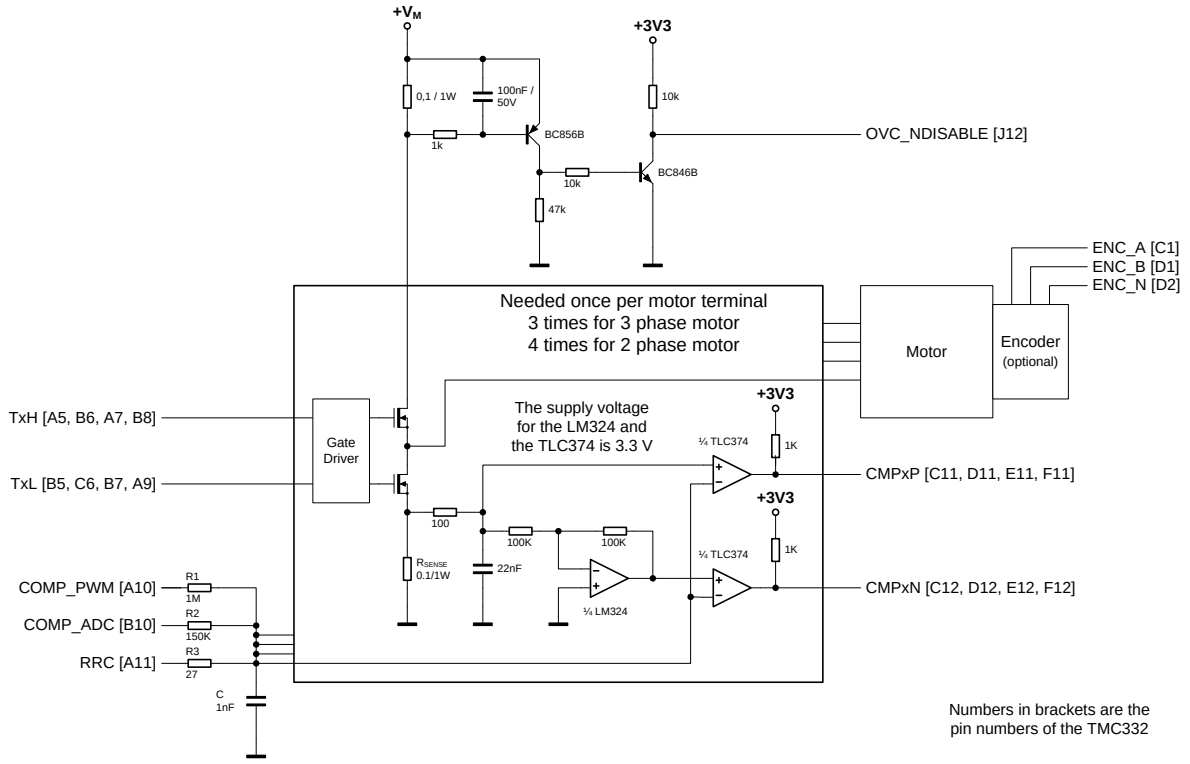

Figure 5: Schematic with over current detection circuit

Figure 5 shows the external components of the TMC332 needed to drive a motor with current measurement/control and over current protection. Not shown are the SPI bus to access the register bank and the control hardware (TMC428 SPI or step/direction interface). The over current detection circuit in the V_M line switches at approximately 5A.

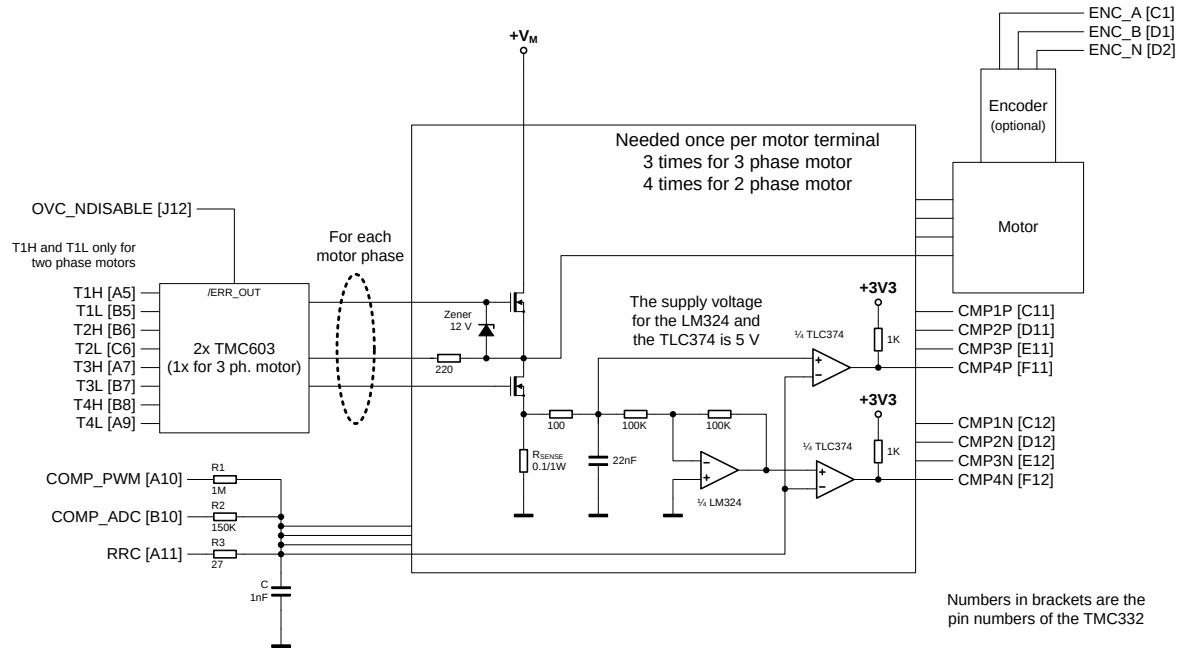

Figure 6: Example layout with TMC603

When using the TMC603 as gate driver, the parts for the over current protection can be omitted as the signal is provided by the TMC603. One TMC603 is able to drive 3 half bridges, which is enough for a three phase motor. For a two phase motor, two TMC603 are required, where every TMC603 drives the half bridges of one motor phase. In that case the error outputs of the two TMC603 are both connected to the OVC_NDISABLE pin.

7.9 Selecting the external components

For different current requirements, the values of the external components can be adapted. The best way is to change the value of the sense resistor.
For good regulation results, the target current value should be around 80 but not higher than 90. With a two phase motor, this equals a charging time of about 10µs for the capacitor C. With the values in the example layouts (R2 = 150k, C = 1nF), this results in a voltage of 0.22V. With an 1A (RMS) motor, a 0.15 Ohm sense resistor would create a 0.21V sense voltage at 1.41A peak phase current.
For three phase motors the charging time should be about 8.125µs, so the voltage is at 0.18V. With a 6A three phase motor, a 0.033 Ohm sense resistor would create 0.198V sense voltage.

The operational amplifiers and the comparators can be different parts than LM324 and TLC374, but care should be taken with the comparators as they should have a low input bias current.

8 Register Description

The TMC332 is equipped with a set of registers for parameterizing. An overview of the registers is given in Table 8-2: TMC332 register address mapping and the detailed explanation follows in Table 8-3: TMC332 functional register description.
$b1 describes the hexadecimal value of the first byte to send to the TMC332. It contains the *w* bit which describes if data should be read (0) or written (1) followed by the seven address bits *%add*. If the *w* bit is 1, the remaining 24 bits of the datagram contain the data that should be written to that address according to the table, if the w bit is 0 for reading data from the TMC332, the value of the remaining bits is not important (don't care).

32 bit DATAGRAM to send from a µC to the TMC332 via pin SDI **read access**																																
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$b1	***w***	***%add***							***data***																							
$00	0	0	0	0	0	0	0	0	x_l																							
$01	0	0	0	0	0	0	0	1																	x_h							
$03	0	0	0	0	0	0	1	1	phi_frac																							
$04	0	0	0	0	0	1	0	0															phi_int									
$06	0	0	0	0	0	1	1	0																	pwm_ampl							
$07	0	0	0	0	0	1	1	1											pwm_offs				pwm_f						bbm_dly			
$08	0	0	0	0	1	0	0	0	status (h=inhibit & o=ovc)																						h	o
$09	0	0	0	0	1	0	0	1	enc_count																							
$0A	0	0	0	0	1	0	1	0															n									
$0B	0	0	0	0	1	0	1	1	i_u/i_x1 (adc4)								i_v/i_x2 (adc3)								i_w/i_y1 (adc2)							
$0C	0	0	0	0	1	1	0	0	i_y2(adc1)										Limit	Err_u	Err_v	Err_w	Err_y2		i_abs: sqrt(i_u²+i_v²+i_w²) / sqrt(i_x²+i_y²)							
$0E	0	0	0	0	1	1	1	0																								f
$0F	0	0	0	0	1	1	1	1	type (332)											version (200)												
$10	0	0	0	1	0	0	0	0	enc_comp_1																							
$11	0	0	0	1	0	0	0	1	enc_comp_2																							
$12	0	0	0	1	0	0	1	0	enc_comp_3																							
$b1	***w***	***%add***							***data***																							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 8-1: TMC332 register address mapping (read access)

32 bit DATAGRAM to send from a µC to the TMC332 via pin SDI

write access

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$b1	***w***	***%add***							***data***																							
$80	1	0	0	0	0	0	0	0	x_l																							
$81	1	0	0	0	0	0	0	1																	x_h							
$82	1	0	0	0	0	0	1	0									dx															
$83	1	0	0	0	0	0	1	1	phi_frac																							
$84	1	0	0	0	0	1	0	0															phi_int									
$85	1	0	0	0	0	1	0	1	not in use																							
$86	1	0	0	0	0	1	1	0			sl_d		stdby			m	stdby_timeout								pwm_ampl							
$87	1	0	0	0	0	1	1	1											pwm_offs			pwm_f							bbm_dly			
$88	1	0	0	0	1	0	0	0	status & control (p=phi_wr & h=inhibit & o=ovc)																				p	h	o
$89	1	0	0	0	1	0	0	1											b	enc_acc_const_int									enc_acc_const_frac			
$8A	1	0	0	0	1	0	1	0														c	t	enc_clr_mode								
$8B	1	0	0	0	1	0	1	1	not in use																							
$8C	1	0	0	0	1	1	0	0																r		targ_curr						
$8D	0	0	0	0	1	1	1	0						tolerance						p_parameter												
$8E	1	0	0	0	1	1	1	0						d	e	s				FS_targ_curr				FS_thr								
$8F	1	0	0	0	1	1	1	1	not in use																							
$90	1	0	0	1	0	0	0	0	enc_comp_1																							
$91	1	0	0	1	0	0	0	1	enc_comp_2																							
$92	1	0	0	1	0	0	1	0	enc_comp_3																							
$b1	***w***	***%add***							***data***																							
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 8-2: TMC332 register address mapping (write access)

The x_l and x_h registers together form a 32 bit wide position counter.
The register dx holds the step width for the position counter.
For each micro step, a 34 bit wide register accumulates the micro step position by an amount of phi where phi is composed by an integer part phi_int and a fractional part phi_frac. With this, one can realize non-integer micro stepping. The most significant 10 bits of the micro step position register form the micro step pointer within the sine wave look-up table.
The register pwm_ampl controls the scaling of the PWM. An additional control bit named 'm' is for selection of a PWM modulation that gives a higher resolution at low values of pwm_ampl.
The pwm_f register controls the PWM frequency and the bbm_dly controls the brake-before-make timing.
The status and control register bits named 'p', 'h' are for enabling of the micro step counters to keep them consistent under every condition. The status bit 'o' indicates an over current condition.
The version of the TMC332 can be read out via a dedicated register.

addr	r/w	register	bit(s)	functional description
$00	r	Position low	x_l	read lower 24 bits of 32 bit physical position counter X, 1st read x_l followed by read x_h to get a consistent 32 bit position
	w			set lower 24 bits of 32 bit physical position counter X
$01	r	Position high	x_h	read higher 8 bits of 32 bit physical position counter X, 1st read x_l before you read x_h to get a consistent 32 bit position
	w			set higher 8 bits of 32 bit physical position counter X
$02	w	Position step width	Dx	write the dx register, the value dx is added or substracted with each step according to the direction input
$03	r/w	micro step width, fractional part	phi_frac	24 bit wide fractional part of the of the micro step width
$04	r/w	Micro step width, integer part	phi_int	10 bit wide integer part of the micro step width
$06	r	pwm amplitude	pwm_ampl	read PWM amplitude
	w		m	enable PWM sub-modulation
			sl_d	slow decay between PWM polarity changes sl_d = %00 : no slow decay phase sl_d = %01 : t*1/2 slow decay phase sl_d = %10 : t*3/4 slow decay phase sl_d = %11 : t*7/8 slow decay phase
			stdby	pwm amplitude/current during power down mode stdby = %00 : 100% of normal value stdby = %01 : 50% of normal value stdby = %10 : 25% of normal value stdby = %11 : 12.5% of normal value
			stdby_timeout	Delay between last full step and power down activation: delay [s] = (stdby_timeout * 1048576) / f_clk[Hz]
			pwm_ampl	set PWM amplitude pwm_ampl is used as the upper limit if the current regulator is active
$07	r/w	pwm control	pwm_offs	PWM offset; adds an offset to the PWM values to create steeper zero crossings of the current wave
			pwm_f	PWM frequency [Hz] = f_clk[Hz] / ((pwm_f+1)*2^{10})
			bbm_dly	break before make delay time [s] = 2^(bbm_dly+1) / f_clk[Hz]
$08	r/w	status & control	p	phi register write bit (change of phi) p = 0 : leave phi untouched p = 1 : update phi := phi_int & phi_frac (write only bit, read always '0')
			h	inhibit step direction interfaces (no count x_h, x_l, phi_int, phi_frac) h = 1 : inhibit STP / DIR (and TMC428_DRV interface) h = 0 : enable STP / DIR (and TMC428_DRV interface)
			o	over current status bit o = 1 : over current condition (read) / reset OVC (write) o = 0 : no over current condition
$09	r	encoder counter	enc_count	24 bit wide encoder counter
	w	encoder step width	b	Interpret fractional part normally (enc_acc_const_frac/16) when 0 or decimal (enc_acc_const_frac/10) when 1
			enc_acc_ const_int	9 bit wide integer part of the encoder constant
			enc_acc_ const_frac	4 bit wide fractional part of the encoder constant
$0A	r	encoder status	n	Encoder is cleared at the next clear condition
	w	encoder control	c	Manual reset of enc_count
			t	Trigger reset of enc_count via bit 5 of cm

			enc_clr_mode	Clear mode (see 9.4.3 Encoder Clear Mode)
$0B	r	current register 1	i_u	measured current of terminal U or X1
			i_v	measured current of terminal V or X2
			i_w	measured current of terminal W or Y1
$0C	r	current register 2	i_y2	Measured current of terminal Y2
			Limit	PWM Amplitude limited by regulator
			Err_u	Error flag of bridge 4 (U/X1)
			Err_v	Error flag of bridge 3 (V/X2)
			Err_w	Error flag of bridge 2 (W/Y2)
			Err_y2	Error flag of bridge 1
			i_abs	Calculated absolute current For two phase stepper motors with nXY_UVW=0 the TMC332 uses i_a = i_x1 or i_a = i_x2 and i_b = i_y1 or i_b = i_y2 to calculate the absolute current i_abs = sqrt($i\_a^2 + i\_b^2$). For three phase stepper motors with nXY_UVW=1 the TMC332 uses i_u, i_v and i_w to calculate the absolute current i_abs = sqrt($i\_u^2 + i\_v^2 + i\_w^2$).
	w	current reg. control	r	Activate current regulator
			targ_curr	Target current of the regulator (compared with i_abs)
$0D	w	current reg. param	tolerance	Tolerance setting of the current regulator: %000 – no tolerance %001 – 3.125% of target current %010 – 6.25% of target current %011 – 12.5% of target current %100 – 25% of target current all other values – no tolerance
			p_param	P-parameter of the regulator (12bit) The difference between the target current and the calculated absolute current is multiplied by this value, divided by 4096 and added to the previous PWM amplitude.
$0E	r	full step state	f	full stepping is enabled when this bit is set
	w	full step register	d	Chopper Mode, 0 opens high side, 1 pulls high side down
			e	enable full stepping permanently (overrides s)
			s	switch between micro and full stepping automatically
			FS_targ_curr	Current limitation for FS mode
			FS_thr	Automatic switching point from micro to full stepping
$0F	r	type & version		type and version read only register of the TMC332, type = $332 identifies the TMC332 and version = $200 identifies version 2.00
$10 $11 $12	r/w	encoder compare registers	enc_comp_1 enc_comp_2 enc_comp_3	24 bit wide encoder compare value, if the encoder counter (enc_count) equals one of these registers, the associated output (ENC_O_1, ENC_O_2, ENC_O_3) is driven high

Table 8-3: TMC332 functional register description

8.1 SPI communication examples

Some examples show the communication to access the register bank.

Read out the type and version of the chip:

Send	Receive	Comment
$0F 00 00 00	$00 33 22 00	The received type and version of the chip are: TMC332 V2.00

Set the micro step width to 4.25:

Send	Receive	Comment
$84 00 00 04	$00 00 00 00	set phi_int to 4
$83 40 00 00	$00 00 00 00	set phi_frac to $40000000 (= 0.25 * 16777216)

Clear the position counter:

Send	Receive	Comment
$80 00 00 00	$00 00 00 00	clear lower 24 bits
$81 00 00 00	$00 00 00 00	clear higher 8 bits

8.2 Full initialization example

This example shows how to initialize the TMC332 for use with a three phase stepper motor with 600 full steps per revolution which should have 36000 micro steps per revolution ($^1/_{100}$ degrees per micro step, 60 micro steps per full step), active current regulation and an encoder with 2000 pulses per revolution that should count the micro steps. The position counter should count 1/10 micro steps.
The detailed explanation of the settings and their calculation follows in chapter 9 (Parameterizing the TMC332).

Datagram to send	Description	
$80 00 00 00	Reset position counter	
$81 00 00 00		
$82 00 00 0A	set Dx to 10	
$83 D8 2D 83	set phi_frac to 14167427	phi := 2.8444444
$84 00 00 02	set phi_int to 2	
$8D 02 00 1E	parameterize the current regulator	
$8C 00 01 50	enable the current regulator with target value 80	
$86 01 00 FF	set the PWM limit to 255	
$89 00 01 20	Encoder constant := 18	
$8A 00 05 A8	Reset encoder counter now and on every falling edge of the N signal	

9 Parameterizing the TMC332

9.1 Position counter (Address $00 to $02)

With each step pulse, the 32 bit wide position counter X accumulates according to X := X + dx. The dx has to be set corresponding to a choice of the micro step width (see 9.2) to count positions within physical units.

9.1.1 Additional notes

Since the position counter has an integer value, the unit of measurement has to be changed to avoid rounding errors. One way is to change the units prefix, e.g. if one step pulse is equal to a linear motion of 0.25 µm, one could use nm instead of µm and set dx to 250. This will of course reduce the distance between two counter overflows from about 4 km to 4 m.

9.1.2 Example

For example, if one micro step is equal to 1 µm for a given value of phi_int an phi_frac and the dx is set to 1, the X resp. x_h and x_l represent the position within unit µm. Changing the phi_int and phi_frac in a way that one micro step is equal to 10 µm would require to set dx to 10. With this, the position would still be within the unit µm.

9.2 Micro step width (Address $03 and $04)

With the configurable micro step width, the rotational angle of one micro step can be set to a value that is optimized for a special application.
With each step pulse, the 34 bit wide micro step angle PHI is accumulated according to PHI := PHI + (phi_int + phi_frac / 2^{24}). The upper 10 bits of PHI represent the micro step position within an electrical period. One electrical period contains 1024 values. For two phase stepper motors, one electrical period is equal to four full steps. For three phase stepper motors, one electrical period is equal to six full steps. The fractional part phi_frac allows micro stepping with non-integer micro step widths.

To calculate phi_sum the following formula can be used:

$$phi\_sum = (FS_R/FS_P) * (1024/pos_R)$$

FS_R is the number of full steps in one revolution of the motor
FS_P is the number of full steps in one electrical period (4 or 6, see above)
pos_R is the needed number of positions in one revolution of the motor

To get phi_int and phi_frac, phi_sum has to be split up: phi_int is directly the integer part of phi_sum, phi_frac is the fractional part of phi_sum multiplied with 2^{24}.

9.2.1 Additional notes

phi_sum should never be set to a value larger than 256 to avoid aliasing effects. The only exception to this is when updating PHI itself (using the p bit in register $88), which has to be done when the motor is stopped (although it will probably move by the fraction of one fullstep when changing PHI). To update PHI itself, one as to set phi_int and phi_frac to the values, that should be written to PHI (**$83 nn nn nn** / **$84 xx xn nn**), then send the datagram that updates PHI (**$88 00 00 04**) and finally set phi_int and phi_frac to the values representing the micro step width (**$83 nn nn nn** / **$84 xx xn nn**).
Due to the 1024 discrete values in the micro step table per electrical period, only the integer part of PHI will be used when accessing the micro step table. The fractional part is only relevant when using micro step widths with a fractional part, e.g. when setting both PHI and phi_sum to 0.25, the values of PHI during motion will be 0.00, 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00, 2.25, 2.50, 2.75,... For accessing the micro step table, only the integer parts will be used: 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2,…

9.2.2 Examples

A two phase stepper motor with 200 full steps per revolution should reach 360 positions (1 degree apart):
phi_sum = (200/4) * (1024/360) = 50 * 2.8444… = 142.2222…
phi_int = 142 = $08E

phi_frac = 2^{24} * 0.2222… = 3728270 = $38E38E
The datagrams sent to the TMC332 would be **$83 38 E3 8E** and **$84 00 00 8E**

The same resolution for a 600 step three phase motor:
phi_sum = (600/6) * (1024/360) = 100 * 2.8444… = 284.4444…
phi_int = 284 = $11C
phi_frac = 2^{24} * 0.4444… = 7456540 = $71C71C
The datagrams sent to the TMC332 in this case would be **$83 71 C7 1C** and **$84 00 01 1C**

Another example could be a 600 step three phase motor driving a lead screw that moves 5mm per revolution. The desired positioning accuracy is 1µm, which means 5000 positions per revolution are needed.
phi_sum = (600/6) * (1024/5000) = 100 * 0.2048 = 20.48
phi_int = 20 = $014
phi_frac = 2^{24} * 0.48 = 8053063 = $7AE147
The datagrams sent to the TMC332 in this case would be **$83 7A E1 47** and **$84 00 00 14**

9.3 PWM settings (Address $06 and $07)

The *pwm_ampl* controls the level and direction of the current through the motor by modifying the maximum duty cycle of the PWM. The sub modulation bit *m* enables a finer resolution of the PWM by periodical changes of the PWM amplitude.
Without current regulation, the PWM amplitude setting for a given current on a stopped two phase motor can be approximated. The required values are the supply voltage (V_{supply}) the resistance of one motor coil (R_{coil}) and the RMS current value (I_{RMS}) or the peak current value (I_{peak}):

$$PWM\_AMPL = I_{RMS} * 1.41 * R_{coil} * 256 / V_{supply} = I_{peak} * R_{coil} * 256 / V_{supply}$$

The pwm_offset parameter is used for fine tuning of the PWM sine curve to avoid short periods of zero current created by the BBM time.
The *pwm_f* parameter can be used to reduce the PWM frequency. The original frequency of f_{CLK}/1024 - usually 15625 Hz - is divided by *pwm_f*+1

$$f_{PWM} = f_{CLK} / (1024 * (pwm\_f + 1))$$

With the *bbm_dly* parameter, the break before make delay can be changed in the range of $2/f_{CLK}$ (usually 1/8 µs) to $2^{16}/f_{CLK}$ (usually 4,096 ms). The exact delay time can be calculated with the following formula:

$$t_{DLY} = (bbm\_dly + 1) / f_{CLK}$$

The optimal setting depends on the gate charge of the transistors and the gate driving current.
An approximation for the value with given gate drivers and transistors can be made:

$$bbm\_dly = (Q_g / I_g) * f_{CLK} - 1$$

Usually a slightly lower value can be chosen. The best setting can be found when no motor is connected, *bbm_dly* is set to 15 (the highest possible value) and the power consumption of the circuit is monitored. Then the *bbm_dly* is lowered until the power consumption rises. The lowest value before the Power consumption rises is the optimal setting for bbm_dly.

The sl_d parameter is used for a configurable time of slow decay on each polarity change. The percentage of normal current flow during each PWM cycle can be described as $t_{ON}/T = 2^{-sl\_d}$.

9.3.1 Additional notes

Generally the PWM frequency does not need to be changed; in most cases a lower frequency leads to audible vibrations in the motor.

9.3.2 Example

Assuming the TMC332 is clocked with 16 MHz, a PWM with an amplitude of 42 ($2A) and no modulation, about 5 kHz frequency and ~1 ms break before make delay should be set up.

The datagram to set up the amplitude would be **$86 00 00 2A** and the frequency and break before make delay is set up with **$87 00 02 0D**.

9.4 ABN encoder interface (Address $09, $0A and $10 to $12)

With the ABN encoder interface it is possible connect an ABN encoder (quadrature encoder) to the TMC332 and read out the position with configurable step width via the SPI interface. If the encoder has open collector outputs, external pull up resistors are needed.

9.4.1 Additional notes

The N-Signal which usually indicates one position on the revolution can be used to reset the counter (see 9.4.3 Encoder Clear Mode). If no reset functionality is needed, the N input should either be connected to GND or 3.3V. If the encoder does not provide such a signal, a microcontroller can generate the N signal from any needed condition. Even if the N input is unused, enc_count can be reset via SPI. (**$8A xx 04 00** – *t* and *enc_clr_mode* are unused in that case)

9.4.2 Encoder constant

The encoder constant (9 bit wide integer part *enc_acc_cons_int*, 4 bit wide fractional part *enc_acc_cons_frac*) is added to an internal register with each encoder step. The integer part from this accumulation register can be read out as *enc_count*. Additionally the fractional part is interpreted as decimal when Bit 13 (b) is set to 1.
So the constant can be adjusted between 0 and 511 15/16 in binary mode (Bit 13 = 0) and between 0 and 511 9/10 in decimal mode.

9.4.3 Encoder Clear Mode

Bit	8	7	6	5	4	3	2	1	0
Name	*cln*	*ntn*	*ntp*	*neo*	*nec*	*ab_ig*	*n_pol*	*b_pol*	*a_pol*

Table 9-1: Description of the enc_clr_mode bits

One way to reset the *enc_count* register is to send a datagram with the manual reset bit (bit 10 at address $8A) set. The datagram to do that is **$8A xx 04 00**.
The advanced clear modes can be configured via the clear mode bits (*bits 8 … 0* at address $8A). The bit that always has to be set, to reset the counter using the N input is *cln*, which enables the reset function.
ntn and *ntp* are used to determine, on which edge of the N signal the counter should be reset. If both bits are cleared (0), the active logic level (which is determined by *n_pol*) of N triggers the reset, any edges of the A and B signals are ignored while N is active. If *ntp* is set to 1, the start of the N condition triggers the reset. That means if N is low active, the falling edge of N is the start of the N condition. If N is high active, the rising edge of N is the start of the N condition. If *ntn* is set to 1, the end of the N condition triggers the reset. It is also possible to set both bits and reset the counter on both edges.
neo makes it possible to reset the counter only on the next clear event. When this bit is set, writing a 1 to bit 9 (*t*) in the register, clears the counter on the next clear event. The combination of *neo* and level sensitive evaluation of N (*ntn* = *ntp* = 0) acts the same way as *neo* combined with edge sensitive evaluation of N responding to the first edge (falling if low active, rising if high active).
If *nec* is set, the counter is cleared on every clear event, which would usually be once per revolution.
To determine if the encoder counter will be cleared at the next clear event, the flag *n* in the encoder status register ($0A) can be read. If this bit is set, the next clear event will clear the encoder counter. If the *nec* bit is set, the *n* bit will always read 1.
ab_ig disables the evaluation of the levels of A and B to determine if a clear event is triggered. When this bit is set, only the N signal is evaluated.
n_pol describes the value of the N input that is interpreted as active. If N is active low, *n_pol* would be set to 0.
The two lowest bits (*b_pol* and *a_pol*) define the logic level, the inputs B and A must have when the edge (or state) of N triggers a clear event.

The following diagrams illustrate the edge/level sensitivity controlled by the bits *ntn* and *ntp* and the difference between the *neo* and *nec* bits.

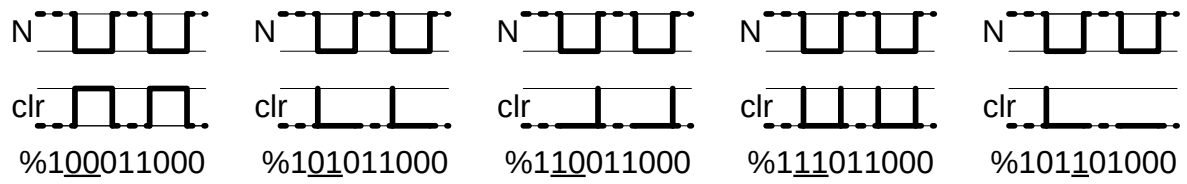

Figure 7: Effects of the Encoder clear mode bits

The first four diagrams above show the N signal and the internal clear signal (clr) which resets *enc_count* to 0 when it is high for different settings of *enc_clr_mode* (shown below each diagram). The reset occurs on every clear state of N (*nec* is set) and regardless of the levels of A and B (*ab_ig* is set).
The fifth diagram shows the use of *neo*, which resets *enc_count* only at the first clear condition after the *t* bit is set. The *enc_clr_mode* is set at the start of the diagram together with the *t* bit and the first falling edge of N resets *enc_count*. All of the following falling edges of N are ignored until the *t* bit is set again.

9.4.4 Encoder compare registers

The three encoder compare registers $10, $11 and $12 can be set to values that are compared to the encoder counter (*enc_count* register). As long as the encoder counter equals one of the registers, the corresponding output (ENC_O_1 for *enc_comp_1*, ENC_O_2 for *enc_comp_2*, ENC_O_3 for *enc_comp_3*) is set.

9.4.5 Examples

Encoder constant

An encoder with 8000 pulses per revolution is used on a 200 steps/revolution motor and the microstepping is set to 1/4 (PHI = 64). The microsteps should be counted by the encoder. 10 encoder counts equal one microstep so the encoder constant is 0.1. This can be reached with the decimal mode by setting the b Bit (Bit 13) to 1, setting *enc_acc_cons_int* to 0 and *enc_acc_cons_frac* to 1. The complete datagram is $89 00 20 01.

Clear mode

The clear mode **%110011000** clears (*cln*) the counter on each (*nec*) rising edge (*ntn*, end of N condition) of the low active (*n_pol*) signal N, regardless of the logic level of A and B (*ab_ig*).
The clear mode **%101100111** clears (*cln*) the counter once on the first (*neo*) rising edge (*ntp*, start of N condition) of the high active (*n_pol*) signal N after *t* (bit 9) has been set. Additionally A and B have to be high (*b_pol* and *a_pol*), otherwise the counter is not reset.

Encoder flag

With the setup from the Encoder constant example above and the *enc_flag* parameter set to 2, the ENC_FLAG pin toggles every 4 positions resulting in 1250 pin changes or 625 pulses per revolution

9.5 Current measurement and current regulator (Address $0B to $0D)

Measurement of the phase currents

The TMC332 uses a single current sensing resistor for each motor driver half bridge. Thus there are four sense resistors for two phase motors and three resistors for three phase motors. This allows a current measurement in each chopper state. Each sense resistor is connected to a signal conditioning circuit, creating an interface to the TMC332 that can measure the voltage drop on all resistors in parallel. The conversions are started on certain points in the PWM period, once in the three phase stepper mode, twice in the two phase stepper mode.

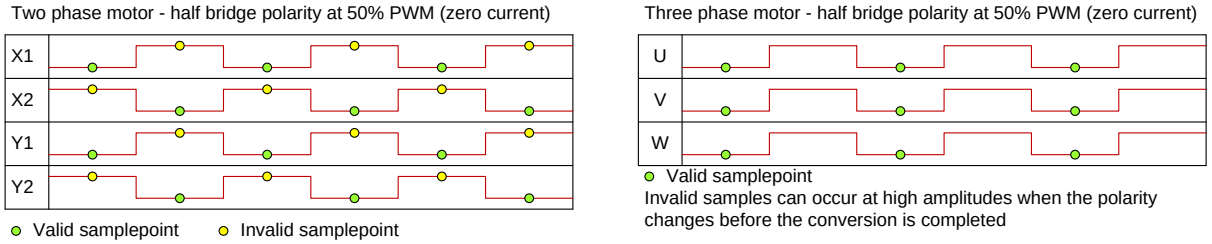

Figure 8 - Conversion start timing for current meaurement

For calculation of the total current, at least two valid measurement results are required. In three phase mode, the third value can be easily determined by the knowledge that the sum of all three currents equals zero ($I_U+I_V+I_W = 0$). In two phase mode, one current value of each motor phase is required. In the case that only one phase current in two phase mode or one current value at all in three phase mode can be acquired, no current regulation is done in this cycle.

Calculating the total current
When all required phase current values are present, the first step is to calculate the total current through the motor. In two phase mode, the Pythagorean Theorem is used:

$$I_{Motor} = \sqrt{(I_{Phase\,x})^2 + \left(I_{Phase\,y}\right)^2}$$

In three phase mode, the calculation is slightly simplified by not using Park's transformation but also the Pythagorean Theorem with all three phase currents:

$$I_{Motor} = \sqrt{(I_U)^2 + (I_V)^2 + (I_W)^2}$$

The resulting difference to Park's transformation is a constant factor of $\sqrt{3/2} \approx 1.2247$.

Calculating the error
After calculating the total motor current, the difference to the target motor current is calculated, resulting in the current error.

Tolerance band
To avoid oscillation near the target current value, a tolerance band can be defined. Current regulation will only be done, if the absolute value of the error exceeds the tolerance. The error will be set to 0 in this case.

The P-parameter
The error – or 0, if the error was in the tolerance band – is multiplied with the P-parameter (proportional gain) to calculate the correction value.

Correction of the PWM amplitude
The correction value is added to the value of the PWM amplitude. Before setting the new PWM amplitude, the sum is tested to be within certain limits. It must not be negative and it must not exceed the highest allowed value, which is 127 for two phase motors and 155 (127*1.2247, see above) for three phase motors. Results out of these limits are changed to 0 for negative results and 127 or 155 for too high results.

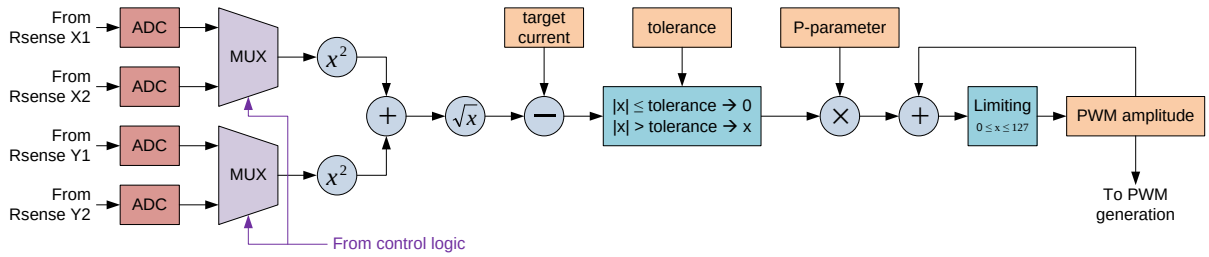

Figure 9 - TMC332 current regulation on two phase motors

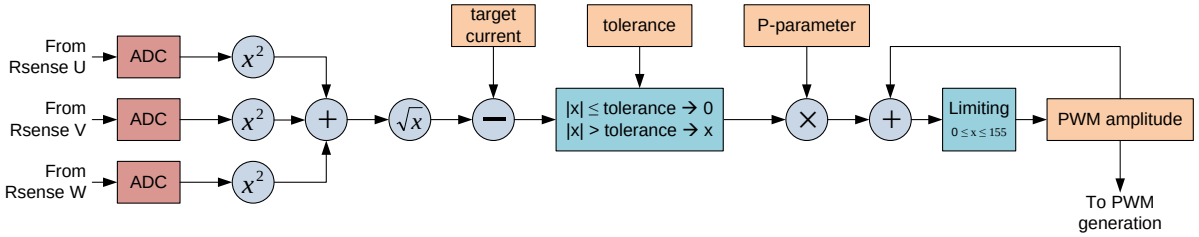

Figure 10 - TMC332 current regulation on three phase motors

In the application

The measured current for each half bridge can be read out via both current registers (*$0B* and *$0C*), where register *$0B* contains the currents for all three bridges of a three phase motor or the first three bridges of a two phase motor (U/X1, V/X2, W/Y1) and register *$0C* contains the current from the fourth half bridge used for two phase motors (Y2), the measurement error flags and the calculated total current.

The current at a given measured value can be calculated with this formula (R2 is the resistor at the COMP_ADC pin and C is the capacitor to ground):

$$I = (2 * i\_ / f_{CLK}) * 3.3V / (R_2 * C * R_{sense})$$

*i_* denotes the current value of either one phase (*i_u*, *i_v*, *i_w* or *i_y2* on two or three phase motors) or the calculated absolute current (*i_abs*) for two phase motors. To calculate the absolute current of a three phase motor, a the total current has to be divided by sqrt(3)/sqrt(2) = 1.2247:

$$I_{total\_3phase} = ((2 * i\_abs / f_{CLK}) * 3.3V / (R_2 * C * R_{sense})) / 1.2247$$

The limits for these values are -124 to 124 for the phase currents, 0 to 124 for the total current with two phase motors and 0 to 151 for three phase motors. A value of +/-127 as single phase current means the current is higher than the valid measuring range and the PWM-amplitude is reduced with current regulation. Absolute current values higher than 124/151 can occur in cases of resonance or high Back EMF, e.g. when the motor is stopped with high deceleration.

To calculate the current setpoint value for the current regulator, the formula needs to be rearranged:

$$targ\_curr_{2phase} = I_{max} * R_{sense} * R_2 * C * f_{CLK} / (2 * 3.3V)$$

$$targ\_curr_{3phase} = 1.2247 * I_{max} * R_{sense} * R_2 * C * f_{CLK} / (2 * 3.3V)$$

Best regulation results can be achieved with a setpoint at about 80. The sense resistors and the RC components should therefore be chosen depending on the desired motor current.
If the Limit-flag in register *$0C* is set (*bit 12*), the current regulator could not reach the current set in register *$8C* with the given *pwm_ampl* limit.

9.5.1 Additional notes

The following order should be followed when enabling the current regulator to prevent malfunctions:

- Set the tolerance and the P-parameter
- Set the target current and the enable bit of the regulator
- Increase the current limit (*pwm_ampl*) during regulated operation

When using gate drivers that can be disabled (e.g. the TMC603), the TMC332 does not notice a disabled driver but measures no current, which makes the regulator raise the PWM amplitude. When

the driver is enabled again, the current can be too high for a moment until the regulator has lowered the PWM amplitude again. With the TMC603, an overcurrent condition can occur. This can be avoided by setting the target current to 0 before disabling the gate driver and enabling the gate driver before setting the target current to the previous value.

9.5.2 Example

First the target current value has to be calculated using the formula above.
Using a two phase stepper motor, a target current of 0.5 A (absolute value) should be achieved with the TMC332 evaluation board, which has the following resistor and capacitor values: R_{sense} = 0.1 Ohm, R_2 = 150 kOhm, C = 1 nF, f_{CLK} = 16 MHz.

targ_curr = 0.5 A * 0.1 Ohm * 150 kOhm * 1 nF * 16 MHz / 3.3V = **18** (rounded)

Now the regulator can be set up. First the tolerance and the P-parameter are set up with the datagram **$8D 02 00 20** which means that the PWM amplitude is kept as long as the actual current and the target current differ no more than 12.5% of the target current and that the difference is multiplied with (**$00 20** / 4096) = 1 / 128 = 7.8125*10^-3 and added to the actual PWM amplitude when the difference is greater than the tolerance.
Next the target current value is set and the regulator is enabled with the datagram **$8C 00 01 12**. The **$01** is the enable bit for the current regulator and the **$12** is the value calculated before (**18**) in hexadecimal.
Now when reading out the second current register one might see the limit bit set (**$0C 00 00 00** -> **$00 xx 1x yy** with **yy** being lower than the previously set PWM amplitude limit), which means that the regulator cannot reach the target current with the actual maximum PWM amplitude. The maximum PWM amplitude can be set to **$FF** (by sending **$86 00 00 FF**) now.

9.6 Automatic micro-/full step switching (Address $0E)

A better performance with higher speeds can be achieved with full stepping. To avoid the disadvantages of full stepping in low speeds the best way is to switch from micro stepping to full stepping at a certain velocity. This can be done by the microcontroller or even better automatically by the TMC332.
The micro-/full step switching can be parameterized with register $8E. To enable full step mode permanently, the *enable permanently bit* (bit 17) and the current limit *FS_targ_curr* (bits 12 ... 8) have to be set. To calculate the current limit parameter (in the range from 0 to 31) this formula can be used:

$$FS\_targ\_curr = I_{max} * 32 * R_{sense} * (R_1 + R_2) / (3.3V * R_2)$$

On the Eval Board, the resistor values are R_1 = 1 MΩ, R_2 = 150 kΩ, R_{sense} = 0.1 Ω resulting in a maximum current of about 4.17 A. With the values used on the Eval Board there a difference of 1 of FS_targ_curr equals about 135 mA of current.

For automatic switching, the *automatic bit* (bit 16), the *FS current limit* parameter and the *FS_threshold* parameter have to be set.
The *FS_threshold* parameter describes the minimal time between two phase changes. The formula to calculate the needed value is:

$$FS\_threshold = f_{CLK} / (256*f_{fullstep})$$

There is also a hysteresis to avoid oscillating between micro- and full step mode when the step frequency is close to the switching point and has small variations.

A good threshold with enabled current regulator is when the regulated PWM amplitude reaches 240 ($F0) with the needed torque applied on the axis. If the motor has resonances in microstep mode at lower speeds, the threshold can be set at a lower point to avoid these resonances.

9.6.1 Additional notes

The *chopper_mode* bit defines how the bridges are controlled during full step choppering. The bridge connecting the coil to the supply voltage is choppered to limit the motor current.
If the *chopper_mode* bit is 0, one end of the coil is left floating while the other stays connected to GND, if the bit is 1 both ends of the coil are connected to GND.

Figure 11 shows the theory of chopper operation on one coil of a two phase motor. When driving a three phase motor in full step mode, the single motor coil would be replaced by two connections of the motor while the third connection is floating (neither the upper nor the lower transistor are on).

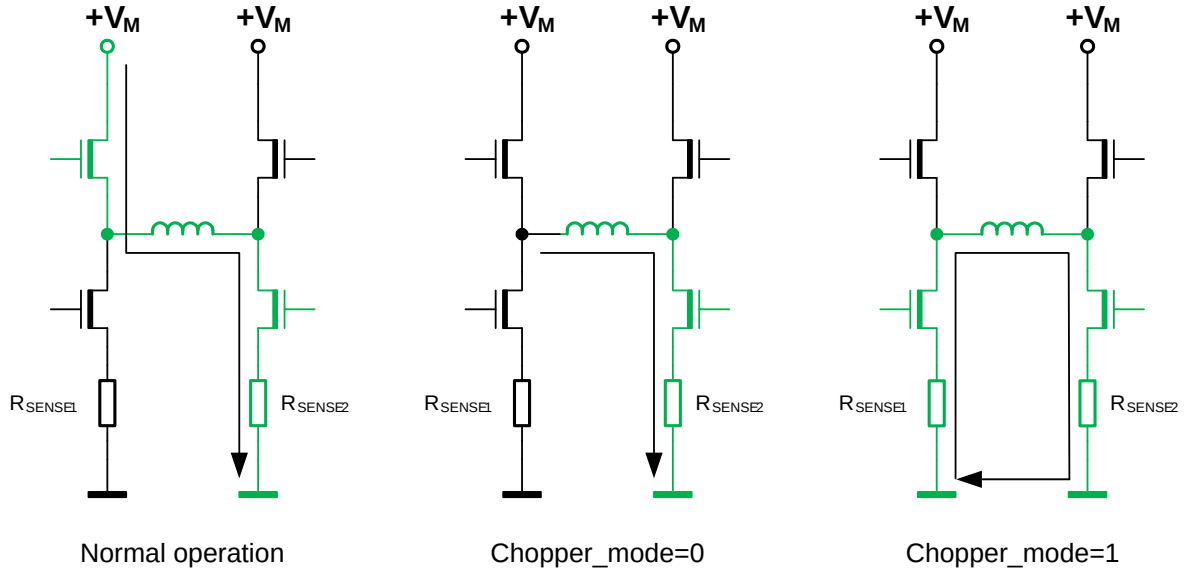

Figure 11: Full step chopper modes

9.6.2 Example

A two phase stepper motor with 200 steps per revolution should be switched from micro- to full stepping at 150rpm. 150rpm equal 30000 full steps/min or 500 full steps/s, so 16000000 / (256*500) = 125 has to be set as *FS_threshold*.

9.7 Calculation of rotational speed

The rotational speed of the motor when applying a given micro step frequency can be calculated with this formula:

$$v = f_{\mu s} * (PHI/1024) * (FS_P/FS_R)$$

v: rotational speed in rps
$f_{\mu s}$: micro step frequency in steps per second
PHI: the phi_sum parameter (see 9.2 Micro step width)
FS_P: the number of full steps per electrical period, 4 for two phase motors, 6 for three phase motors
FS_R: the number of full steps per revolution of the motor

9.7.1 Example for high rotational speeds

The following values are the results from tests with the TMC332 Eval Board (24V motor voltage). The speeds can differ with other motors – even of the same type – and of course with different transistors. The usable torque at such high speeds is low.

Trinamic QMot QSH 4218-41-10-035

Parameter	Value	Datagram
tolerance	12.5%	**$8D 02 00 1E**
p_parameter	30/4096	
current regulator on (*r*)	1	**$8C 00 01 32**
targ_curr	50	
Full step chopper mode (*d*)	0	**$8E 01 06 40**
Automatic switching (*e*)	1	
FS_targ_curr	6	
FS_thr	64	
pwm_ampl (limit for regulator)	255	**$86 01 00 FF**
phi_int	64	**$84 00 00 40**
phi_frac	0	**$83 00 00 00**

The settings on the TMC428 are: P_MUL = 200; P_DIV = 7; PULSEDIV = 2; RAMPDIV = 3; Acceleration = 4; Velocity = 1750

With these settings, a speed of approximately 8000 rpm is reached.

10 Notation of Number Systems

Decimal numbers are used as usual without additional identification. Binary numbers are identified by a prefixed % character. Hexadecimal numbers are identified by a prefixed $ character. So, for example the decimal number 42 in the decimal system is written as %101010 in the binary number system, and it is written as $2A in the hexadecimal number system. In binary and hexadecimal numbers x is used as don't care, so %1100x010 can be %11000010 or %11001010 or $4x can be anything from $40 to $4F. Don't care means that either the TMC332 ignores the state of the bit(s) or the state is unimportant for the explanation of a reply from the TMC332.

11 Revision History

Version	Date (Initials)	Comment
1.00	November 3rd, 2008 (SL)	first published version
	November 4th, 2008 (LL)	package image updated
1.01	November 26th, 2008 (SL)	added formula for current calculation
	December 11th, 2008 (SL, LL)	core voltage within feature list corrected to 1.5V (section 1, page 1), headline packaging TMC332 (section 5.1.1, page 7), TMC454 updated with TMC457 within drawing outline Figure 4, page 9.
	December 17th, 2008 (SL, LL)	Outline TMC457 with TMC332, Figure 4, page 9 corrected (SPI interface between TMC457 and TMC332)
2.00		Changes for TMC332 V2.00
	March 20th, 2009 (SL)	Documentation for Register $07 and $0E (read access)
	March 23th, 2009 (SL)	Different current calculation for three phase motors added
	April 8th, 2009 (SL)	Current regulator modifications (Tolerance and P-parameter registers)
		Updated Figures 5 and 6 with changed values, removed Encoder Flag Output from datasheet (has been removed from chip)
2.01	April 17th, 2009 (SL)	Description for SC_PD_AMOUNT (7.7.1) Determination of bbm_dly parameter (9.3)
2.02	September 10th, 2009 (SL)	Changed Pins M11, L10 and L12 to ground connection (Table 5-1)
2.03	September 17th, 2009 (SL)	Current controller description
2.04	November 9th, 2009 (SL)	Functional description of current measurement and regulation (9.5)
2.05	January 8th, 2010 (SL)	Added Step/Direction interface pin section (0)
2.06	April 20th, 2010 (SL)	Added TMC428 configuration information (7.1)

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 by request to info@trinamic.com