Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

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

Contact us

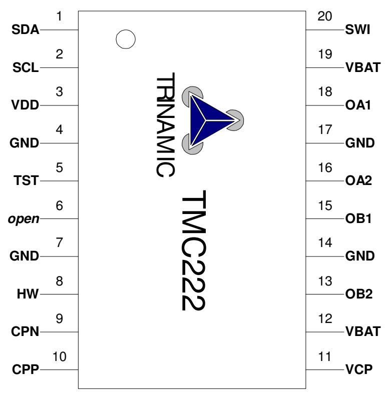
Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

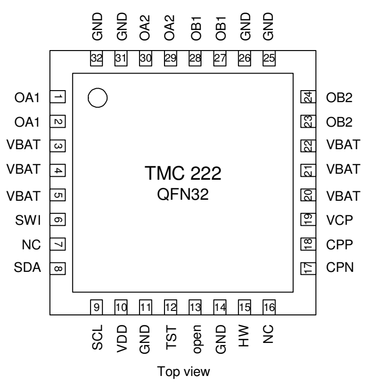
TMC222 – DATASHEET

Micro Stepping Stepper Motor Controller / Driver with Two Wire Serial Interface

TRINAMIC Motion Control GmbH & Co. KG
Waterloohain 5
D – 22769 Hamburg
GERMANY

www.trinamic.com





1 Features

The TMC222 is a combined micro-stepping stepper motor motion controller and driver with RAM and OTP memory. The RAM or OTP memory is used to store motor parameters and configuration settings. The TMC222 allows up to four bit of micro stepping and a coil current of up to 800 mA. After initialization it performs all time critical tasks autonomously based on target positions and velocity parameters. Communications to a host takes place via a two wire serial interface. Together with an inexpensive micro controller the TMC222 forms a complete motion control system. The main benefits of the TMC222 are:

- **Motor driver**
  - Controls one stepper motor with four bit micro stepping
  - Programmable Coil current up to 800 mA
  - Supply voltage range operating range 8V ... 29V
  - Fixed frequency PWM current control with automatic selection of fast and slow decay mode
  - Full step frequencies up to 1 kHz
  - High temperature, open circuit, short, over-current and under-voltage diagnostics
- **Motion controller**
  - Internal 16-bit wide position counter
  - Configurable speed and acceleration settings
  - Build-in ramp generator for autonomous positioning and speed control
  - On-the-fly alteration of target position
  - reference switch input available for read out
- **Two wire serial interface**
  - Transfer rates up to 350 kbps
  - Diagnostics and status information as well as motion parameters accessible
  - Field-programmable node addresses (32)

Copyright © 2004-2009 TRINAMIC Motion Control GmbH & Co. KG

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.

© TRINAMIC Motion Control GmbH & Co. KG 2011

Information given in this data sheet is believed to be accurate and reliable. However no responsibility is assumed for the consequences of its use nor for any infringement of patents or other rights of third parties which may result form its use.

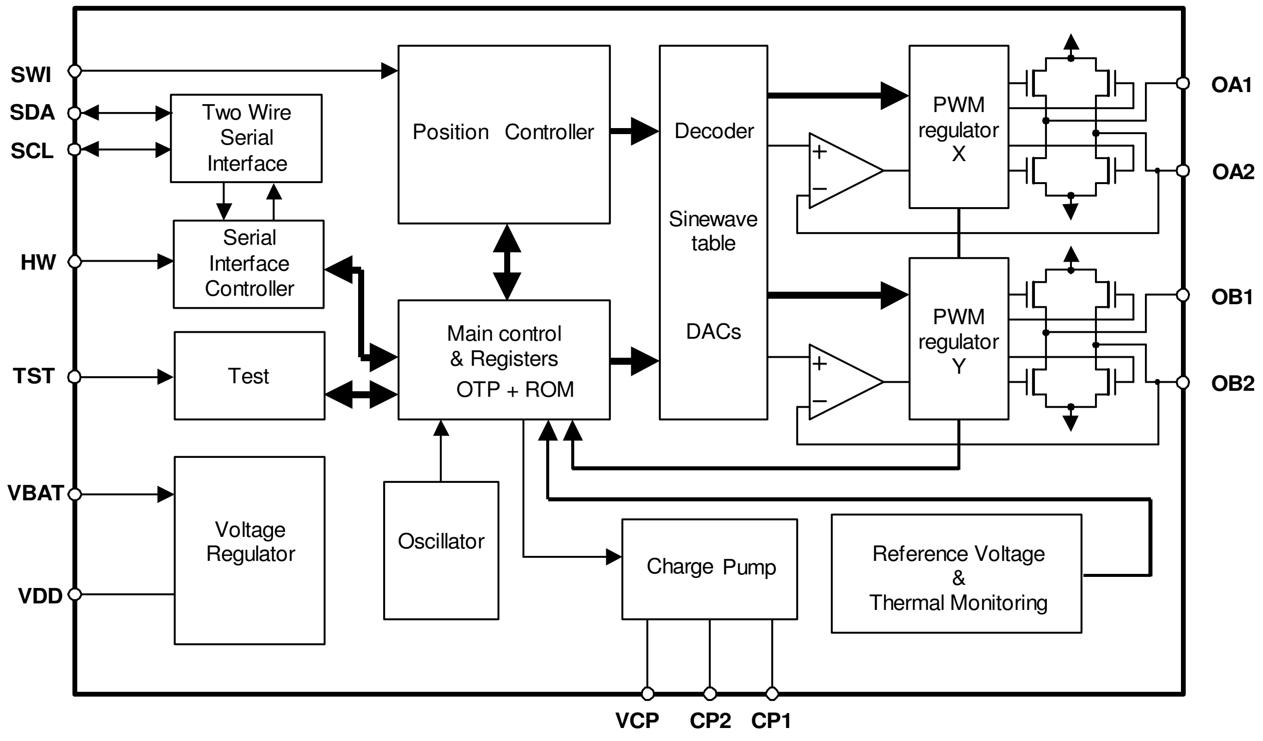
Specifications subject to change without notice.

Table of Contents

Copyright © 2004-2009 TRINAMIC Motion Control GmbH & Co. KG

2 General Description

2.1 Block Diagramm



2.2 Position Controller / Main Control

Motor parameters, e.g. acceleration, velocity and position parameters are passed to the main control block via the serial interface. These information are stored internally in RAM or OTP memory and are accessible by the position controller. This block takes over all time critical tasks to drive a stepper motor to the desired position under abiding the desired motion parameters.
The main controller gets feedback from the stepper motor driver block and is able to arrange internal actions in case of possible problems. Diagnostics information about problems and errors are transferred to the serial interface block.

2.3 Stepper Motor Driver

Two H-bridges are employed to drive both windings of a bipolar stepper motor. The internal transistors can reach an output current of up to 800 mA. The PWM principle is used to force the given current through the coils. The regulation loop performs a comparison between the sensed output current and the internal reference. The PWM signals to drive the power transistors are derived from the output of the current comparator.

2.4 Two Wire Serial Interface

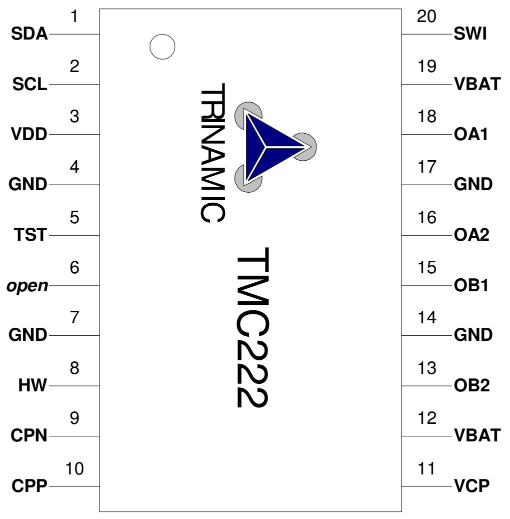
Communication between a host and the TMC222 takes places via the two wire bi-directional serial interface. Motion instructions and diagnostics information are provided to or from the Main Control block. It is possible to connect up to 32 devices on the same bus. Slave addresses are programmable via OTP memory or an external pin.

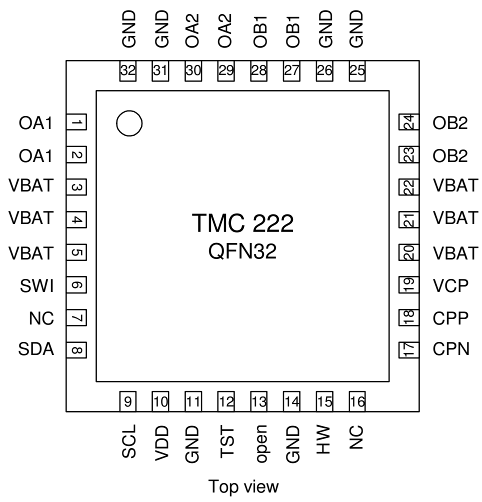
2.5 Miscellaneous

Besides the main blocks the TMC222 contains the following:

- an internal charge pump used to drive the high side transistors.
- an internal oscillator running at 4 MHz +/- 10% to clock the two wire serial interface, the positioning unit, and the main control block
- internal voltage reference for precise referencing
- a 5 Volts voltage regulator to supply the digital logic
- protection block featuring Thermal Shutdown, Power-On-Reset, etc.

2.6 Pin and Signal Descriptions





Name	SOIC20	QFN32	Description
SDA	1	8	SDA Serial Data input/output
SCL	2	9	SCL Serial Clock input
VDD	3	10	internal supply (needs external decoupling capacitor)
GND	4,7,14,17	11,14,25,26, 31,32	ground, heat sink
TST	5	12	test pin (to be tied to ground in normal operation)
open	6	13	*must be left open*
HW	8	15	hard-wired serial interface address bit input **Hint:** This is not a logic level input as usual; it needs to be connected via 1K resistor either to +VBAT or GND;
CPN	9	17	negative connection of external charge pump capacitor
CPP	10	18	positive connection of external charge pump capacitor
VCP	11	19	connection of external charge pump filter capacitor
VBAT	12, 19	3-5,20-22	battery voltage supply (Vbb)
OB2	13	23,24	negative end of phase B coil
OB1	15	27,28	positive end of phase B coil
OA2	16	29,30	negative end of phase A coil
OA1	18	1,2	positive end of phase A coil
SWI	20	6	reference switch input; **Hint:** This is not a logic level input as usual; it needs to be connected via 1K resistor either to +VBAT or GND;
NC		7,16	internally not connected (shields when connected to ground)

Table 1: TMC222 Signal Description

3 Typical Application

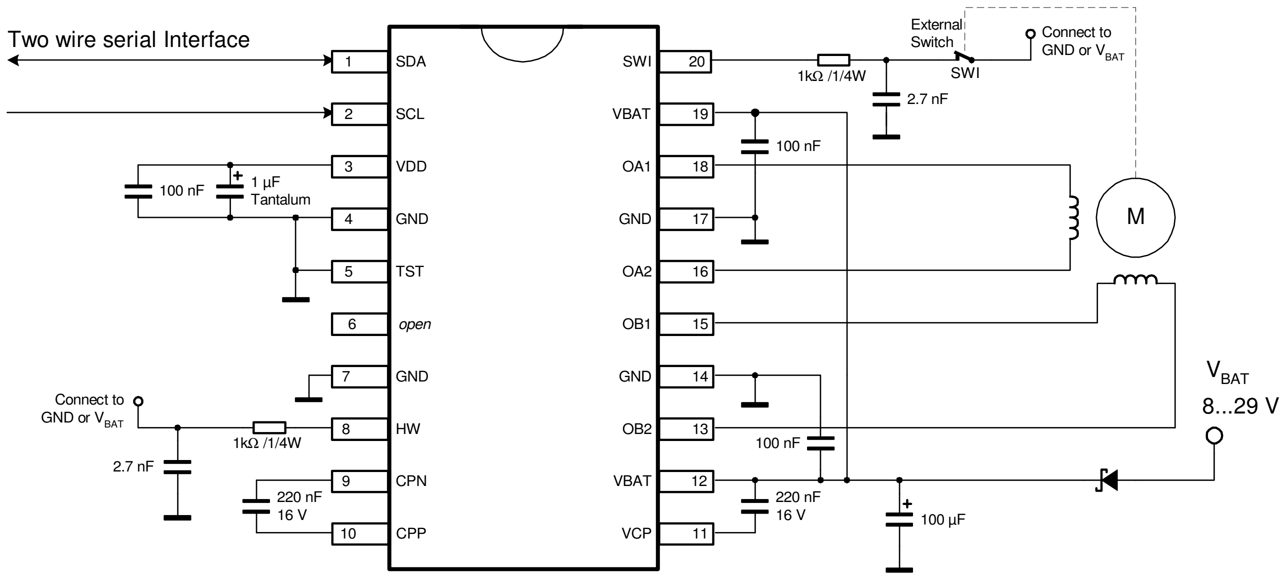


Figure 1: TMC222 Typical Application

Notes :

- Resistors tolerance +- 5%
- 2.7nF capacitors: 2.7nF is the minimum value, 10nF is the maximum value
- the 1µF and 100µF must have a low ESR value
- 100nF capacitors must be close to pins V_{BB} and V_{DD}
- 220nF capacitors must be as close as possible to pins CPN, CPP, V_{CP} and V_{BB} to reduce EMC radiation.

4 Ordering Information

Part No.	Package	Peak Current	Temperature Range
TMC222-PI20 (pre-series marking, same IC as TMC222-SI)	SOIC-20	800 mA	-40°C..125°C
TMC222-SI	SOIC-20	800mA	-40°C..125°C
TMC222-LI	QFN32	800mA	-40°C..125°C

Table 2: Ordering Information

5 Functional Description

5.1 Position Controller and Main Controller

5.1.1 Stepping Modes

The TMC222 supports up to 16 micro steps per full step, which leads to smooth and low torque ripple motion of the stepping motor. Four stepping modes (micro step resolutions) are selectable by the user (see also Table 11):

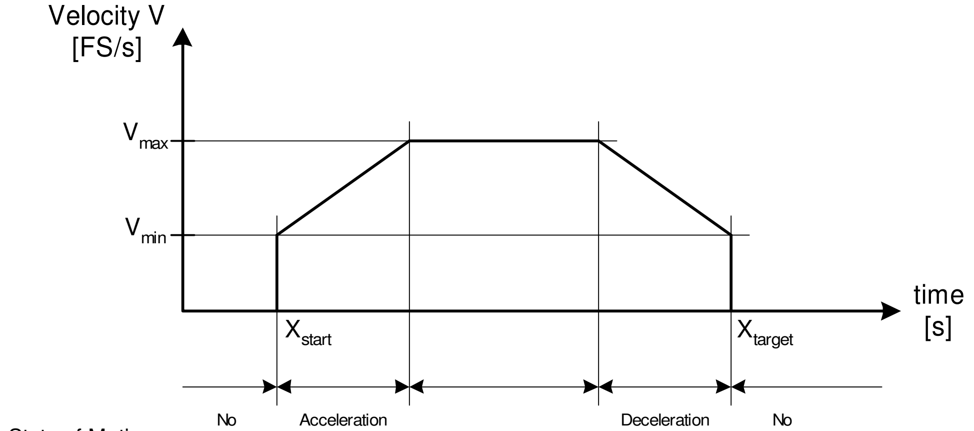
- Half step Mode
- 1/4 Micro stepping
- 1/8 Micro stepping
- 1/16 Micro stepping

5.1.2 Velocity Ramp

A common velocity ramp where a motor drives to a desired position is shown in the figure below. The motion consists of a acceleration phase, a phase of constant speed and a final deceleration phase. Both the acceleration and the deceleration are symmetrical. The acceleration factor can be chosen from a table with 16 entries. (Table 5: Acc Parameter on page 11). A typical motion begins with a start velocity Vmin. During acceleration phase the velocity is increased until Vmax is reached. After acceleration phase the motion is continued with velocity Vmax until the velocity has to be decreased in order to stop at the desired target position. Both velocity parameters Vmin and Vmax are programmable, whereas Vmin is a programmable ratio of Vmax. (See Table 3: Vmax Parameter on page 10 and Table 4: Vmin on page 11). The user has to take into account that Vmin is not allowed to change while a motion is ongoing. Vmax is only allowed to change under special circumstances. (See 5.1.4 Vmax Parameter on page 10).

The peak current value to be fed to each coil of the stepper-motor is selectable from a table with 16 possible values. It has to be distinguished between the run current Irun and the hold current Ihold. Irun is fed through the stepper motor coils while a motion is performed, whereas Ihold is the current to hold the stepper motor before or after a motion. More details about Irun and Ihold can be found in 5.3.1. and 5.3.2.

Velocity resp. acceleration parameters are accessable via the serial interface. These parameters are written via the SetMotorParam command (see 6.8.9) and read via the GetFullStatus1 command (see 6.8.1).



5.1.3 Examples for different Velocity Ramps

The following figures show some examples of typical motions under different conditions:

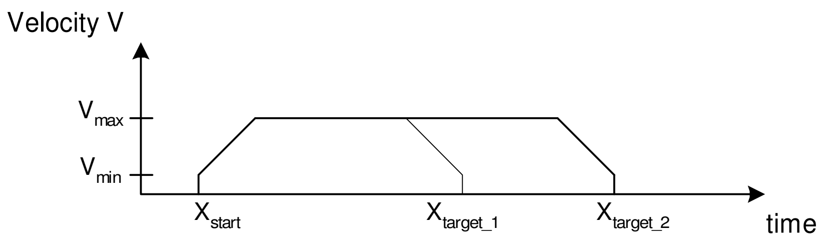


Figure 2: Motion with change of target position

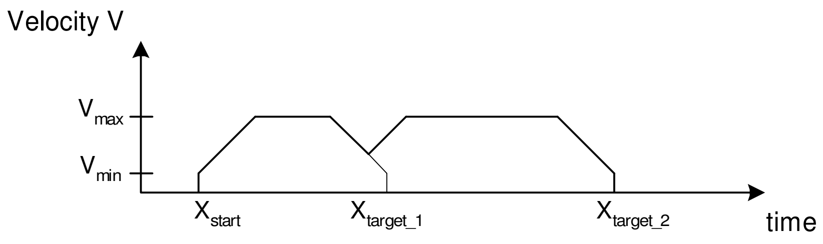


Figure 3: Motion with change of target position while in deceleration phase

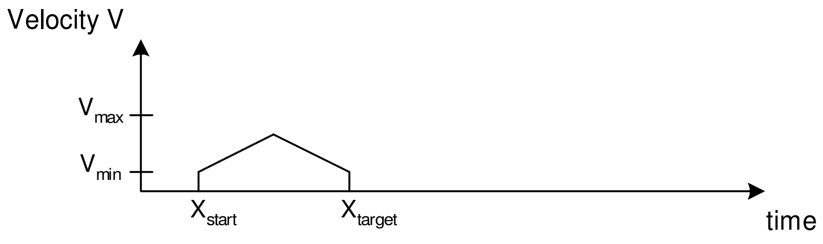


Figure 4: Short Motion Vmax is not reached

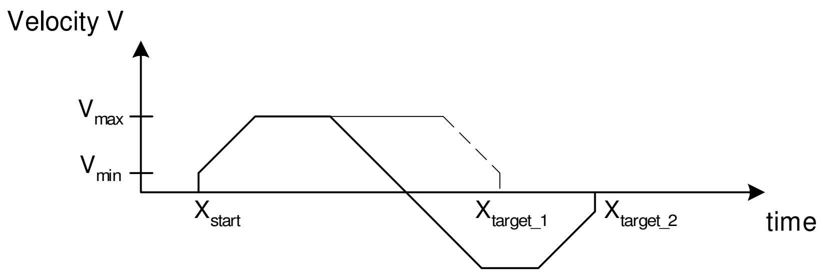


Figure 5: Linear Zero crossing (change of target position in opposite direction)

The motor crosses zero velocity with a linear shape. The velocity can be smaller than the programmed Vmin value during zero crossing. Linear zero crossing provides very low torque ripple to the stepper motor during crossing.

5.1.4 Vmax Parameter

The desired maximum velocity Vmax can be chosen from the table below:

Vmax index	Vmax [FS/s]	Vmax group	Stepping Mode			
			Half-Step Mode [half-steps/s]	1/4 micro stepping [micro-steps/s]	1/8 micro stepping [micro-steps/s]	1/16 micro stepping [micro-steps/s]
0	99	A	197	395	790	1579
1	136	B	273	546	1091	2182
2	167		334	668	1335	2670
3	197		395	790	1579	3159
4	213		425	851	1701	3403
5	228		456	912	1823	3647
6	243		486	973	1945	3891
7	273	C	546	1091	2182	4364
8	303		607	1213	2426	4852
9	334		668	1335	2670	5341
10	364		729	1457	2914	5829
11	395		790	1579	3159	6317
12	456		912	1823	3647	7294
13	546	D	1091	2182	4364	8728
14	729		1457	2914	5829	11658
15	973		1945	3891	7782	15564

Table 3: Vmax Parameter

Under special circumstances it is possible to change the Vmax parameters while a motion is ongoing. All 16 entries for the Vmax parameter are divided into four groups A, B, C and D. When changing Vmax during a motion take care that the new Vmax value is within the same group. Background: The TMC222 uses an internal pre-divider for positioning calculations. Within one group the pre-divider is equal. When changing Vmax between different groups during a motion, correct positioning is not ensured anymore.

5.1.5 Vmin Parameter

The minimum velocity parameter is a programmable ratio between 1/32 and 15/32 of Vmax. It is also possible to set Vmin to the same velocity as Vmax by setting Vmin index to zero. The table below shows the possible rounded values of Vmin given within unit [FS/s].

Vmin index	Vmax factor	Vmax group [A...D] and Vmax index [0…15]															
		A	B						C						D		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	99	136	167	197	213	228	243	273	303	334	364	395	456	546	729	973
1	1/32	3	4	5	6	6	7	7	8	8	10	10	11	13	15	19	26
2	2/32	6	8	10	11	12	13	14	15	17	19	21	23	27	30	42	57
3	3/32	9	12	15	18	19	21	22	25	27	30	32	36	42	50	65	88
4	4/32	12	16	20	24	26	28	30	32	36	40	44	48	55	65	88	118
5	5/32	15	21	26	30	32	35	37	42	46	52	55	61	71	84	111	149
6	6/32	18	25	30	36	39	42	45	50	55	61	67	72	84	99	134	179
7	7/32	22	30	36	43	46	50	52	59	65	72	78	86	99	118	156	210
8	8/32	24	33	41	49	52	56	60	67	74	82	90	97	112	134	179	240
9	9/32	28	38	47	55	59	64	68	76	84	94	101	111	128	153	202	271
10	10/32	30	42	52	61	66	71	75	84	94	103	112	122	141	168	225	301
11	11/32	34	47	57	68	72	78	83	94	103	114	124	135	156	187	248	332
12	12/32	37	50	62	73	79	85	91	101	112	124	135	147	170	202	271	362
13	13/32	40	55	68	80	86	92	98	111	122	135	147	160	185	221	294	393
14	14/32	43	59	72	86	92	99	106	118	132	145	158	172	198	236	317	423
15	15/32	46	64	78	92	99	107	114	128	141	156	170	185	214	256	340	454

Table 4: Vmin values [FS/s] for all Vmin index – Vmax index combinations

5.1.6 Acceleration Parameter

The acceleration parameter can be chosen from a wide range of available values as described in the table below. Please note that the acceleration parameter is not to change while a motion is ongoing.

Acceleration Values in [FS/s^2] dependent on Vmax																
Acc index	Vmax [FS/s]															
	99	136	167	197	213	228	243	273	303	334	364	395	456	546	729	973
0	49							106						473		
1	218													735		
2	1004															
3	3609															
4	6228															
5	8848															
6	11409															
7	13970															
8	16531															
9	14785	19092														
10		21886														
11		24447														
12		27008														
13		29570														
14		29570						34925								
15								40047								

Table 5: Acc Parameter

The amount of equivalent full steps during acceleration phase can be computed by the next equation:

$$N_{step} = \frac{V_{max}{}^2 - V_{min}{}^2}{2 \cdot Acc}$$

5.1.7 Position Ranges

Position information is coded by using two's complement format. Depending on the stepping mode (See 5.1.1) the position ranges are as listed in the following table:

Stepping Mode	**Position Range**	**Full range excursion**
Half-stepping	-4096…+4095 (-2^{12}…$+2^{12}-1$)	8192 half-steps 2^{13}
1/4 micro-stepping	-8192…+8191 (-2^{13}…$+2^{13}-1$)	16384 micro-steps 2^{14}
1/8 micro-stepping	-16384…+16383 (-2^{14}…$+2^{14}-1$)	32768 micro-steps 2^{15}
1/16 micro-stepping	-32768…+32767 (-2^{15}…$+2^{15}-1$)	65536 micro-steps 2^{16}

Table 6: Position Ranges

Target positions can be programmed via serial interface by using the SetPosition command (see 6.8.11). The actual motor position can be read by the GetFullStatus2 command (see 6.8.2).

5.1.8 Secure Position

The GotoSecurePosition command drives the motor to a pre-programmed secure position (see 6.8.4). The secure position is programmable by the user. Secure position is coded with 11 bits, therefore the resolution is lower than for normal positioning commands, as shown in the following table.

Stepping Mode	Secure Position Resolution
Half-stepping	4 half steps
1/4 micro stepping	8 micro steps (1/4th)
1/8 micro stepping	16 micro steps (1/8th)
1/16 micro stepping	32 micro steps (1/16th)

Table 7: Secure Position Resolution

5.1.9 External Switch SWI

Pin SWI (see Figure 1, on page 7) will attempt to source and sink current in/from the external switch pin. This is to check whether the external switch is open or closed, resp. if the pin is connected to ground or Vbat. The status of the switch can be read by using the GetFullStatus1 command. As long as the switch is open, the <ESW> flag is set to zero.

The ESW flag just represents the status of the input switch. The SWI input is intended as a physical interface for a mechanical switch that requires a cleaning current for proper operation. The SWI input detects if the switch is open or connected either to ground or to Vbat. The SWI input is not a digital logic level input. The status of the switch does not automatically perform actions as latching of the actual position. Those actions have to be realized by the application software.

Important Hint: The SWI is not a logic level input as usual; it needs to be connected via 1K resistor either to +VBAT or GND;

5.1.10 Motor Shutdown Management

The TMC222 is set into motor shutdown mode as soon as one of the following condition occurs:

- The chip temperature rises above the thermal shutdown threshold T_{tsd}. See 5.1.12 Temperature Management on Page 15
- The battery voltage drops below UV2 See 5.1.13 Battery Voltage Management on Page 16.
- An electrical problem occurred, e.g. short circuit, open circuit, etc. In case of such an problem flag <ElDef> is set to one.
- Chargepump failure, indicated by <CPFail> flag set to one.

During motor shutdown the following actions are performed by the main controller:

- H-bridges are set into high impedance mode
- The target position register TagPos is loaded with the contents of the actual position register ActPos.

The two-wire-serial-interface remains active during motor shutdown. To leave the motor shutdown state the following conditions must be true:

- Conditions which led to a motor shutdown are not active anymore
- A GetFullStatus1 command is performed via serial interface.

Leaving the motor shutdown state initiates the following

- H-bridges in Ihold mode
- Clock for the motor control digital circuitry is enabled
- The charge pump is active again

Now the TMC222 is ready to execute any positioning command.

IMPORTANT NOTE:
First, a GetFullStatus1 command has to be executed after power-on to activate the TMC222.

5.1.11 Reference Search / Position initialization

A stepper motor does not provide information about the actual position of the motor. Therefore it is recommended to perform a reference drive after power-up or if a motor shutdown happened in case of a problem. The RunInit command initiates the reference search. The RunInit command consists of a Vmin and Vmax parameter and also position information about the end of first and second motion (6.8.8 RunInit).

A reference drive consists of two motions (Figure 6: RunInit): The first motion is to drive the motor into a stall position or a reference switch. The first motion is performed under compliance of the selected Vmax and Vmin parameter and the acceleration parameter specified in the RAM. The second motion has got a rectangular shape, without a acceleration phase and is to drive the motor out of the stall position or slowly towards the stall position again to compensate for the bouncing of the faster first motion to stop as close to the stall position as possible. The maximum velocity of the second motion equals to Vmin. The positions of Pos1 and Pos2 can be chosen freely (Pos1 > Pos2 or Pos1 < Pos2). After the second motion the actual position register is set to zero. Finally, the secure position will be traveled to if it is enabled (different from the most negative decimal value of –1024).

Once the RunInit command is started it can not be interrupted by any other command except a condition occurs which leads to a motor shutdown (See 5.1.10 Motor Shutdown Management) or a HardStop command is received. Furthermore the master has to ensure that the target position of the first motion is **not** equal to the actual position of the stepper motor and that the target positions of the first and the second motion are not equal. This is very important otherwise the circuit goes into a deadlock state. Once the circuit finds itself in a deadlock state only a HardStop command followed by a GetFullStatus1 command will cause the circuit to leave the deadlock state.

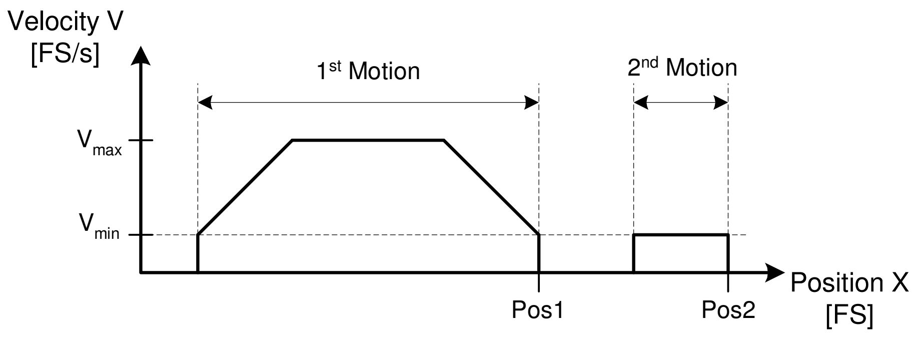
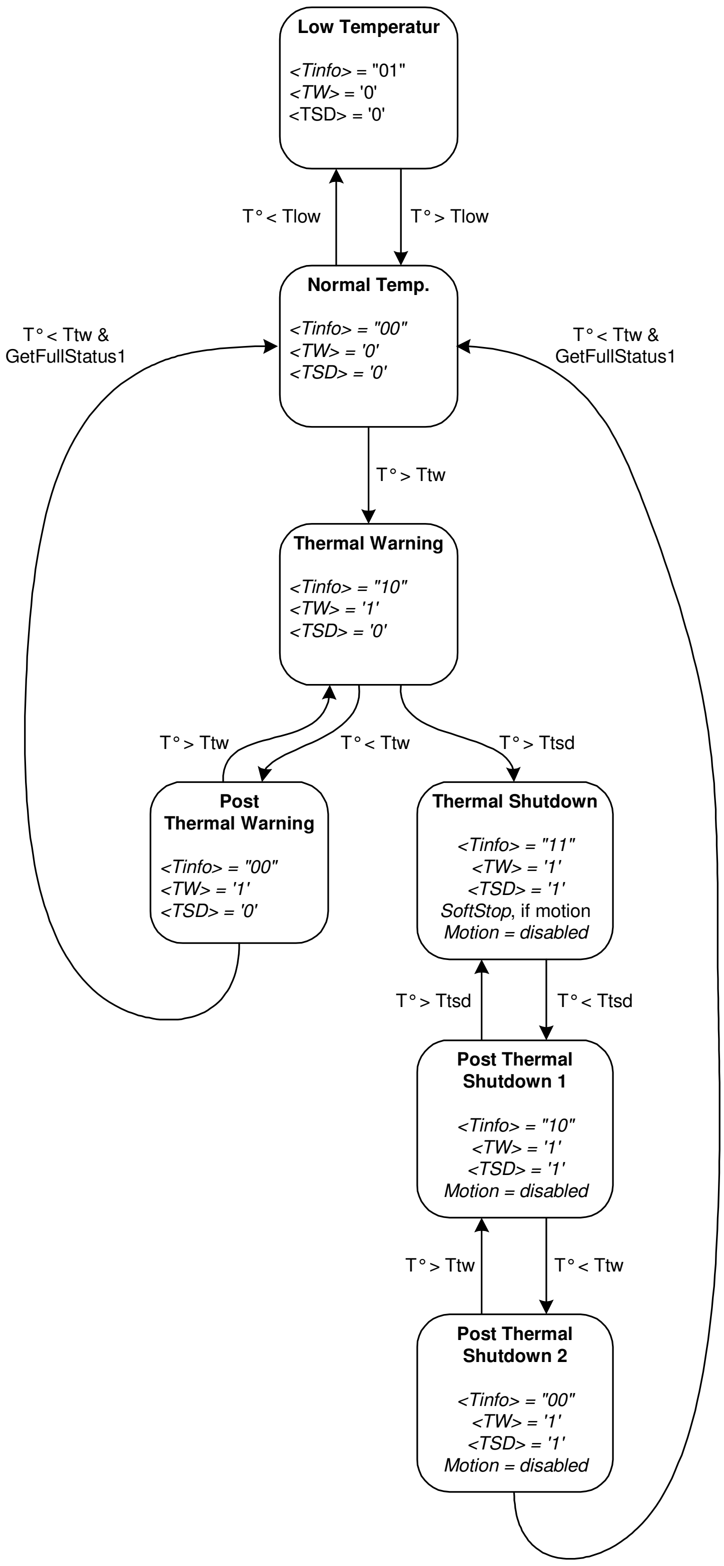


Figure 6: RunInit

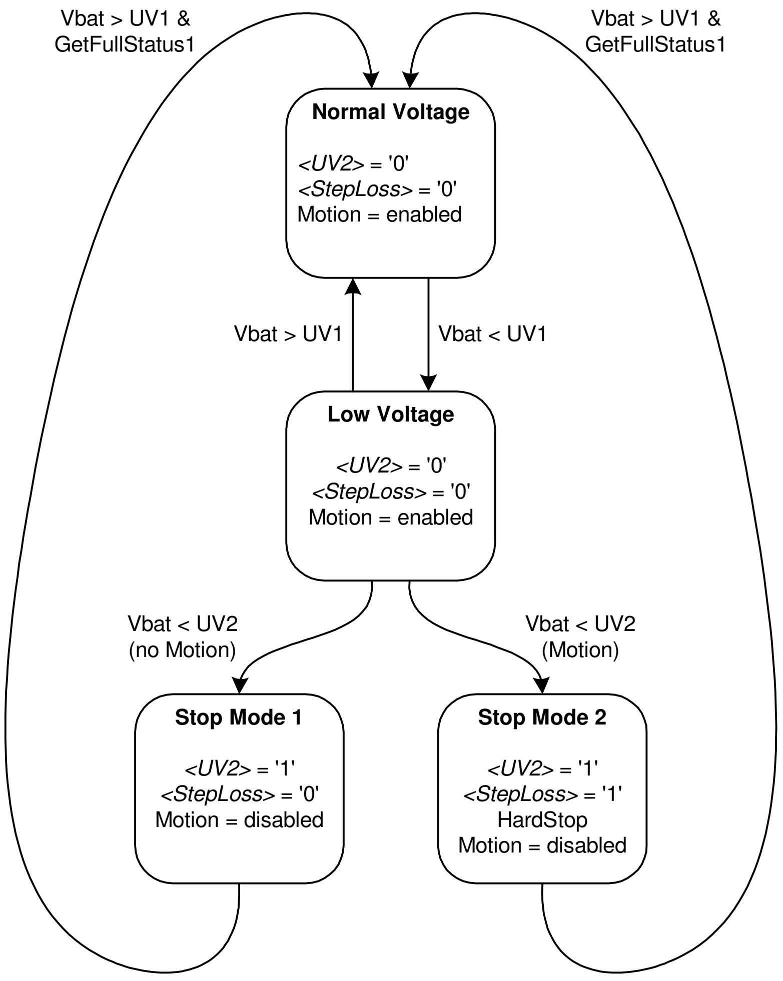
5.1.12 Temperature Management

The TMC222 provides an internal temperature monitoring. The circuit goes into shutdown mode if the temperature exceeds threshold T_{tsd}, furthermore two thresholds are implemented to generate a temperature pre-warning.



5.1.13 Battery Voltage Management

The TMC222 provides an internal battery voltage monitoring. The circuit goes into shutdown mode if the battery voltage falls below threshold UV2, furthermore one threshold UV1 is implemented to generate a low voltage warning.



5.1.14 Internal handling of commands and flags

The internal handling of commands and flags differs. Commands are handled with different priorities depending on the current state and the current status of internal flags, see figure below. SetPosition or GotoSecurePosition commands are ignored as long as the <StepLoss> flag is set. Details can be found in Table 8: Priority Encoder.

Note: A HardStop command is sent by the master or triggered internally in case of an electrical defect or over temperature.

A description of the available commands can be found in 6.8 Command Description. A list of the internal flags can be found in 5.2.2 Status Flags.

As an example: When the circuit drives the motor to its programmed target position, state "GotoPos" is entered. There are three events which can cause to leave this state: HardStop command received, SoftStop command received or target position reached. If all three events occur at the same time the HardStop command is executed since it has the highest priority. The Motion finished event (target position reached) has the lowest priority and thus will only cause transition to "Stopped" state when both other events do not occur.

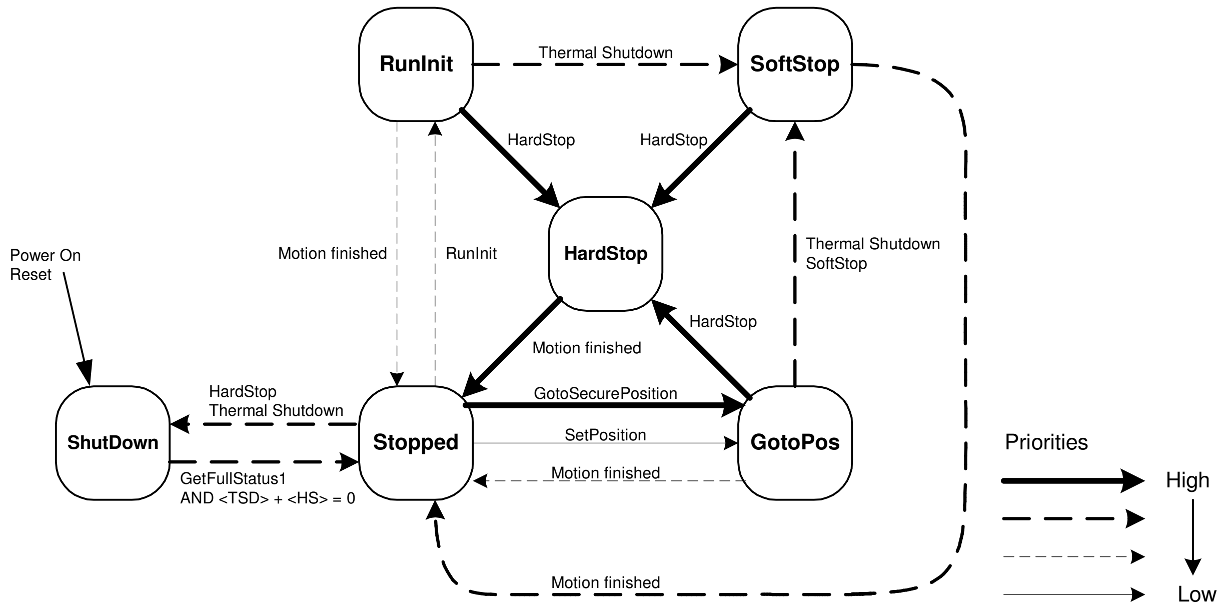


Figure 7: Internal handling of commands and flags

State →	**Stopped**	**GotoPos**	**RunInit**	**SoftStop**	**HardStop**	**ShutDown**
Command ↓	motor stopped, Ihold in coils	motor motion ongoing	no influence on RAM and `TagPos`	motor decelerating	motor forced to stop	motor stopped, H-bridges in Hi-Z
`GetFullStatus2`	I²C in-frame response	I²C in-frame response	I²C in-frame response	I²C in-frame response	I²C in-frame response	I²C in-frame response
`GetOTPParam`	OTP refresh; I²C in-frame	OTP refresh; I²C in-frame	OTP refresh; I²C in-frame	OTP refresh; I²C in-frame	OTP refresh; I²C in-frame	OTP refresh; I²C in-frame
`GetFullStatus1` [attempt to clear <TSD> and <HS> flags]	I²C in-frame response	I²C in-frame response	I²C in-frame response	I²C in-frame response	I²C in-frame response	I²C in-frame response; if (`<TSD>` or `<HS>`) = '1' then **→ Stopped**
`ResetToDefault` [`ActPos` and `TagPos` are not altered]	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset (note 2)	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset	OTP refresh; OTP to RAM; `AccShape` reset
`SetMotorParam` [Master takes care about proper update]	RAM update	RAM update	RAM update	RAM update	RAM update	RAM update
`ResetPosition`	`TagPos` and `ActPos` reset					`TagPos` and `ActPos` reset
`SetPosition`	`TagPos` updated; **→GotoPos**	`TagPos` updated	`TagPos` updated			
`GotoSecurePosition`	If `<SecEn>` = '1' then `TagPos` = `SecPos`; **→GotoPos**	If `<SecEn>` = '1' then `TagPos` = `SecPos`	If `<SecEn>` = '1' then `TagPos` = `SecPos`			
`RunInit`	**→RunInit**					
`HardStop`		**→HardStop**; `<StepLoss>` = '1'	**→HardStop**; `<StepLoss>` = '1'	**→HardStop**; `<StepLoss>` = '1'		
`SoftStop`		**→SoftStop**				
`HardStop` [⇔ (`<CPFail>` or `<UV2>` or `<ElDef>`) = '1' ⇒ `<HS>` = '1']	**→Shutdown**	**→HardStop**	**→HardStop**	**→HardStop**		
Thermal shutdown [`<TSD>` = '1']	**→Shutdown**	**→SoftStop**	**→SoftStop**			
Motion finished	n.a.	**→Stopped**	**→Stopped**	**→Stopped**; `TagPos` =`ActPos`	**→Stopped**; `TagPos` =`ActPos`	n.a.

Table 8: Priority Encoder

Color code:

░	Command ignored
▒	Transition to another state
▓	Master is responsible for proper update (see note 5)

Notes:

1 After Power on reset, the Shutdown state is entered. The Shutdown state can only be left after a GetFullStatus1 command (so that the Master could read the <VddReset> flag).

2 A RunInit sequence runs with a separate set of RAM registers. The parameters which are not specified in a RunInit command are loaded with the values stored in RAM at the moment the RunInit sequence starts. AccShape is forced to '1' during second motion even if a ResetToDefault command is issued during a RunInit sequence, in which case AccShape at '0' will be taken into account after the RunInit sequence. A GetFullStatus1 command will return the default parameters for Vmax and Vmin stored in RAM.

3 Shutdown state can be left only when <TSD> and <HS> flags are reset.

4 Flags can be reset only after the master could read them via a GetFullStatus1 command, and provided the physical conditions allow for it (normal temperature, correct battery voltage and no electrical or charge pump defect).

5 A SetMotorParam command sent while a motion is ongoing (state GotoPos) should not attempt to modify Acc and Vmin values. This can be done during a RunInit sequence since this motion uses its own parameters, the new parameters will be taken into account at the next SetPosition command.

6 <SecEn> = '1' when register SecPos is loaded with a value different from the most negative value (i.e. different from 0x400 = "100 0000 0000")

7 <Stop> flag allows to distinguish whether state Stopped was entered after HardStop/SoftStop or not. <Stop> is set to '1' when leaving state HardStop or SoftStop and is reset during first clock edge occurring in state Stopped.

8 While in state Stopped, if ActPos ≠ TagPos there is a transition to state GotoPos. This transition has the lowest priority, meaning that <Stop>, <TSD>, etc. are first evaluated for possible transitions.

9 If <StepLoss> is active, then SetPosition and GotoSecurePosition commands are ignored (they will not modify TagPos register whatever the state) and motion to secure position is forbidden. Other commands like RunInit or ResetPosition will be executed if allowed by current state. <StepLoss> can only be cleared by a GetFullStatus1 command.

5.2 RAM and OTP Memory

Some RAM registers (e.g. Ihold, Irun) are initialized with the content of the OTP (One Time Programmable) memory. The content of RAM registers that are initialized via OTP can be changed afterwards. This allows user initialization default values, whereas the default values are one time programmable by the user. Some OTP bits are address bits of the TMC222.

5.2.1 RAM Registers

Register	Mnemonic	Length (bit)	Related commands	Comment	Reset State
Actual Position	ActPos	16	GetFullStatus2 ResetPosition	Actual Position of the Stepper Motor. 16-bit signed	0x0000
Target Position	TagPos	16	SetPosition GetFullStatus2 ResetPosition	Target Position of the Stepper Motor. 16-bit signed	
Acceleration Shape	AccShape	1	GetFullStatus1 SetMotorParam ResetToDefault	0 = Acceleration with Acc Parameter. 1 = Velocity set to Vmin, without acceleration	
Coil Peak Current	Irun	4	GetFullStatus1 SetMotorParam ResetToDefault	Coil current when motion is ongoing (Table 12: Irun / Ihold Settings)	OTP Memory
Coil Hold Current	Ihold	4	GetFullStatus1 SetMotorParam ResetToDefault	Coil current when motor stands still (Table 12: Irun / Ihold Settings)	
Minimum Velocity	Vmin	4	GetFullStatus1 SetMotorParam ResetToDefault	Start Velocity of the stepper motor (Table 4: Vmin)	
Maximum Velocity	Vmax	4	GetFullStatus1 SetMotorParam ResetToDefault	Target Velocity of the stepper motor (Table 3: Vmax Parameter)	
Shaft	Shaft	1	GetFullStatus1 SetMotorParam ResetToDefault	Direction of motion	
Acceleration / Deceleration	Acc	4	GetFullStatus1 SetMotorParam ResetToDefault	Parameter for acceleration (Table 5: Acc Parameter)	
Secure Position	SecPos	11	GetFullStatus2 ResetToDefault	Target Position for GotoSecurePosition command (6.8.4 GotoSecurePosition); 11 MSBs of 16-bit position (LSBs fixed to '0')	
Stepping Mode	StepMode	2	GetFullStatus1 GetFullStatus2 ResetToDefault	Micro stepping mode (5.1.1 Stepping Modes)	

5.2.2 Status Flags

The table below shows the flags which are accessable by the serial interface in order to receive information about the internal status of the TMC222.

Flag	Mnemonic	Length (bit)	Related Command	Comment	Reset state
Digital supply Reset	VddReset	1	GetFullStatus1	Set to '1' after power-up or after a micro-cut in the supply voltage to warn that RAM contents may have been lost. Is set to '0' after GetFullStatus1 command.	'1'
Over current in coil A	OVC1	1	GetFullStatus1	Set to '1' if an over current in coil #1 was detected. Is set to '0' after GetFullStatus1 command.	'0'
Over current in coil B	OVC2	1	GetFullStatus1	Set to '1' if an over current in coil #2 was detected. Is set to '0' after GetFullStatus1 command.	'0'
StepLoss	StepLoss	1	GetFullStatus1	Set to '1' when under voltage, over current or over temperature event was detected. Is set to '0' after GetFullStatus1 command. SetPosition and GotoSecurePosition commands are ignored when <StepLoss> = 1	'0'
Secure position enabled	SecEn	1	Internal use	'0' if SecPos = "100 0000 0000" '1' otherwise	n.a.
Electrical Defect	ElDef	1	GetFullStatus1	Set to '1' if open circuit or a short was detected, (<OVC1> or <OVC2>). Is. Is set to '0' after GetFullStatus1 command.	'0'
Temperature Info	Tinfo	2	GetFullStatus1	Indicates the chip temperature "00" = normal temperature "01 = low temperature warning "10" = high temperature warning "11" = motor shutdown	"00"
Thermal Warning	TW	1	GetFullStatus1	Set to one if temperature raises above 145 ℃. Is set to '0' after GetFullStatus1 command.	'0'
Thermal Shutdown	TSD	1	GetFullStatus1	Set to one if temperature raises above 155° C. Is set to '0' after GetFullStatus1 command and Tinfo = "00".	'0'
Motion Status	Motion	3	GetFullStatus1	Indicates the actual behavior of the position controller. "000": Actual Position = Target Position; Velocity = 0 "001": Positive Acceleration; Velocity > 0 "010": Negative Acceleration; Velocity > 0 "011": Acceleration = 0 Velocity = maximum pos Velocity "100": Actual Position /= Target Position; Velocity = 0 "101": Positive Acceleration; Velocity < 0 "110": Positive Acceleration; Velocity < 0 "111": Acceleration = 0 Velocity = maximum neg Velocity	"000"
External Switch Status	ESW	1	GetFullStatus1	Indicates the status of the external switch. '0' = open '1' = close	'0'
Charge Pump failure	CPFail	1	GetFullStatus1	'0' charge pump OK '1' charge pump failure	'0'
Electrical flag	HS	1	Internal use	<CPFail> or <UV2> or <ElDef>	'0'

5.2.3 OTP Memory Structure

The table below shows where the OTP parameters are stored in the OTP memory.

Note: If the OTP memory has not been programmed, or if the RAM has not be programmed by a SetMotorParam command, or if anyhow <VddReset> = '1', any positioning command will be ignored, in order to avoid any consequence due to unwanted RAM content. Please check that the correct supply voltage is applied to the circuit before zapping the OTP (See: Table 21: DC Parameters Supply and Voltage regulator on page 45), otherwise the circuit will be destroyed.

OTP Address	OTP Bit Order							
	7	6	5	4	3	2	1	0
0x00	OSC3	OSC2	OSC1	OSC0	IREF3	IREF2	IREF1	IREF0
0x01		TSD2	TSD1	TSD0	BG3	BG2	BG1	BG0
0x02					AD3	AD2	AD1	AD0
0x03	Irun3	Irun2	Irun1	Irun0	Ihold3	Ihold2	Ihold1	Ihold0
0x04	Vmax3	Vmax2	Vmax1	Vmax0	Vmin3	Vmin2	Vmin1	Vmin0
0x05	SecPos10.	SecPos9	SecPos8	Shaft	Acc3	Acc2	Acc1	Acc0
0x06	SecPos7	SecPos6	SecPos5	SecPos4	SecPos3	SecPos2	SecPos1	SecPos0
0x07					StepMode1	StepMode0	LOCKBT	LOCKBG

Table 9 : OTP Memory Structure

Parameters stored at address 0x00 and 0x01 and bit LOCKBT are already programmed in the OTP memory at circuit delivery, they correspond to the calibration of the circuit and are just documented here as an indication. These might vary between different components. These bits (gray within Table 9 : OTP Memory Structure) should not be used after readout. Each OPT bit is at '0' when not zapped. Zapping a bit will set it to '1'. Thus only bits having to be at '1' must be zapped. Zapping of a bit already at '1' is disabled, to avoid any damage of the Zener diode. It is important to note that only one single OTP byte can be programmed at the same time (see command SetOTPParam).

Once OTP programming is completed, bit LOCKBG can be zapped, to disable unwanted future zapping, otherwise any OTP bit at '0' could still be zapped.

Lock bit	Protected byte
LOCKBT (zapped before delivery)	0x00 to 0x01
LOCKBG	0x02 to 0x07

Table 10 : OTP Lock bits

The command used to load the application parameters via the serial bus into the RAM prior to an OTP Memory programming is SetMotorParam. This allows for a functional verification before using a SetOTPParam command to program and zap separately one OTP memory byte. A GetOTPParam command issued after each SetOTPParam command allows to verify the correct byte zapping.

5.3 Stepper Motor Driver

The StepMode parameter in SetMotorParam command (6.8.9 SetMotorParam on page 34) is used to select between different stepping modes. Following modes are available:

StepMode parameter	**Mode**
00	Half Stepping
01	1/4 µStepping
10	1/8 µStepping
11	1/16 µStepping

Table 11: StepMode

5.3.1 Coil current shapes

The next four figures show the current shapes fed to each coil of the motor in different stepping modes.

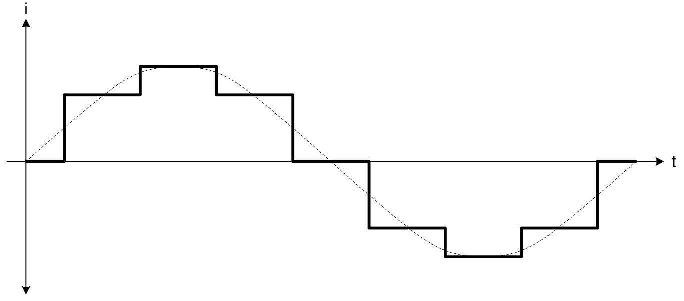


Figure 8: Coil Current for Half Stepping Mode

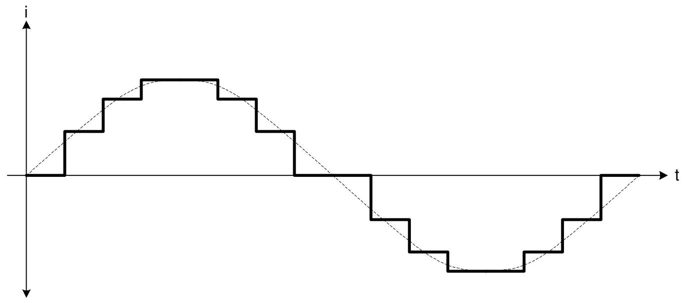


Figure 9: Coil Current for 1/4 Micro Stepping Mode

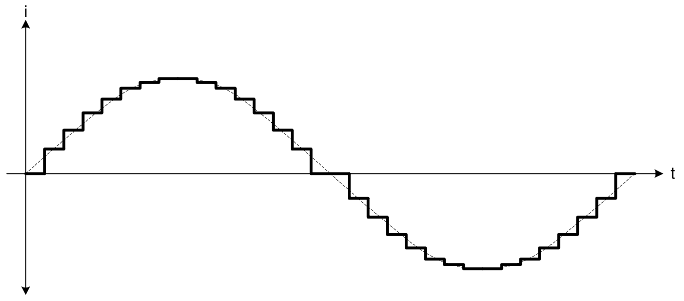


Figure 10: Coil Current for 1/8 Micro Stepping Mode

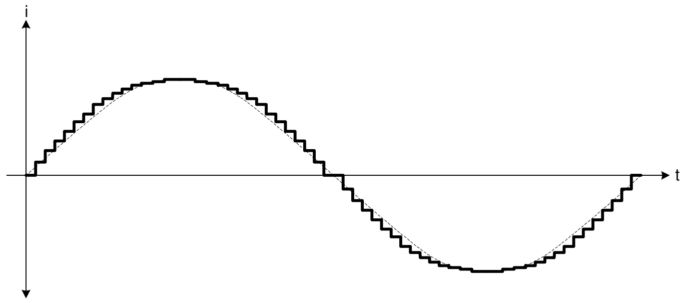


Figure 11: Coil Current for 1/16 Micro Stepping Mode

5.3.2 Transition Irun to Ihold

At the end of a motor motion the actual coil currents Irun are maintained in the coils at their actual DC level for a quarter of an electrical period (two half steps) at minimum velocity. Afterwards the currents are then set to their hold values Ihold. The figure below illustrates the mechanism:

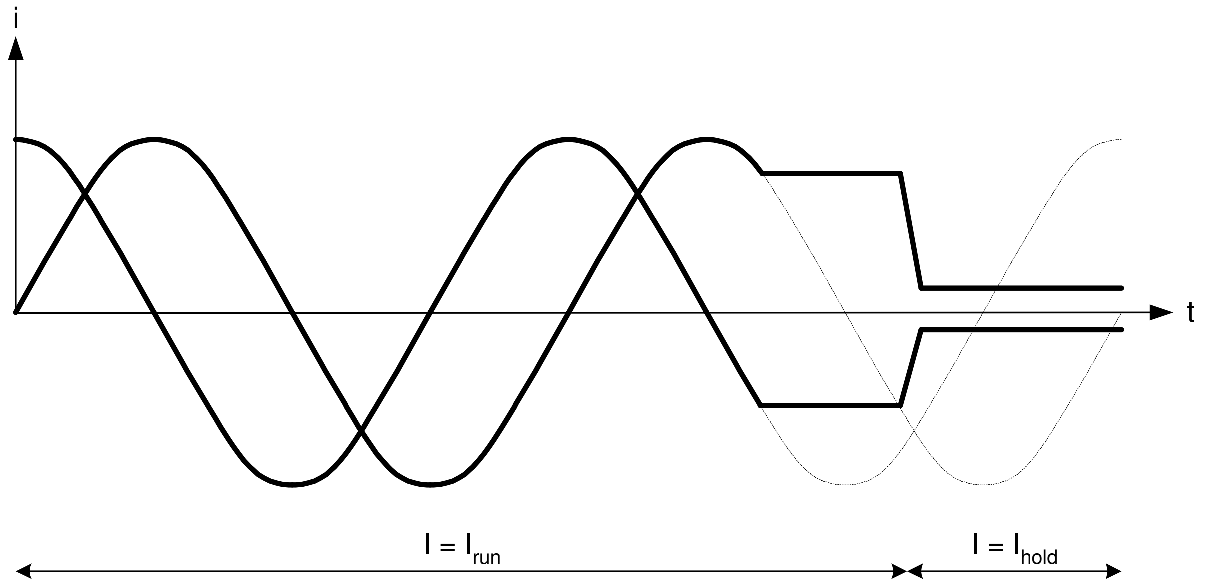


Figure 12: Transition Irun to Ihold

Both currents Irun and Ihold are parameterizeable using the command SetMotorParam. 16 values are available for Irun current and 16 values for Ihold current. The table below shows the corresponding current values.

Irun / Ihold setting (hexadecimal)	Peak Current [mA]
0x0	59
0x1	71
0x2	84
0x3	100
0x4	119
0x5	141
0x6	168
0x7	200
0x8	238
0x9	283
0xA	336
0xB	400
0xC	476
0xD	566
0xE	673
0xF	800

Table 12: Irun / Ihold Settings

5.3.3 Chopper Mechanism

The chopper frequency is fixed as specified in chapter 10.4 AC Parameters on page 46. The TMC222 uses an intelligent chopper algorithm to provide a smooth operation with low resonance. The TMC222 uses internal measurements to derive current flowing through coils. If the current is less than the desired current, the TMC222 switches a H-bridge in a way that the current will increase. Otherwise if the current is too high, the H-bridge will be switched to decrease the current. For decreasing two modes are available, slow decay and fast decay, whereas fast decay decreases the current faster than slow decay. The figure below shows the chopper behavior.

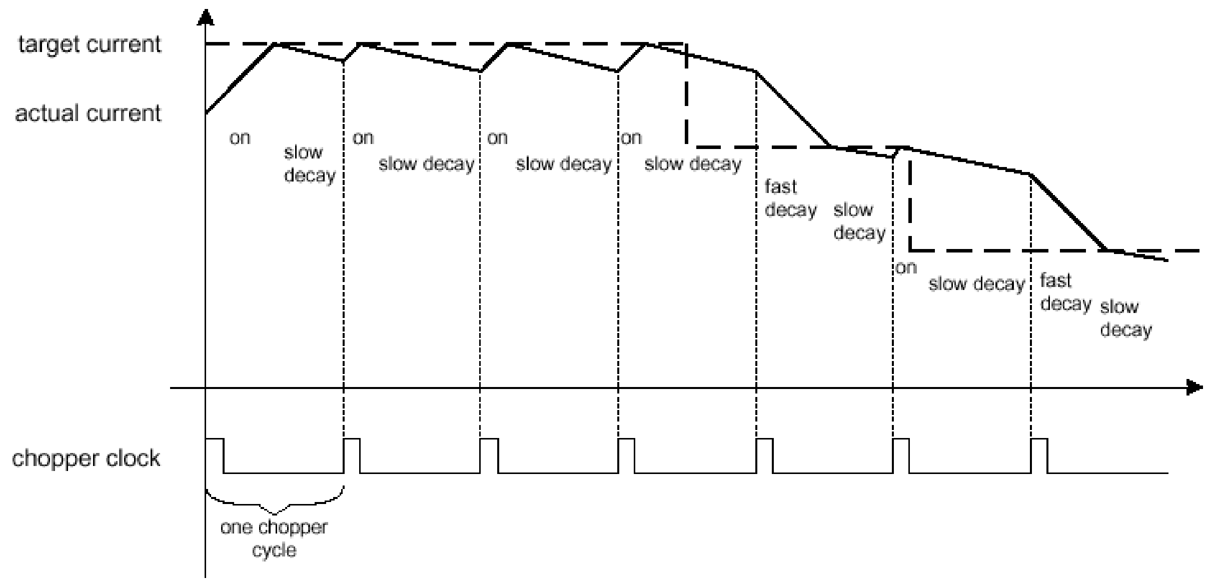


Figure 13: Different Chopper Cycles with Fast and Slow Decay

6 Two-Wire Serial Interface

6.1 Physical Layer

Both SDA and SCL lines are connected to positive supply voltage via a current source or pull-up resistor (see figure below). When there is no traffic on the bus both lines are high. Analog glitch filters are implemented to suppress spikes with a length of up to 50 ns.

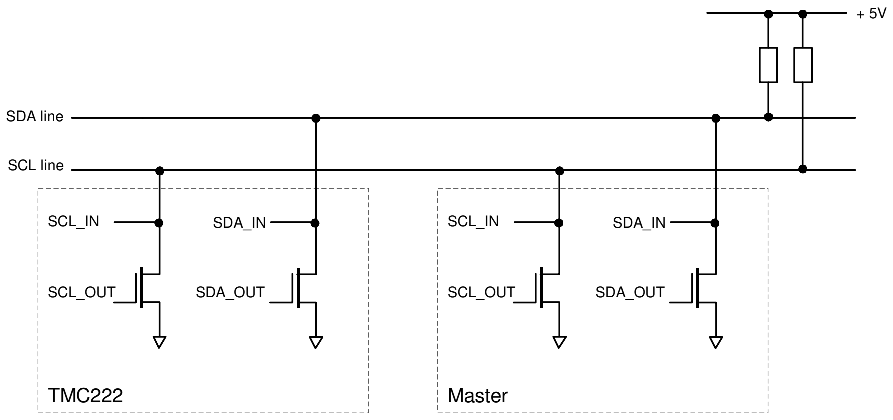


Figure 14: Two Wire Serial Interface - Physical Layer

6.2 Communication on Two Wire Serial Bus Interface

Each datagram starts with a Start condition and ends with a Stop condition. Both conditions are unique and cannot be confused with data. A high to low transition on the SDA line while SCL is high indicates a Start condition. A low to high transition on the SDA line while SCL is high defines a Stop condition (see figure below).

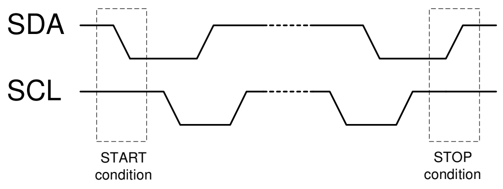


Figure 15: Two Wire Serial Interface - Start / Stop Conditions

The SCL clock is always generated by the master. On every rising transition of the SCL line the data on SDA is valid. Data on SDA line is only allowed to change as long as SCL is low (see figure below).

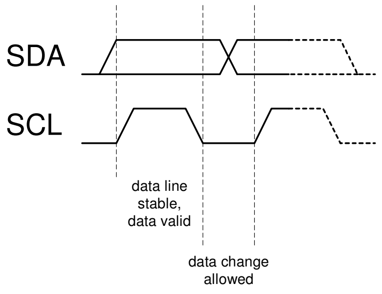


Figure 16: Two Wire Serial Interface - Bit transfer

Copyright © 2004-2011 TRINAMIC Motion Control GmbH & Co. KG