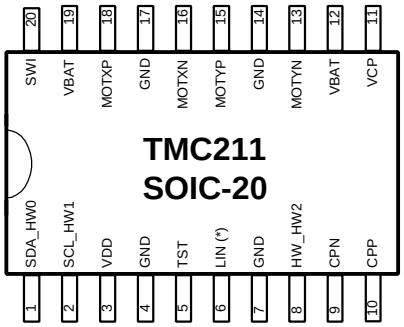
TMC211 – DATASHEET

Micro Stepping Stepper Motor Controller / Driver with LIN Interface



TRINAMIC® Microchips GmbH
Deelbögenkamp 4c
D – 22297 Hamburg
GERMANY

P +49 - (0) 40 - 51 48 06 - 0
F +49 - (0) 40 - 51 48 06 - 60
www.trinamic.com
info@trinamic.com

1 Features

The TMC211 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 TMC211 allows up to four bit of microstepping 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. The TMC211 provides two different interfaces: LIN and a two wire serial interface. Communications to a host takes place either via LIN interface or via two wire serial interface. Together with an inexpensive microcontroller the TMC211 forms a complete motion control system. The main benefits of the TMC211 are:

- **Motor driver**
  - Controls one stepper motor with four bit microstepping
  - 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

- **LIN interface**
  - Physical and Data-Link Layers conform to LIN rev. 1.2
  - Field-programmable node addresses (128)
  - Dynamically allocated identifiers
  - Diagnostics and status information as well as motion parameters accessable
  - LIN bus short-circuit protection to supply and ground and Lost LIN safe operation
  - Lost LIN safe operation

Life support policy

TRINAMIC Microchips GmbH does not authorize or warrant any of its products for use in life support systems, without the specific written consent of TRINAMIC Microchips GmbH.

Life support systems are equipment intended to support or sustain life, and whose failure to perform, when properly used in accordance with instructions provided, can be reasonably expected to result in personal injury or death.

Table of contents

List of figures

List of tables

Block Diagramm

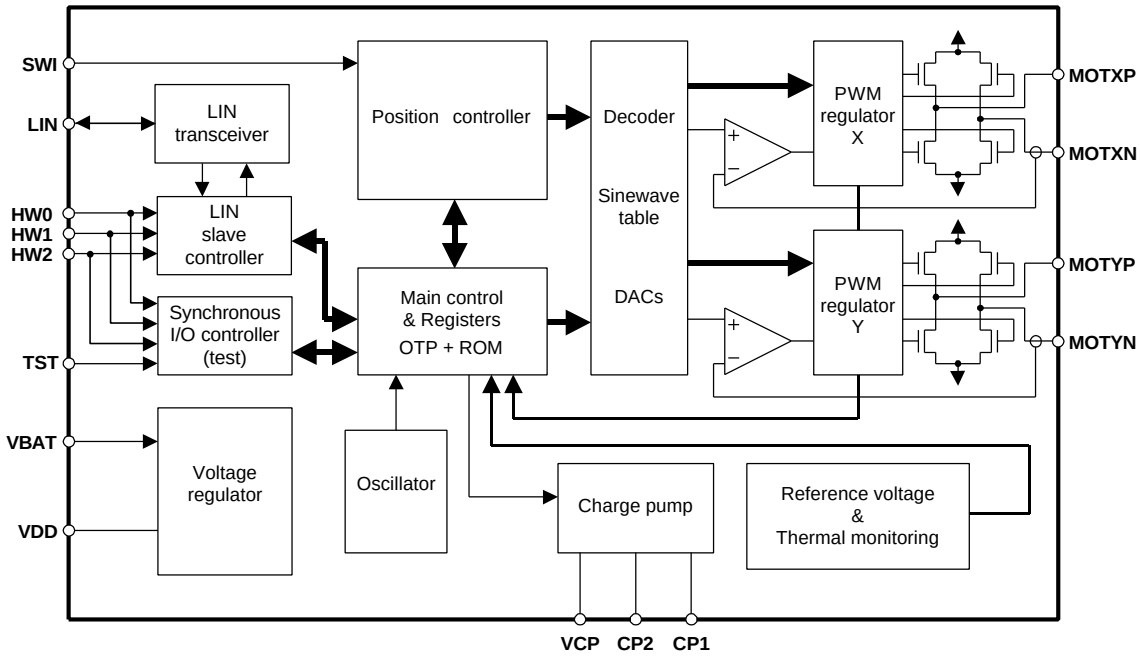


Figure 1: Block diagram

2 General Description

2.1 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 accessable 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.

2.2 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.

2.3 LIN Interface

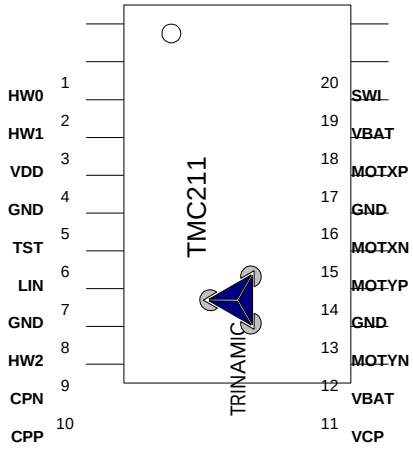
Communication between a host and the TMC211 takes places via the bi-directional LIN interface. Motion Instructions and diagnostic information are provided to or from the Main Control block It is possible to connect up to 128 devices on the same bus. Slave addresses are programmable via OTP memory or external pins. The LIN interface implements the MAC and LLC layers according to the OSI reference model.

2.4 Miscellaneous

Beside the main blocks the TMC211 contains the following:

- an internal charge pump is used to drive the high side transistors.
- an internal oscillator running at 4 MHz +/- 10% to clock the LIN protocol handler, 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.

2.5 Pinning (preliminary)



Pin	SOIC20	QFN32	Description
HW0	1		hard-wire programmable LIN address bit #0 named HW0
HW1	2		hard-wire programmable LIN address bit #1 named HW1
VDD	3		Internal supply (needs external decoupling capacitor)
GND	4,7,14,17		ground, heat sink
TST	5		test pin (to be tied to ground in normal operation)
LIN (*)	6		LIN-bus connection
HW2	8		hard-wire programmable LIN address bit #2 named HW2
CPN	9		negative connection of charge pump capacitor
CPP	10		positive connection of charge pump capacitor
VCP	11		charge pump filter capacitor connection
VBAT	12, 19		battery voltage supply
MOTYN	13		negative end of phase Y coil
MOTYP	15		positive end of phase Y coil
MOTXN	16		negative end of phase X coil
MOTXP	18		positive end of phase X coil
SWI	20		reference switch input

Table 1: TMC211 (*preliminary*) Pin Out

3 Application Environment – Configurations

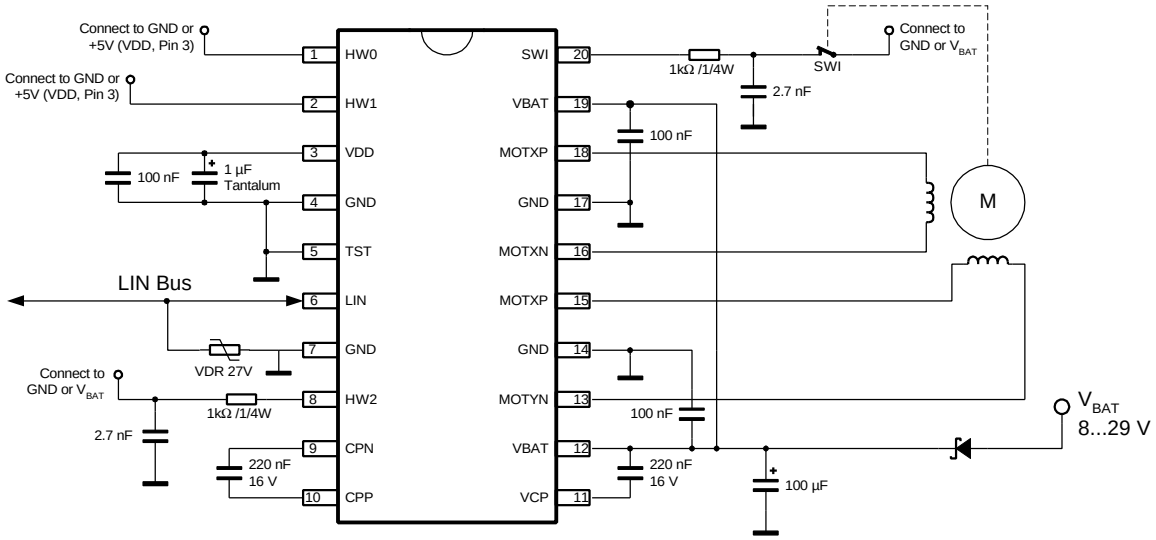


Figure 2: Application Environnent

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
TMC211-S20	SOIC-20	800mA	-40°C..125°C
TMC211-Q28	QFN-28	800mA	-40°C..125°C

Table 2: Ordering Information

4.1 Package Thermal Resistance

4.1.1 SOIC-20 Package

The junction case thermal resistance is 28°C/W, leading to a junction ambient thermal resistance of 63°C/W, with the PCB ground plane layout condition given of the figure beside and with

- PCB thickness = 1.6mm
- 1 layer
- Copper thickness = 35µm

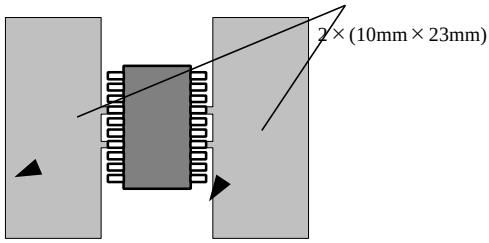


Figure 3: Layout consideration

4.1.2 QFN-32 Package

To be documented when available.

5 Pin-Out

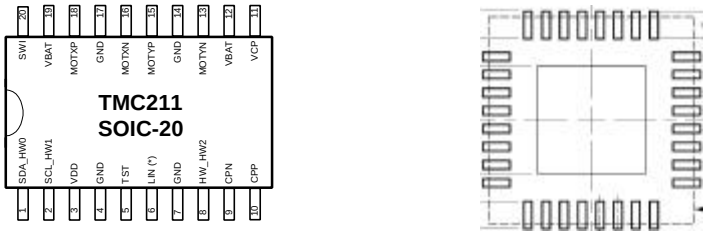


Figure 4: Pin-Out SOIC-20 Package (left side) and QFN-32 Package (right side)

6 Functional Description

6.1 Position Controller and Main Controller

6.1.1 Stepping Modes

The TMC211 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 6.3 Stepper Motor Driver on page 24.

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

6.1.2 Velocity Ramp

A common velocity ramp where a motor drives to a desired position is shown in Figure 5: Typical Velocity Ramp on page 11. The motion consists of a acceleration phase, a phase of constant speed and at least a deceleration phase. Both the acceleration and the deceleration are symmetrical. The acceleration factor can be chosen from a table with 16 entries. (Table 6: Acc Parameters on page 14).
A typical movement 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 Parameters on page 13 and Table 5: Vmin Parameters on page 14).
The user has to take into account that Vmin is not allowed to change while a motion is moving. Vmax is only allowed to change under special circumstances. (See 6.1.4 Vmax Parameters on page 13).
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 distinguish between the run current Irun and 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 are to be found in 6.3.2 Transition Irun to Ihold on page 26. The tables with the current setting for Irun and Ihold are residing at Table 14: Irun Settings on page 26 and Table 15: IHold Settings on page 27. More details about how switching from Irun to Ihold is done can be found in chapter 6.3.2 Transition Irun to Ihold on page 26.
Velocity resp. acceleration parameters are accessable via the LIN interface. These parameters are written via the SetMotorParam command (See Page 42) and read via the GetFullStatus1 command (See Page 36).

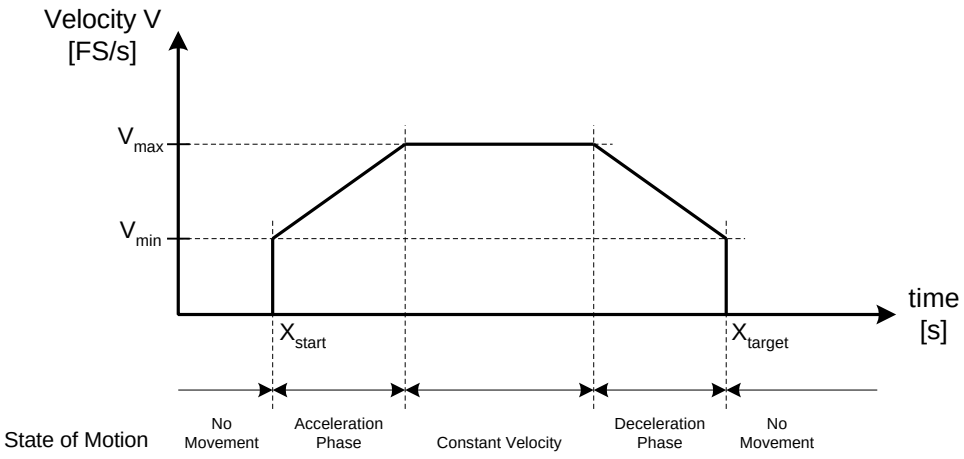


Figure 5: Typical Velocity Ramp

6.1.3 Examples for different Velocity Ramps

The following figures are showing some examples of typical movements under different conditions:

6.1.3.1 Motion with change of target position

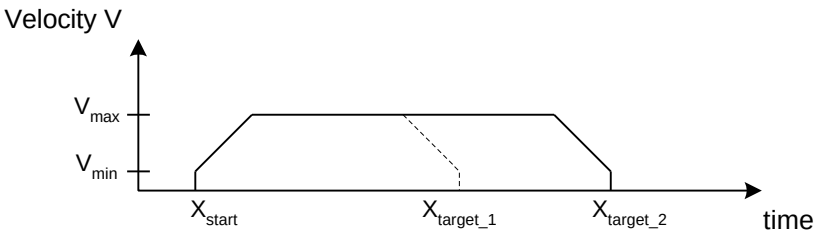


Figure 6: Motion with change of target position

6.1.3.2 Motion with change of target position while in deceleration phase

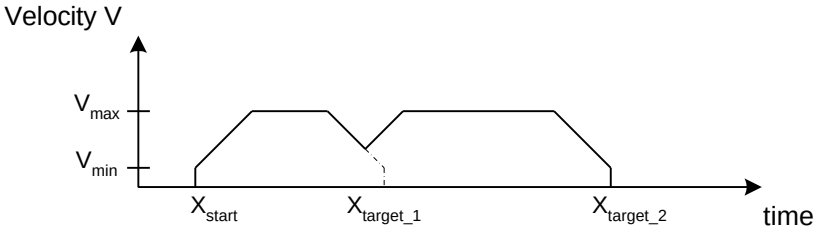


Figure 7: Motion with change of target position while in deceleration phase

6.1.3.3 Short Motion Vmax is not reached

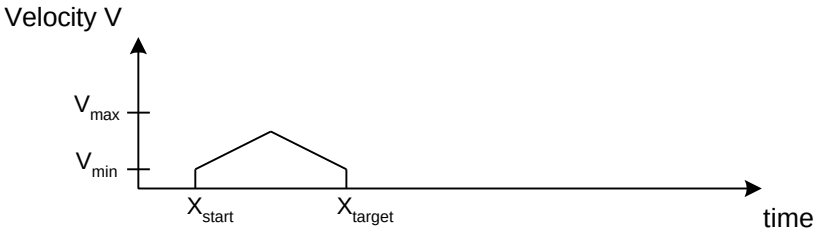


Figure 8: Short Motion Vmax is not reached

6.1.3.4 Motion with change of target position in opposite direction

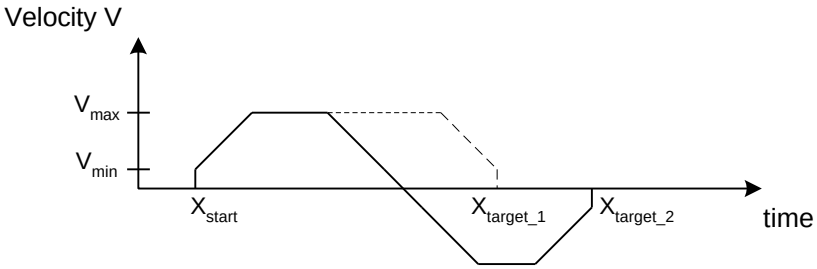


Figure 9: Linear Zero crossing

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.

6.1.4 Vmax Parameters

The desired velocity Vmax can be chosen from the corresponding table “Table 3: Vmax Parameters” which contains 16 different settings for Vmax.

Vmax index	Vmax [FS/s]	Stepping Mode			
		Half-Step Mode [half-steps/s]	1/4 micro stepping [micro-steps/s]	1/8microstepping [micro-steps/s]	1/16 micro stepping [micro-steps/s]
0	99	197	395	790	1579
1	136	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	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	1091	2182	4364	8728
14	729	1457	2914	5829	11658
15	973	1945	3891	7782	15564

Table 3: Vmax Parameters

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 (Table 4: Vmax Groups, Page 13). When changing Vmax during a motion take care that the new Vmax parameters is within the same group. Background: The TMC211 uses a internal pre-divider for positioning calculations. Within one group the pre-divider is equal. When changing Vmax from different groups during motion, correct positioning is not ensured anymore

Group	A	B	C	D
Vmax Index	0	1, 2, 3, 4, 5, 6	7, 8, 9, 10, 11, 12	13, 14, 15

Table 4: Vmax Groups

6.1.5 Vmin Parameters

The 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 next table “Table 5: Vmin Parameters” shows the possible values.

Vmin index	Vmax factor	Vmax [Full-step/s]															
		106	**137**	**167**	**198**	**213**	**228**	**243**	**274**	**304**	**334**	**365**	**395**	**456**	**547**	**730**	**973**
0	1	106	137	167	198	213	228	243	274	304	334	365	395	456	547	730	973
1	1/32	3	4	5	6	7	7	8	9	10	10	11	12	14	17	23	30
2	2/32	7	9	10	12	13	14	15	17	19	21	23	25	29	34	46	61
3	3/32	10	13	16	19	20	21	23	26	29	31	34	37	43	51	68	91
4	4/32	13	17	21	25	27	29	30	34	38	42	46	49	57	68	91	122
5	5/32	17	21	26	31	33	36	38	43	48	52	57	62	71	85	114	152
6	6/32	20	26	31	37	40	43	46	51	57	63	68	74	86	103	137	182
7	7/32	23	30	37	43	47	50	53	60	67	73	80	86	100	120	160	213
8	8/32	27	34	42	50	53	57	61	69	76	84	91	99	114	137	183	243
9	9/32	30	39	47	56	60	64	68	77	86	94	103	111	128	154	205	274
10	10/32	33	43	52	62	67	71	76	86	95	104	114	123	143	171	228	304
11	11/32	36	47	57	68	73	78	84	94	105	115	125	136	157	188	251	334
12	12/32	40	51	63	74	80	86	91	103	114	125	137	148	171	205	274	365
13	13/32	43	56	68	80	87	93	99	111	124	136	148	160	185	211	297	395
14	14/32	46	60	73	87	93	100	106	120	133	146	160	173	200	239	319	426
15	15/32	50	64	78	93	100	107	114	128	143	157	171	185	214	256	342	456

Table 5: Vmin Parameters

6.1.6 Acceleration Parameters

The acceleration parameter can be chosen from a wide range of available values as described in the next table “Table 6: Acc Parameters”. Please take into account that the acceleration parameter is not to change while a motion is ongoing.

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

Table 6: Acc Parameters

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

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

Equation 1

6.1.7 Position Ranges

Position information is coded by using two's complement format. Dependent on the stepping mode (See 6.1.1) the position range will be pictured as 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 7: Position Ranges

Target Positions can be programmed via LIN interface by using the SetPosition command (7.1.11.11 SetPosition on page 41). The actual motor position can be read by GetFullStatus2 command (7.1.11.2 Get on page 37).

6.1.8 Secure Position

In case of emergency (communication loss) or GotoSecPos command (7.1.11.5 GotoSecurePosition on page 39) the motor drives to the secure position. 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 ($1/4^{th}$)
1/8 micro stepping	16 micro steps ($1/8^{th}$)
1/16 micro stepping	32 micro steps ($1/16^{th}$)

Table 8: Secure Position Resolution

6.1.9 External Switch

Pin SWI (and pin HW2 (7.1.4 LIN - Physical Address of the circuit on page 29)) will alternatively attempt to source and sink current in/from the external switch (pin) (Figure 2: Application Environnent on page 9). 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 one.

6.1.10 Motor Shutdown Management

The TMC211 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 6.1.13 Temperature Management on Page 18

- The battery voltage drops below UV2 See 6.1.14 Battery Voltage Management on Page 19.
- 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 LIN interface resp. two-wire-serial-interface remains active during motor shutdown. To leave the motor shutdown state the following conditions must be true:

- Conditions which leads to a motor shutdown are not active anymore
- A GetFullStatus 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 TMC211 is ready to execute any positioning command.

IMPORTANT NOTE:
First, a GetFullStatus command has to be executed after power-on to activate the TMC211.

6.1.11 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 due in case of a problem happened. 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. (7.1.11.10 RunInit command on page 41)

A reference drive consists of two motions (Figure 10: 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. The maximum velocity of the second motion equals to Vmin. After second motion the actual position register is set to zero.

Once the RunInit command is started it can not be interrupted by any other command. Except a condition which leads to a motor shutdown (See 6.1.10 Motor Shutdown Management) happens or a HardStop command is received.
Furthermore the master has to check that the actual position of the stepper motor corresponds **not** to the target position of the first motion. 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 GetFullStatus command will cause the circuit to leave the deadlock state.

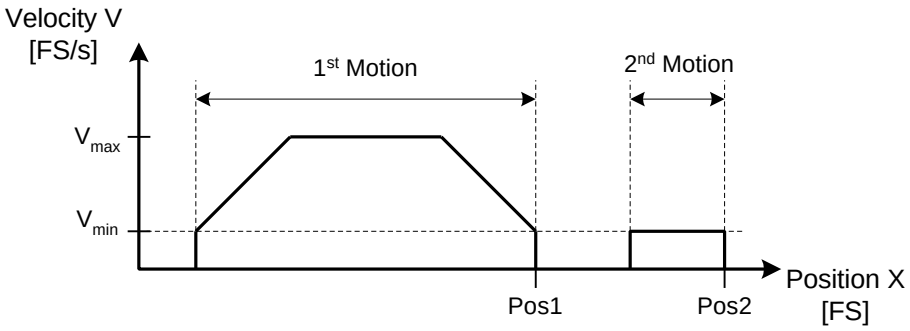


Figure 10: RunInit

6.1.12 Sleep Mode

When entering in Sleep mode, the stepper-motor can be driven to its secure position. After which, the circuit is completely powered down, apart from the LIN receiver which remains active to detect dominant state on the bus. In case sleep mode is entered while a motion is ongoing, a transition will occur towards secure position as described

Sleep mode can be entered in the following cases:

- The circuit receives a LIN frame with identifier 0x3C and first Data byte containing 0x00, as required by LIN specification rev 1.2.
- The LIN bus remains inactive (or is lost) during more than 25000 time slots (1.30s at 19.2 kbit/s), after which a time-out signal switches the circuit to sleep mode.

The circuit will return to normal mode if a valid LIN frame is received while entering the Sleep mode (this valid frame can be addressed to another slave). For more information refer to 7.1.11.15 Sleep Mode on page 45.

6.1.13 Temperature Management

The TMC211 has got an internal temperature monitoring. The circuit goes into shutdown mode if the temperature exceed threshold T_{tsd}, furthermore two thresholds are implemented to generate a temperature pre-warning. Figure 11: Temperature Management on page 18.

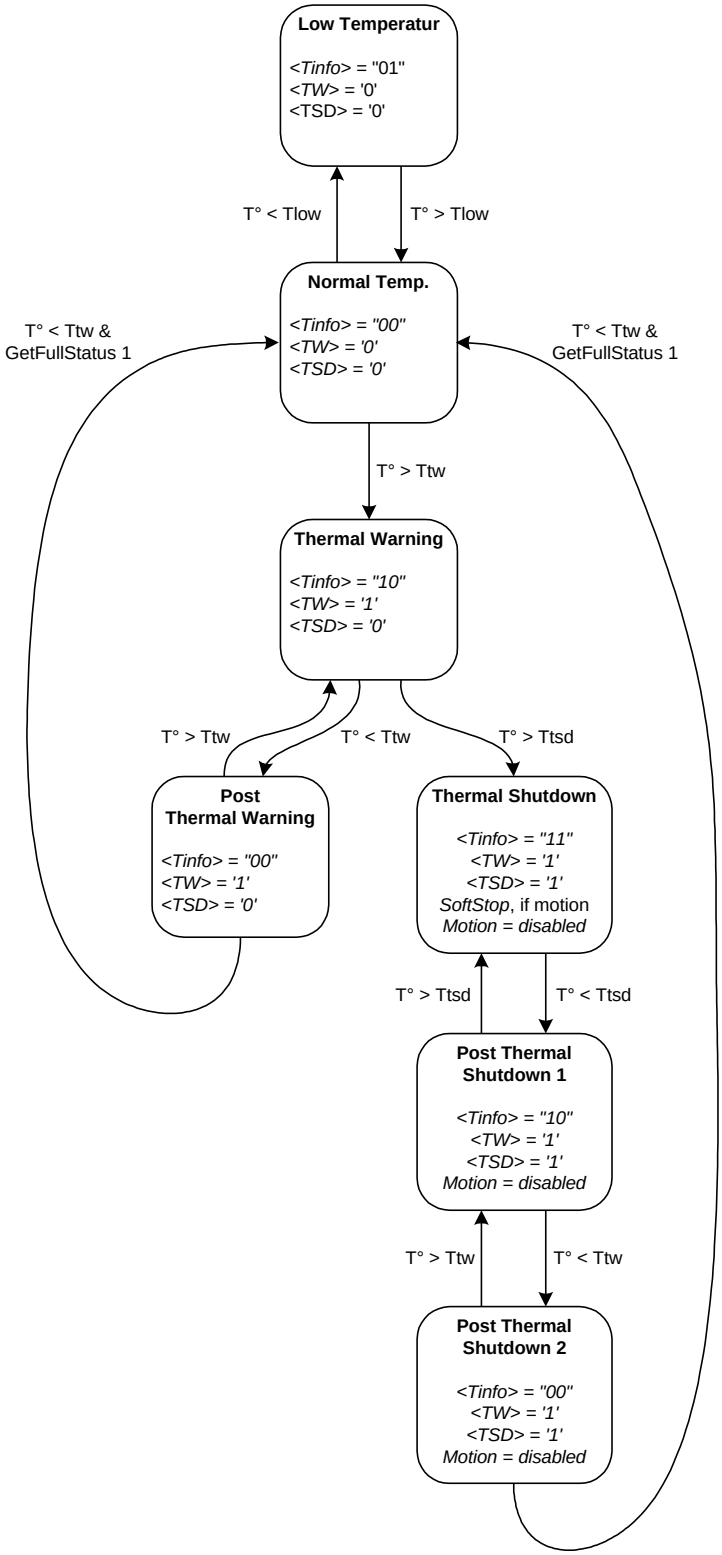


Figure 11: Temperature Management

6.1.14 Battery Voltage Management

The TMC211 has got 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. Figure 12: Battery Voltage Management on page 19.

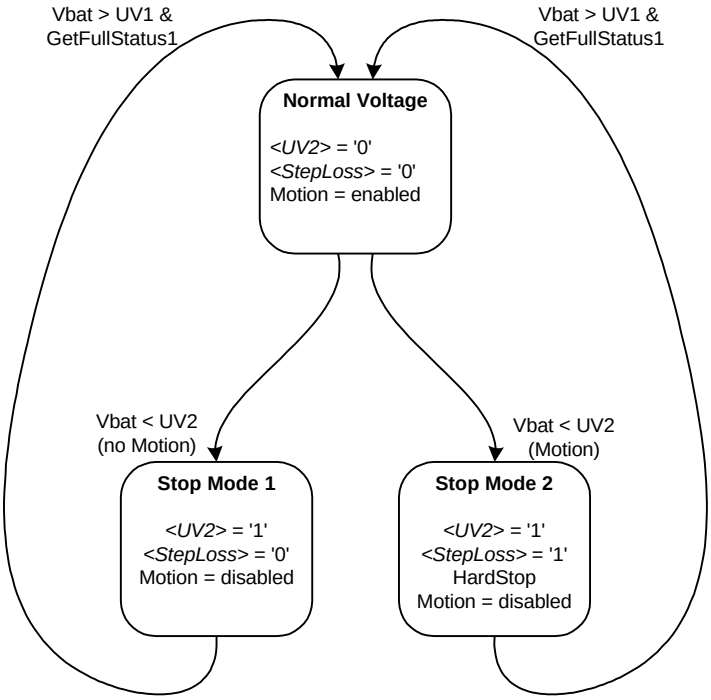


Figure 12: Battery Voltage Management

6.1.15 Internal handling of commands and flags

Due to the sleep mode, the internal handling of commands and flags differs. Commands are handled with different priorities dependent on the current state and the current status of internal flags. Concerning the LIN interface see Figure 13: Internal handling of commands and flags on page 20, a detailed table on this issue can be found at on page 22.

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 chapter 7.1.11 LIN - Command Description on page 36 - e.g.: When the circuit drives the motor to its programmed target position, state “GotoPos” is entered. There are three possibilities / commands to leave this state: HardStop, SoftStop or TargetPosition reached. The HardStop command has got the highest priority and will overwrite the alternatives possibilities. Motion finished (target position reached) has got the lowest priority.

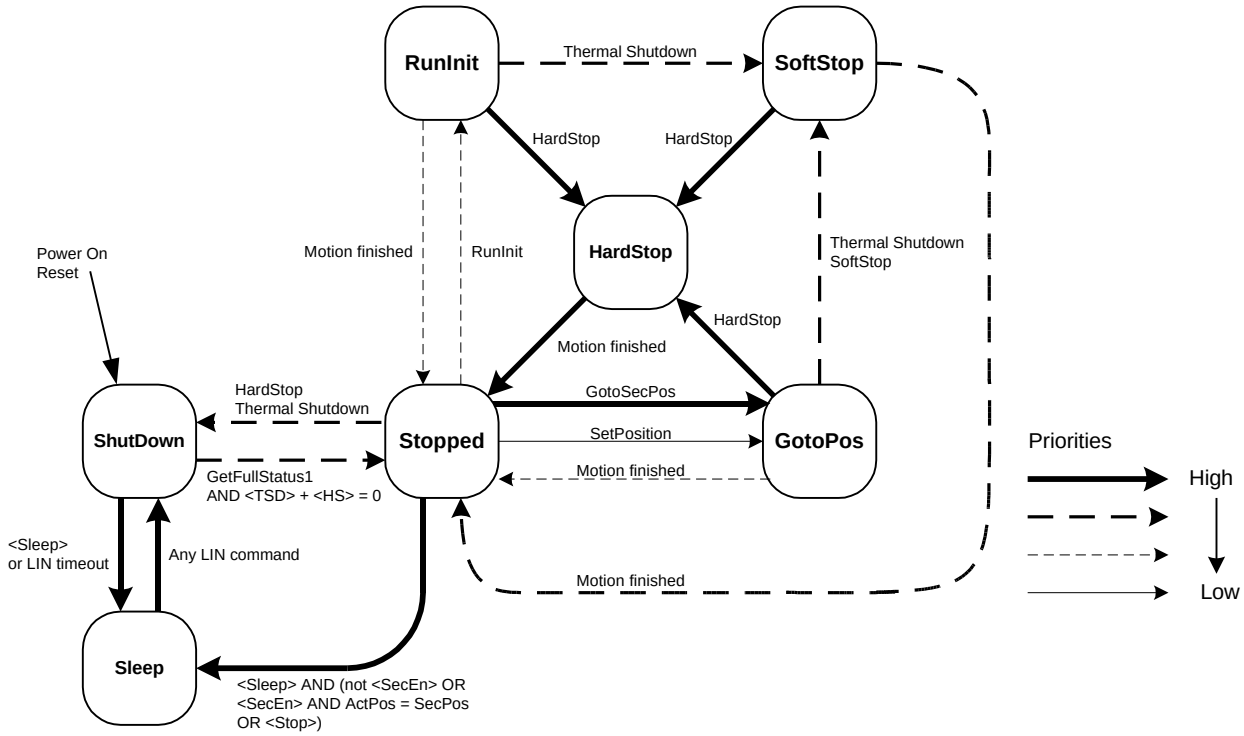


Figure 13: Internal handling of commands and flags

State ® / Command	Stopped	GotoPos	RunInit	SoftStop	HardStop	ShutDown	Sleep
	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	no power (note 1)
GetActualPos	LIN in-frame response	LIN in-frame response	LIN in-frame response	LIN in-frame response	LIN in-frame response	LIN in-frame response	
GetOTPparam	OTP refresh; LIN in-frame	OTP refresh; LIN in-frame	OTP refresh; LIN in-frame	OTP refresh; LIN in-frame	OTP refresh; LIN in-frame	OTP refresh; LIN in-frame	
GetFullStatus or GetStatus [attempt to clear <TSD> and <HS> flags]	LIN in-frame response	LIN in-frame response	LIN in-frame response	LIN in-frame response	LIN in-frame response	LIN 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 3)	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				
SetPositionShort [half-step mode only)]	TagPos updated; ® **GotoPos**	TagPos updated	TagPos updated				
GotoSecPosition	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**					
Sleep or LIN timeout [⇒ <Sleep> = '1', reset by any LIN command received later]	See note 9	If <SecEn> = '1' then TagPos = SecPos else → **SoftStop**	If <SecEn> = '1' then TagPos = SecPos; will be evaluated after RunInit	No action; <Sleep> flag will be evaluated when motor stops	No action; <Sleep> flag will be evaluated when motor stops	® **Sleep**	
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.	n.a.

With the following color code:

Command ignored

Transition to another state

Master is responsible for proper update (see note 7)

Notes:

1 Leaving Sleep state is equivalent to Power on reset.

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

3 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.

4 The <Sleep> flag is set to '1' when a LIN timeout or a Sleep command occurs. It is reset by the next LIN command (<Sleep> is cancelled if not activated yet).

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

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

7 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 or SetPositionShort command.

8 Some transitions like GotoPos → Sleep are actually done via several states: GotoPos → SoftStop → Stopped → Sleep (see diagram below).

9 Two transitions are possible from state Stopped when <Sleep> = '1':

1) Transition to state Sleep if (<SecEn> = '0') or ((<SecEn> = '1') and (ActPos = SecPos)) or <Stop> = '1'
2) Otherwise transition to state GotoPos, with TagPos = SecPos

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

11 <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.

12 Command for dynamic assignment of IDs is decoded in all states except Sleep and has not effect on the current state

13

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

15 If <StepLoss> is active, then SetPosition, SetPositionShort and GotoSecurePosition commands are ignored (they will not modify TagPos register whatever the state), and motion to secure position is forbidden after a Sleep command or a LIN timeout (the circuit will go into Sleep state immediately, without positioning to secure position). Other command like RunInit or ResetPosition will be executed if allowed by current state. <StepLoss> can only be cleared by a GetStatus or GetFullStatus command.

Table 9: Priority Encoder (ref. Figure 13 , page 20)

6.2 RAM and OTP Memory

6.2.1 RAM Registers

Register	Mnemonic	Length (bit)	Related commands	Comment	Reset State
Actual Position	ActPos	16	GetFullStatus 2 ResetPosition	Actual Position of the Stepper Motor. 16-bit signed	0x0000
Target Position	TagPos	16	SetPosition GetFullStatus 2 ResetPosition	Target Position of the Stepper Motor. 16-bit signed	
Acceleration Shape	AccShape	1	GetFullStatus 1 SetMotorParam ResetToDefault	0 = Acceleration with Acc Parameter. 1 = Velocity set to Vmin, without acceleration	
Coil Peak Current	Irun	4	GetFullStatus 1 SetMotorParam ResetToDefault	Coil current when motion is ongoing (Table 14: Irun Settings)	OTP Memory
Coil Hold Current	Ihold	4	GetFullStatus 1 SetMotorParam ResetToDefault	Coil current when motor stands still (Table 15: IHold Settings)	
Minimum Velocity	Vmin	4	GetFullStatus 1 SetMotorParam ResetToDefault	Start Velocity of the stepper motor (Table 5: Vmin Parameters)	
Maximum Velocity	Vmax	4	GetFullStatus 1 SetMotorParam ResetToDefault	Target Velocity of the stepper motor (Table 3: Vmax Parameters)	
Shaft	Shaft	1	GetFullStatus 1 SetMotorParam ResetToDefault	Direction of movement	
Acceleration / Deceleration	Acc	4	GetFullStatus 1 SetMotorParam ResetToDefault	Parameter for acceleration (Table 6: Acc Parameters)	
Secure Position	SecPos	11	GetFullStatus 2 ResetToDefault	Target Position for GotoSecurePosition command (7.1.11.5 GotoSecurePosition) or when LIN connection fails; 11 MSBs of 16-bit position (LSBs fixed to '0')	
Stepping Mode	StepMode	2	GetFullStatus 1 GetFullStatus 2 ResetToDefault	Micro stepping mode (6.1.1 Stepping Modes)	

Table 10: RAM Registers

6.2.2 OTP Memory Structure

The table below shows how the parameters to be stored in the OTP memory are located.

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 52: DC Parameters Supply and Voltage regulator on page 50), 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	EnableLIN	TSD2	TSD1	TSD0	BG3	BG2	BG1	BG0
0x02	ADM	N.A.	N.A.	N.A.	AD3	AD2	AD1	AD0
0x03	Irun3	Irun2	Irun1	Irun0	Ihold3	Ihold2	Ihold1	Ihold0
0x04	Vmax3	Vmax2	Vmax1	Vmax0	Vmin3	Vmin2	Vmin1	Vmin0
0x05	N.A.	StepMode1	StepMode0	Shaft	Acc3	Acc2	Acc1	Acc0
0x06	SecPos7	SecPos6	SecPos5	SecPos4	SecPos3	SecPos2	SecPos1	SecPos0
0x07	N.A.	N.A.	N.A.	SecPos10	SecPos9	SecPos8	LOCKBT	LOCKBG

Table 11: 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.
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 12: OTP Lock bits

The command used to load the application parameters via the LIN bus in 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.

6.3 Stepper Motor Driver

The StepMode parameter in SetMotorParam command (7.1.11.12 SetMotorParameter on page 42) are 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 13: StepMode

6.3.1 Coil current shapes

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

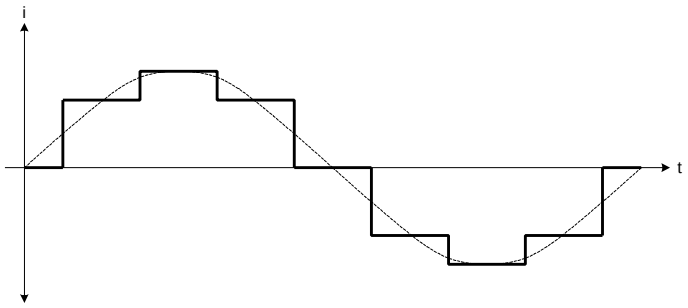


Figure 14: Coil Current for Half Stepping Mode

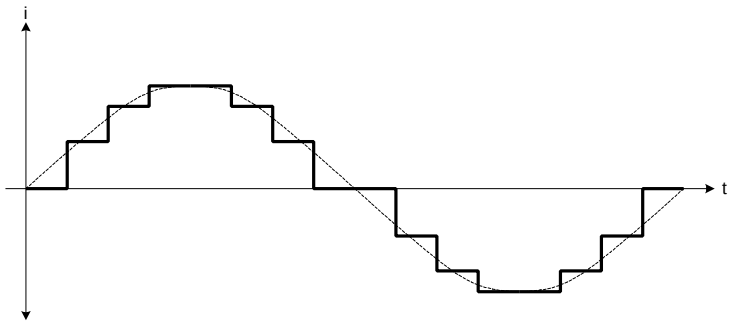


Figure 15: Coil Current for 1/4 Micro Stepping Mode

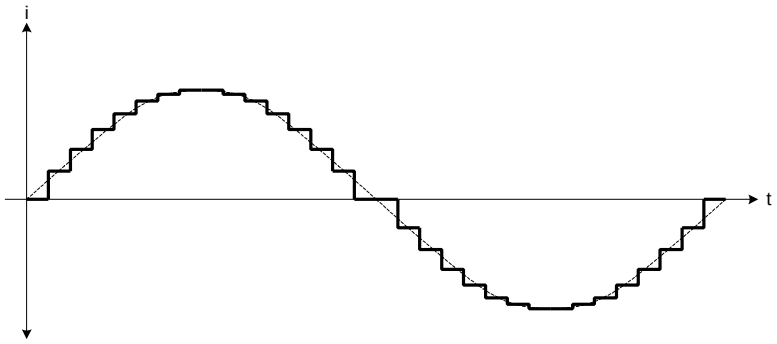


Figure 16: Coil Current for 1/8 Micro Stepping Mode

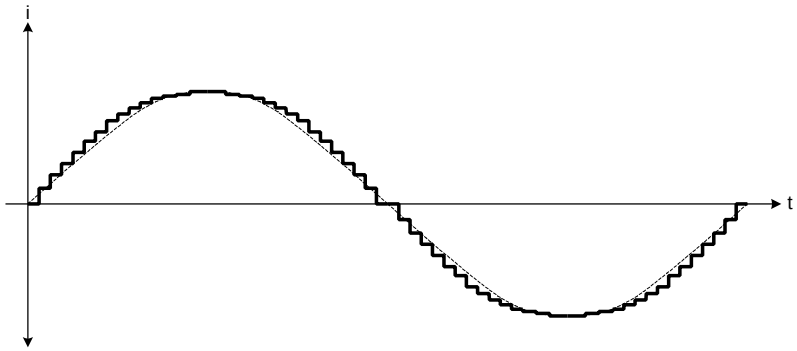


Figure 17: Coil Current for 1/16 Micro Stepping Mode

6.3.2 Transition Irun to Ihold

At the end of an motor motion the actual coil currents "I_{run}" 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 "I_{hold}". The next Figure 18: Transition Irun to Ihold illustrates the mechanism.

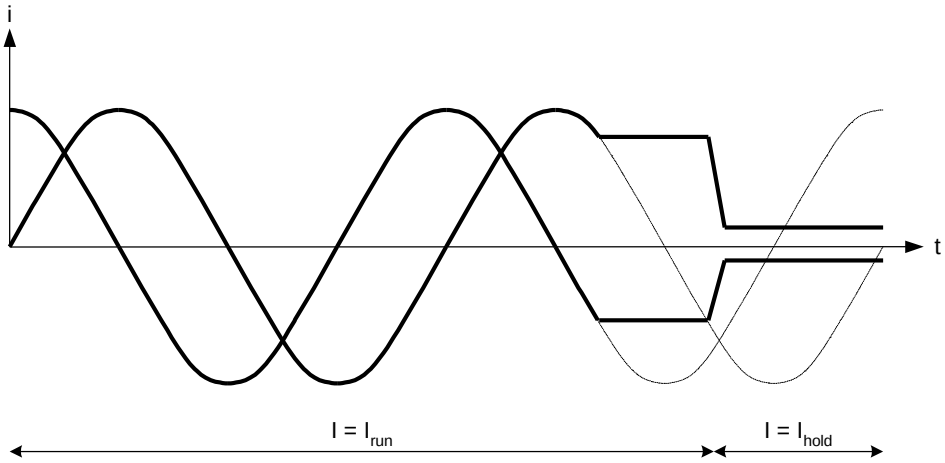


Figure 18: Transition Irun to Ihold

Both currents Irun and Ihold are parameterizeable via the corresponding serial datagram. 16 values are available for Irun current and 16 values for Ihold current. The next tables Table 14 and Table 15 show the corresponding current values.

Irun					Peak Current [mA]
Hexadecimal	Binary				
0	0	0	0	0	59
1	0	0	0	1	71
2	0	0	1	0	84
3	0	0	1	1	100
4	0	1	0	0	119
5	0	1	0	1	141
6	0	1	1	0	168
7	0	1	1	1	200
8	1	0	0	0	238
9	1	0	0	1	283
A	1	0	1	0	336
B	1	0	1	1	400
C	1	1	0	0	476
D	1	1	0	1	566
E	1	1	1	0	673
F	1	1	1	1	800

Table 14: Irun Settings

Ihold					Peak Current [mA]
Hexadecimal	Binary				
0	0	0	0	0	59
1	0	0	0	1	71
2	0	0	1	0	84
3	0	0	1	1	100
4	0	1	0	0	119
5	0	1	0	1	141
6	0	1	1	0	168
7	0	1	1	1	200
8	1	0	0	0	238
9	1	0	0	1	283
A	1	0	1	0	336
B	1	0	1	1	400
C	1	1	0	0	476
D	1	1	0	1	566
E	1	1	1	0	673
F	1	1	1	1	800

Table 15: IHold Settings

6.3.3 Chopper Mechanism

The chopper frequency is fixed to the frequency as specified in chapter 9.4 AC Parameters on page 50. The TMC211 uses an intelligent chopper algorithm to provide a smooth operation with low resonance. The TMC211 uses internal measurements to derive current flowing through coils.
If the current is less than the desired current, the TMC211 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. Figure 19: Different Chopper Cycles with Fast and Slow Decay on page 27 shows the chopper behavior.

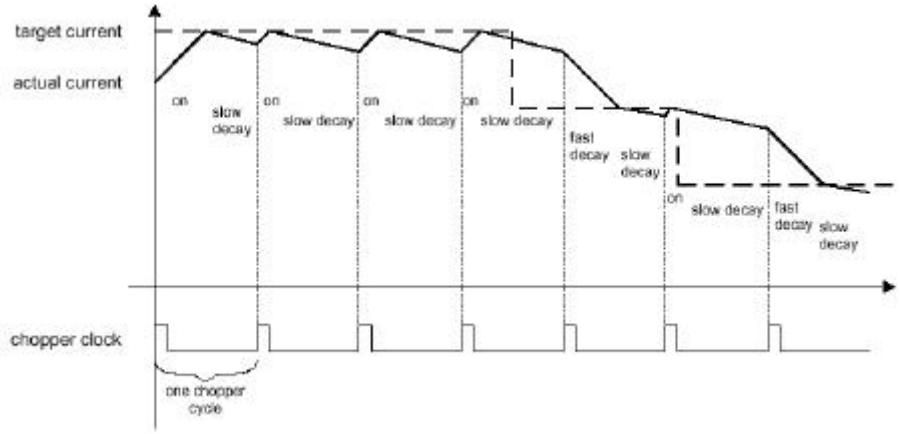


Figure 19: Different Chopper Cycles with Fast and Slow Decay

7 LIN Interface

7.1.1 LIN - General Description

The LIN (Local Interconnect Network) is a serial communication protocol that is used mainly for distributed mechatronic systems in automotive applications. The LIN implementation in the TMC211 corresponds to one slave node which follows to LIN rev. 1.2 specification.

Features:

- Single master / multiple-slave communication
- Self synchronizing slave nodes / no quartz or ceramics resonator necessary
- Guaranteed latency times for signal transmission
- Single-wire communication
- Transmission speed up to 20 kbit/s
- Selectable length of Message Frame: 2, 4, and 8 bytes
- Configuration flexibility
- Data checksum security and error detection
- Detection of defective nodes in the network

For more information about the LIN protocol please refer to the official website and the LIN Protocol Specification. Both can be found at http://www.lin-subbus.org/.

7.1.2 LIN - Physical Layer

The physical layer is a single wire with pull-up resistor is every node. The bus is directly powered from the vehicle power net V_{bb}.

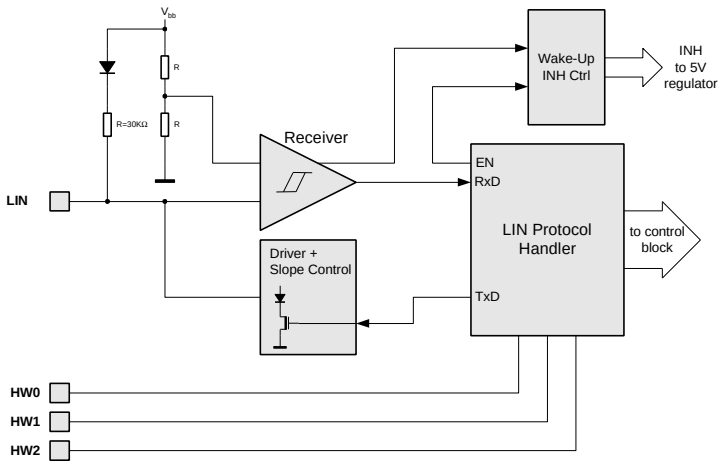


Figure 20: LIN - Physical Layer

7.1.2.1 LIN - Analog Part

The transmitter is a low-side driver with a pull-up resistor and slope control. Figure 21: LIN - Analog Part on page 29 shows the characteristics of the transmitted signal, including the delay between internal TxD Signal and LIN Signal. See Table 59: AC Parameters LIN Transmitter on page 51 and Table 60: AC Parameters LIN Receiver on page 51 for timing values.

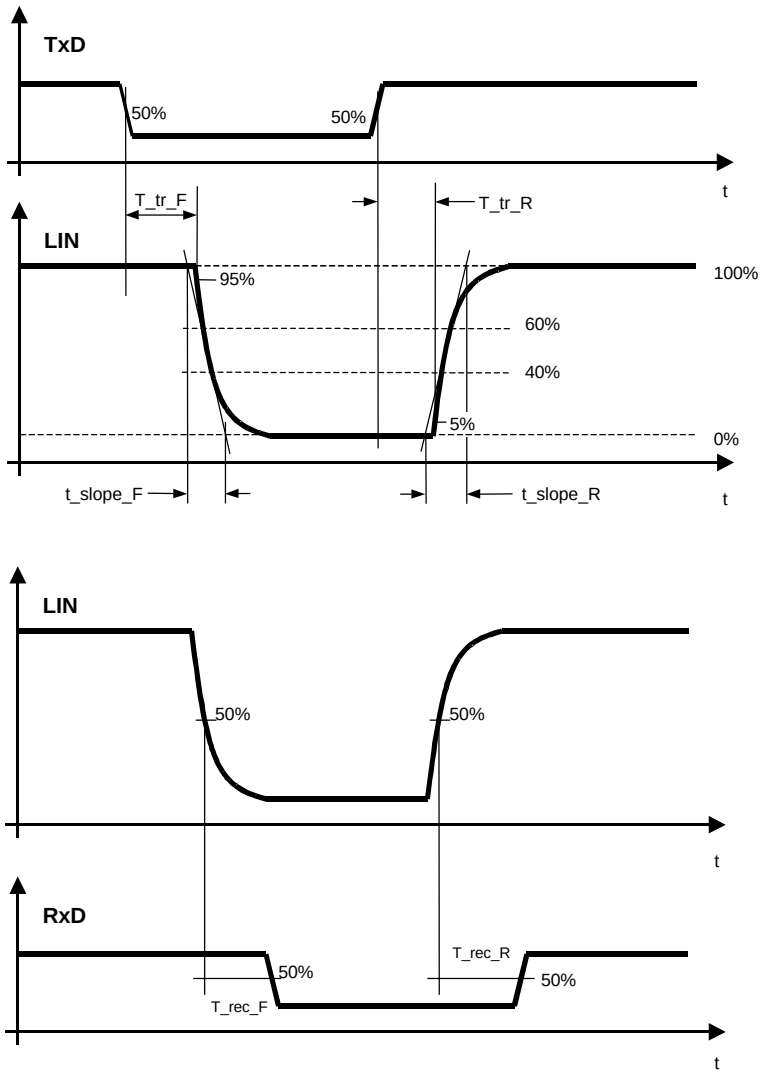


Figure 21: LIN - Analog Part

7.1.3 Slave operational range for proper self synchronization

The internal oscillator having a +/- 10% accuracy over the voltage and temperature range, the LIN interface will synchronize properly in the following conditions:

- Vbb > tbd
- Ground shift between Master node and Slave node < +/- 1 Volt

It is highly recommended to use the same type of reverse battery voltage protection diode for the Master and the Slave nodes.

7.1.4 LIN - Physical Address of the circuit

The circuit must be provided with a physical address in order to discriminate this circuit from other ones on the LIN bus. This address is coded on 7 bits, yielding the theoretical possibility of 128 different circuits on the same bus. It is a combination of 4 OTP memory bits (see 6.2.2 OTP Memory Structure on page 24) and of the 3 hardwired address bits (pins HW[2:0]).

The TCM211 supports broadcasting. When the <Broad> bit is set to zero, broadcasting is active and each slave on the LIN bus will be addressed.

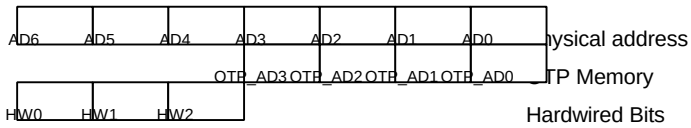


Figure 22: LIN - Physical Slave Address

The amount of physical addresses can be expanded by using bit ADM. This bit allows following expansion:

ADM	AD6	AD5	AD4	AD3	AD2	AD1	AD0
0	HW0	HW1	HW2	PA3	PA2	PA1	PA0
1	PA0	HW0	HW1	HW2	PA3	PA2	PA1

Table 16: LIN - Physical Address Expansion

Note: pins HW0 and HW1 are 5Volts digital inputs, whereas pin HW2 is compliant with a 12Volts level, e.g. it can be connected to Vbat or Gnd via a terminal of the PCB. To provide cleaning current for this terminal, the system used for pin SWI is also implemented for pin HW2 (See 6.1.9 External Switch on page 15)

7.1.5 LIN - Electro Magnetic Compability

EMC behavior fulfills requirements defined by LIN specification, rev.1.2.

7.1.6 LIN - Error Status Register

The LIN interface implements a register containing an error status of the LIN communication. This register is as specified in

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Not used	Not used	Not used	Not used	Timeout Error	Data Error Flag	Header Error Flag	Bit Error Flag

Table 17: LIN - Error Status Register

Note:
Data Error Flag = Checksum error OR StopBit error OR Length error
Header Error Flag = Parity error OR Synch Field error

A GetFullStatus command will reset the error status register

7.1.7 LIN - Dynamic Assignment of Identifiers

The identifier field in the LIN datagram denotes the content of the message. Six identifier bits and two parity bits are used to represent the content. The identifiers 0x3C and 0x3F are reserved for command frames and extended frames. Slave nodes need to be very flexible to adapt itself to a given LIN network in order to avoid conflicts with slave nodes from different manufacturers. Dynamic assignment of the identifiers will fulfill this requirement by writing identifiers into the circuits RAM. ROM pointers are linking commands and dynamic identifiers together.

A writing frame with identifier 0x3C issued by the LIN master will write dynamic identifiers into the RAM. One writing frame is able to assign 4 identifiers, therefore 3 frames are needed to assign all identifiers. Each ROM pointer ROMp_x [3:0] place the corresponding dynamic identifier Dyn_ID_x [5:0] at the correct place in the RAM (see Table 18: LIN - Dynamic Identifiers Writing Frame).

When setting <BROAD> to zero broadcasting is active and each slave on the LIN bus will store the same dynamic identifiers, otherwise only the slave with the corresponding slave address is programmed.

Dynamic Identifiers Writing Frame									
Byte	**Content**	**Structure**							
		Bit7	**Bit6**	**Bit5**	**Bit4**	**Bit3**	**Bit2**	**Bit1**	**Bit0**
0	Identifier	0x3C							
1	AppCMD	0x80							
2	CMD	1	0x11						
3	Address	Broad	AD6	AD5	AD4	AD3	AD2	AD1	AD0
4	Data	DynID_1 [3:0]				ROMp_1 [3:0]			
5	Data	DynID_2 [1:0]		ROMp_2 [3:0]				DynID_1 [5:4]	
6	Data	ROMp_3 [3:0]							
7	Data	ROMp_4 [1:0]		DynID_3 [5:0]					
8	Data	DynID_4 [5:0]						ROMp_4 [3:2]	

Table 18: LIN - Dynamic Identifiers Writing Frame

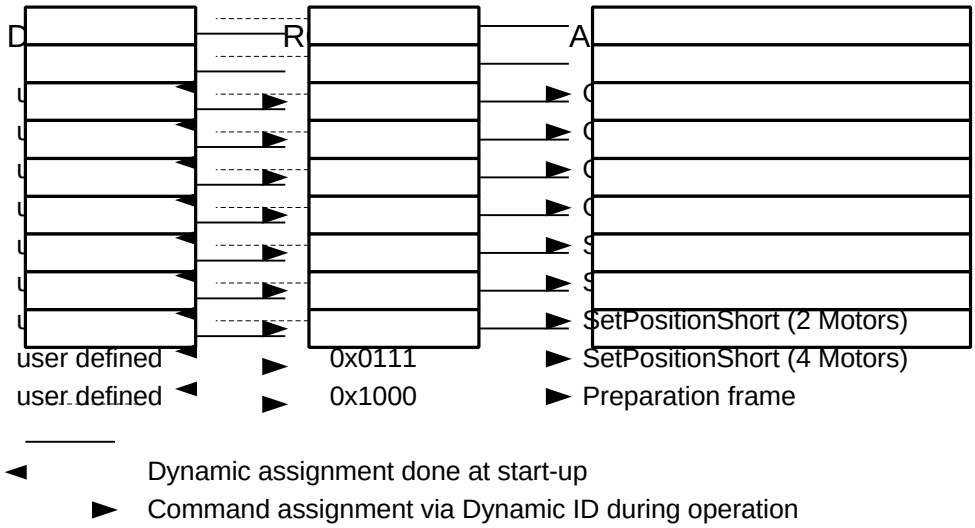


Figure 23: LIN - Principle of dynamic command assignment

Command Mnemonic	Command Byte (CMD)		Dynamic ID	ROM pointer
	binary	hex		
GetActualPos	000000	0x00	100xxx	0010
GetFullStatus	000001	0x01	Not used	
GetOTPParam	000010	0x02	Not used	
GetStatus	000011	0x03	000xxx	0011
GotoSecurePosition	000100	0x04	Not used	
HardStop	000101	0x05	Not used	
ResetPosition	000110	0x06	Not used	
ResetToDefault	000111	0x07	Not used	
RunInit	001000	0x08	Not used	
SetMotorParam	001001	0x09	Not used	
SetOTPParam	010000	0x10	Not used	
SetPosition	001011	0x0B	010xxx	0100
SetPositionShort (1 Motor)	001100	0x0C	001001	0101

SetPositionShort (2 Motors)	001101	0x0D	101001	0110
SetPositionShort (4 Motors)	001110	0x0E	111001	0111
Sleep	Not used		Not used	
SoftStop	001111	0x0F	Not used	
Dynamic ID assignment	010001	0x11	Not used	
General Purpose 2 Data Bytes			011000	0000
General Purpose 4 Data Bytes			101000	0001
Preparation Frame			011010	1000

Table 19: LIN - Commands and Corresponding Dynamic IDs

7.1.8 LIN Frames

7.1.8.1 Overview

As specified in LIN rev. 1.2 specification a LIN frame consists of an 8-bit identifier field, followed by 2, 4 or 8 data field and a checksum field. A LIN frame can either be a writing frame, with one of the following tasks:

- Program the OTP memory
- Provide motion parameters, e.g. velocity, position, torque to the TMC211

Or a LIN frame can be a reading frame which is used to:

- Read actual position or status information of the stepper motor
- Verification of correct programming and configuration

7.1.8.2 LIN - Writing Frames

According to the LIN specification there is only a fixed amount of dynamic identifiers available. In order to expand the amount of identifiers resp. the amount of command different types of writing frames are introduced. The TMC211 supports four different write frames and four different read frames. The following figures will illustrate the differences.

7.1.8.2.1 LIN - Writing Frame Type#1 (2 or 4 bytes)

This type is used to provide 2 or 4 bytes of data to the slave nodes. When <Broad> is set to zero, broadcasting is active and the command is valid for all slave nodes. When <Broad> is one the command is only valid for the slave node which physical address corresponds to the one provided by the writing frame.
A writing frame of 2 bytes issues only a defined command to the slave node(s), e.g. HardStop command. Whereas a writing frame of 4 bytes issues a command and 2 bytes of data to the slave node(s), e.g. SetPosition command.

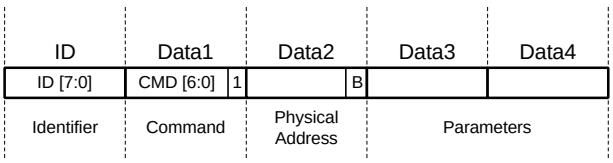


Figure 24: LIN - Writing Frame Type#1

7.1.8.2.2 LIN - Writing Frame Type#2 (2, 4 or 8 bytes)

2, 4 or 8 data bytes writing frame with an identifier dynamically assigned to a particular slave node together with an application command, regardless of the physical address of the circuit. e.g. SetPositionShort command.

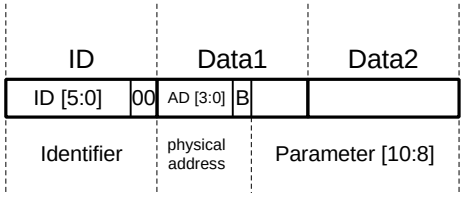


Figure 25: LIN - Writing Frame Type#2

7.1.8.2.3 LIN - Writing Frame Type#3 (2 bytes)

2 data bytes writing frame with an identifier dynamically assigned to a particular slave node together with an application command. This type of frame requires that there are as many dynamically assigned identifiers as there are TMC211 circuits using this command connected to the LIN bus.

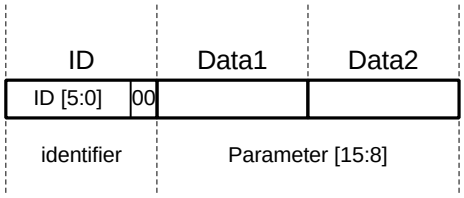


Figure 26: LIN - Writing Frame Type#3

7.1.8.2.4 LIN - Writing Frame Type#4 (8 bytes)

8 data bytes writing frame with 0x3C identifier. The structure is similar to type#1 but uses the reserved identifier 0x3C. Using a reserved identifier followed by a particular application command will expand the amount of possible LIN commands.

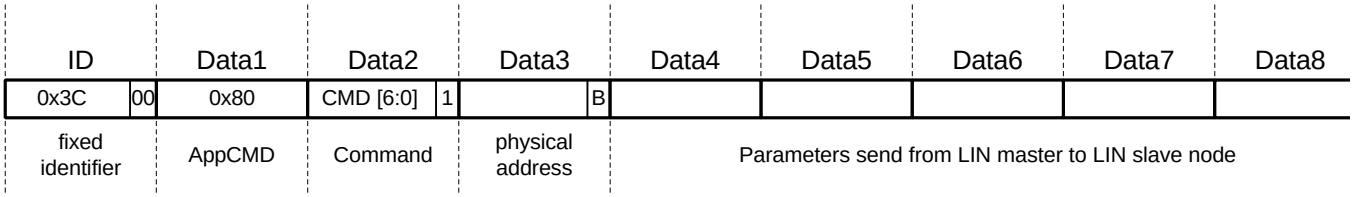


Figure 27: LIN - Writing Frame Type#4

7.1.8.3 LIN - Reading Frames

When using reading frames the master initiates the communication by sending a header field, which contains the synchronization and identifier field. The TMC211 supports two types of identifiers:

- Direct ID: The identifier points to a particular slave node. As describe in the LIN specification the slave sends the data after the in-frame response slave. Direct ID gives the fastest access to the required data.
- Indirect ID: Indirect ID contains of two datagrams. The first datagram, called preparation frame, which issues the slave's physical address to the particular slave node. The second datagram specifies only a reading command by using the reserved identifier 0x3D. Indirect ID has the advantage to use a reserved identifier and therefore provides more flexibility.

7.1.8.3.1 LIN - Reading Frame Type#5 (2, 4 or 8 bytes)

Type#5 is a reading frame, which uses direct ID, therefore the master initiates the communication and the slave responds after receiving the identifier. Dependent on the identifier the slave transmits 2, 4 or 8 bytes of data. GetActualPos command uses Type#5 reading frame.

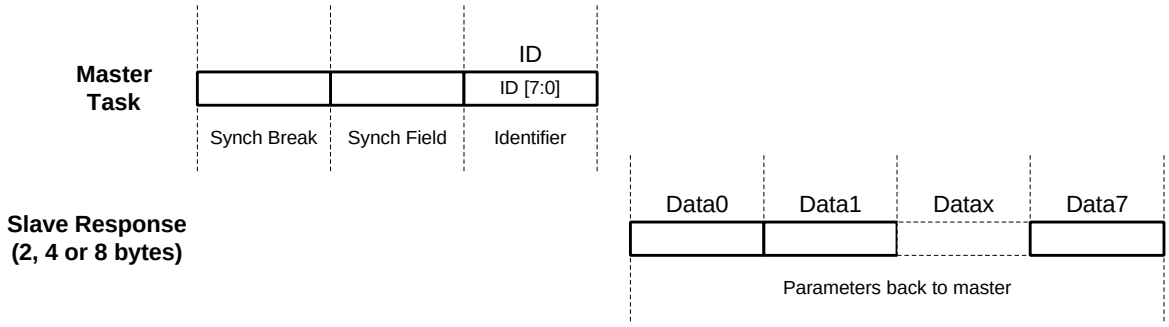


Figure 28: LIN - Reading Frame Type#5

7.1.8.3.2 LIN - Reading Frame Type#6 (8 bytes)

Reading frame Type#6 uses indirect ID and therefore a preparation frame of Type#7 or #8 is needed. The preparing frame dumps the reading command into a particular slave node. Data from the slave is then transmitted after the next reading frame. The reading frame must always be consecutive to a preparing frame, otherwise it is not valid and not taken into account.

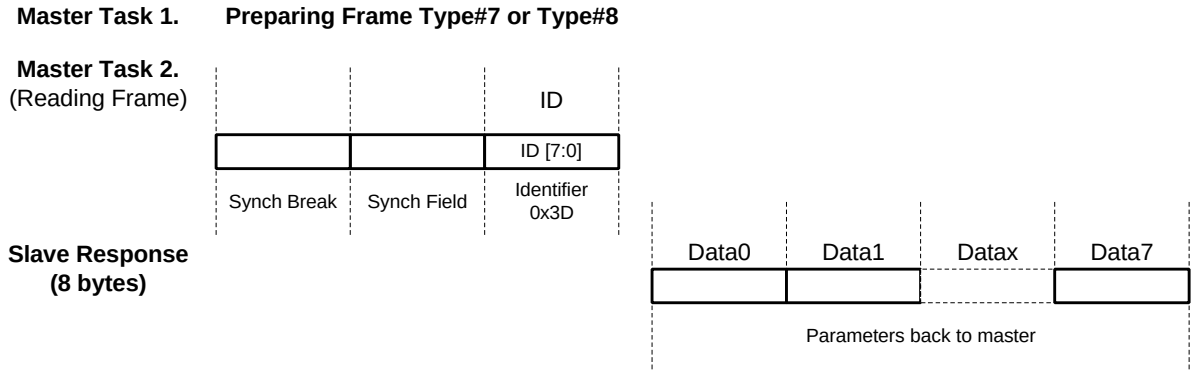


Figure 29: LIN - Reading Frame Type#6

7.1.8.3.3 LIN - Reading Frame Type#7 (Preparing frame)

A preparing frames prepares a particular slave node, that it has to answer after the next reading frame. Preparing frames are needed when using indirect ID. A preparing frame consists of the physical address of the slave and a command indicating which kind of information is to provide to the master.

Type#7 preparing frame consists of a dynamically assigned identifier, the command and the physical address of the slave.

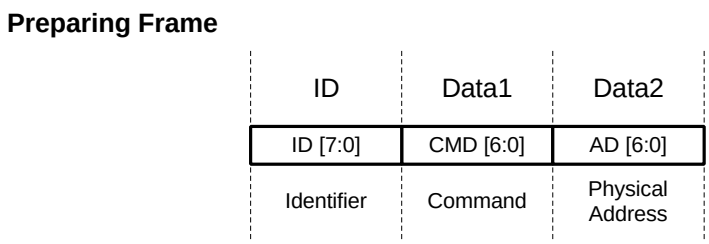


Figure 30: LIN - Preparing Frame Type#7

7.1.8.3.4 LIN - Reading Frame Type#8 (Preparing frame)

Type#8 preparing frame uses the reserved identifier 0x3C, followed by the application command 0x80, then the particular reading command and the physical address is provided by the slave.

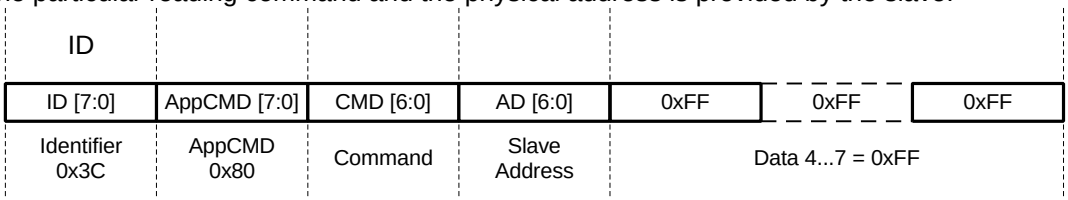


Figure 31: LIN - Preparing Frame Type#8

7.1.9 LIN - Description of Application Commands

Communications between the TMC211 and a LIN Master takes place via a large set of commands.

Reading commands are used to:
- Get actual status information, e.g. error flags
- Get actual position of the Stepper Motor
- Verify the right programming and configuration of the TMC211

Writing commands are used to:
- Program the OTP Memory
- Configure the TMC211 with motion parameters (e.g. max/min speed, acceleration, stepping mode, a.s.o.)
- Provide target positions to the Stepper motor

7.1.10 LIN - Command Overview

Command Mnemonic	Function
GetActualPos	Returns actual position of the motor
GetFullStatus	Returns actual, target and secure position also the complete status of the circuit
GetOTPParam	Returns OTP memory content
GetStatus	Returns quick status of the circuit
GotoSecurePosition	Drives motor to secure position
HardStop	Immediate full stop
ResetPosition	Actual and target position becomes zero
ResetToDefault	Overwrites the chip RAM with OTP contents
RunInit	Reference Search
SetMotorParam	Sets motor parameter
SetOTPparam	Programs the selected byte of OTP memory
SetPosition	Drives the motor to the target position
SetPositionShort (1 Motor)	Drives the motor to the target position (Half stepping mode only)
SetPositionShort (2 Motors)	Drives 2 motors to the target position (Half stepping mode only)
SetPositionShort (4 Motors)	Drives 4 motors to the target position (Half stepping mode only)
Sleep	Causes circuit to go into sleep mode
SoftStop	Stops the motor with deceleration phase

Table 20: LIN - Command Overview

7.1.11 LIN - Command Description

7.1.11.1 GetFullStatus1

This command is provided to the circuit by the Master to get a complete status of the circuit and of the Stepper-motor. The parameters sent via LIN interface to the master are:

- coil peak and hold currents value (Irun and Ihold)
- maximum and minimum velocities for the Stepper-motor (Vmax and Vmin)
- direction of movement clockwise / counterclockwise (Shaft)
- stepping mode (StepMode) (Table 13: StepMode on page 24)
- acceleration (deceleration) for the Stepper motor (Acc)
- acceleration shape (AccShape)
- status information (see further)
  - motion status <Motion [2:0]>
  - over current flags for coil #1 <OVC1> and coil #2 <OVC2>
  - digital supply reset <VddReset>
  - charge pump status <CPFail>
  - external switch status <ESW>
  - step loss <StepLoss>
  - electrical defect <ElDef>
  - under voltage <UV2>
  - temperature information <Tinfo>
  - temperature warning <TW>
  - temperature shutdown <TSD>

The following flags are attempt to reset:
<TW>, <TSD>, <UV2>, <ElDef>, <StepLoss>, <CPFail>, <OVC1>, <OVC2> and <VddReset>

Note: GetFullStatus corresponds to 2 successive LIN in-frame responses with 0x3D indirect ID. It is not mandatory for the LIN master to initiate the second in-frame response if position information are not needed by the application.

GetFullStatus1 Preparing Frame (type#7 or #8)									
Source	**Byte**	**Content**	**Structure**							
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x01						
	2	Slave Address	1	AD [6:0]						

Table 21: GetFullStatus preparing frame

Note: * according to parity calculation

GetFullStatus1 In-frame response 1 (type#6)									
Source	**Byte**	**Content**	**Structure**							
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	1	1	1	1	1	0	1
Slave	1	Slave Address	1	AD [6:0]						
	2	Irun + Ihold	Irun [3:0]				Ihold [3:0]			
	3	Vmax + Vmin	Vmax [3:0]				Vmin [3:0]			
	4	Status + Acc	Acc Shape	StepMode [1:0]		Shaft	Acc [3:0]			
	5	Status	VddReset	StepLoss	ElDef	UV2	TSD	TW	Tinfo [1:0]	
	6	Status	Motion [2:0]			ESW	OVC1	OVC2	1	CPFail
	7	LIN error status register	See Table 17: LIN - Error Status Register on page 30							
	8	Do not care	0xFF							

Table 22: GetFullStatus In-frame response1

GetFullStatus1 In-frame response 2 (type#6)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	1	1	1	1	1	0	1
Slave	1	Slave Address	1	AD [6:0]						
	2	ActualPosition	ActPos [15:8]							
	3		ActPos [7:0]							
	4	TargetPosition	TagPos [15:8]							
	5		TagPos [7:0]							
	6	SecurePosition	SecPos [7:0]							
	7		1	1	1	1	1	SecPos [10:8]		
	8	Do not care	0xFF							

Table 23: GetFullStatus In-frame response2

7.1.11.2 GetActualPos

This command is provided to the circuit by the LIN master to get the actual position of the stepper motor. GetActualPos provides also a quick status of the circuit and of the stepper motor, identical to that obtained by command GetFullStatus. The GetActualPos will not attempt to reset any flags.
GetActualPos with direct ID:

GetActualPos with direct ID (type#5)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	1	0	ID3	ID2	ID1	ID0
Slave	1	Slave Address	ESW	AD [6:0]						
	2	Actual Position	ActPos [15:8]							
	3		ActPos [7:0]							
	4	Status	VddReset	StepLoss	ElDef	UV2	TSD	TW	Tinfo [1:0]	

Table 24: GetActualPos with direct ID

Note: * according to parity calculation

GetActualPos with indirect ID:

GetActualPos Preparing Frame (type#7 or #8)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x00						
	2	Slave Address	1	AD [6:0]						

Table 25: GetActualPos preparing frame

Note: * according to parity calculation, ID [4:0]: Dynamically allocated identifier.

GetActualPos In-frame response (type#6)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	1	1	1	1	1	0	1
Slave	1	Slave Address	1	AD [6:0]						
	2	ActualPosition	ActPos [15:8]							
	3		ActPos [7:0]							
	4	Status	VddReset	StepLoss	ElDef	UV2	TSD	TW	Tinfo [1:0]	
	5	Do not care	0xFF							
	6	Do not care	0xFF							
	7	Do not care	0xFF							
	8	Do not care	0xFF							

Table 26: GetActualPos In-frame response

7.1.11.3 GetOTPParam

This command is provided to the circuit by to read the content of an OTP Memory. For more information refer to Table 11: OTP Memory Structure on page 24.

GetOTPParam Preparing Frame (type#7 or #8)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x02						
	2	Slave Address	1	AD [6:0]						

Table 27: GetOTPParam preparing frame

Note: * according to parity calculation

GetOTPParam In-frame response (type#6)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	1	1	1	1	1	0	1
Slave	1	OTP Memory	OTP byte @0x00							
	2	OTP Memory	OTP byte @0x01							
	3	OTP Memory	ADM	HW2	HW1	HW0	PA3	PA2	PA1	PA0
	4	OTP Memory	OTP byte @0x03							
	5	OTP Memory	OTP byte @0x04							
	6	OTP Memory	OTP byte @0x05							
	7	OTP Memory	OTP byte @0x06							
	8	OTP Memory	OTP byte @0x07							

Table 28: GetOTPParam In-frame response

HW [2:0]: Not stored in OTP memory, the hardwired address is returned by GetOTPParam as if stored at address 0x02 of the OTP memory.

7.1.11.4 GetStatus

This command is provided to the circuit by the LIN master to get a quick status (compared to that of GetFullStatus command) of the circuit and the stepper motor.

The following flags are attempt to reset:

<TW>, <TSD>, <UV2>, <ElDef>, <StepLoss>, <CPFail>, <OVC1>, <OVC2> and <VddReset>

GetStatus with direct ID (type#5)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
Slave	1	Slave Address	ESW	AD [6:0]						
	2	Status	VddReset	StepLoss	ElDef	UV2	TSD	TW	Tinfo[1:0]	

Table 29: GetStatus with direct ID

Note: * according to parity calculation
ID [4:0]: Dynamically allocated identifier.

7.1.11.5 GotoSecurePosition

This command is provided by the LIN master to one or all stepper motors to move to the secure position SecPos [10:0]. It can be also triggered if the LIN communication is lost or after an initialization phase. If <Broad> is set to zero all stepper motors connected to the LIN bus will reach their secure position.

GotoSecurePosition writing frame (type#1 or type#4)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x04						
	2	Slave Address	Broad	AD [6:0]						

Table 30: GotoSecurePosition writing frame

Note: * according to parity calculation

7.1.11.6 HardStop

This command is internally triggered when an electrical problem is detected in one or both coils, leading to switch off the H-bridges. If this problem is detected while the motor is moving, the <StepLoss> flag is raised allowing to warn the Master that steps may have been lost at the next GetStatus command. A HardStop command can also be issued by the Master for some safety reasons. If <Broad> is set to zero all the stepper motors connected to the LIN bus will stop.

HardStop writing frame (type#1 or type#4)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x05						
	2	Slave Address	Broad	AD [6:0]						

Table 31: HardStop writing frame

Note: * according to parity calculation

7.1.11.7 SoftStop

If a SoftStop command occurs during a motion of the Stepper motor, it provokes an immediate deceleration to VMIN followed by a stop, regardless of the position reached. One the motor is stopped

TagPos register is overwritten with value in ActPos register to ensure keeping the stop position. This command occurs in the following cases:

- The chip temperature rises the Thermal shutdown threshold.
- The Master requests a SoftStop.

If <Broad> is set to zero all the stepper motors connected to the LIN bus will stop with deceleration.

SoftStop writing frame (type#1 or type#4)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x0F						
	2	Slave Address	Broad	AD [6:0]						

Table 32: SoftStop writing frame

Note: * according to parity calculation

7.1.11.8 ResetPosition

This command is provided to the circuit by the Master to reset ActPos and TagPos registers, in order allow a positioning for an initialization of the Stepper-motor position. If <Broad> is set to zero all circuits connected to the LIN bus will reset their ActPos and TagPos registers.

ResetPosition writing frame (type#1 or type#4)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x06						
	2	Slave Address	Broad	AD [6:0]						

Table 33: SoftStop writing frame

Note: * according to parity calculation

7.1.11.9 ResetToDefault

This command is provided to the circuit by the Master in order to reset the whole Slave node into the initial state. ResetToDefault will for instance **overload the RAM** with the Reset state of the Registers parameters. This is another way for the Master to initialize a slave node in case of emergency, or simply to refresh the RAM content.
Note: ActPos is not modified by a ResetToDefault command, and it's value is copied into TagPos register in order to avoid an attempt to position the motor to '0'. If <Broad> is set to zero all circuits connected to the LIN bus will reset to default.

ResetToDefault writing frame (type#1 or type#4)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x07						
	2	Slave Address	Broad	AD [6:0]						

Table 34: ResetToDefault writing frame

Note: * according to parity calculation

7.1.11.10 RunInit

This command is provided to the circuit by the Master in order to initialize positioning of the motor by seeking the zero (or reference) position. Refer to 6.1.11 Reference Search / Position initialization on page 16.

It leads to a sequence of the following commands:
- SetMotorParam(Vmax, Vmin);
- SetPosition(Pos1);
- SetMotorParam(Vmin, Vmin);
- SetPosition(Pos2);
- ResetPosition

Once the RunInit command is started it can not be interrupted by any other command. Except a condition which leads to a motor shutdown (See 6.1.10 Motor Shutdown Management) happens or a HardStop command is received.
Furthermore the master has to check that the actual position of the stepper motor corresponds **not** to the target position of the first motion. This is very important otherwise the circuit goes into a deadlock state. Once the circuit finds into deadlock state only a HardStop command followed by a GetFullStatus command will cause the circuit to leave the deadlock state. If <Broad> is set to zero all circuits connected to the LIN bus will run the init sequence.

RunInit with indirect ID (type#4)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	0	1	1	1	1	0	0
	1		AppCMD = 0x80							
	2	Command	1	CMD [6:0] = 0x08						
	3	Slave Address	Broad	AD [6:0]						
	4	Vmax + Vmin	Vmax [3:0]				Vmin [3:0]			
	5	Target Position 1	Pos1 [15:8]							
	6		Pos1 [7:0]							
	7	Target Position 2	Pos2 [15:8]							
	8		Pos2 [7:0]							

Table 35: RunInit with indirect ID

Note: Vmax [3:0]: Maximum Velocity for first motion of the run
Vmin [3:0]: Minimum Velocity for first motion and maximum velocity for the second motion of the run
Pos1 [15:0]: First target position to be reached during the init run.
Pos2 [15:0]: Second target position to be reached during the init run.

7.1.11.11 SetPosition

This command is provided to the circuit by the Master to the motors to a given position relative to the zero position, defined in number of half or micro steps, according to StepMode[1:0] value.
SetPosition will not be performed if one of the following flags is set to one:
- temperature shutdown <TSD>
- under voltage <UV2>
- step loss <StepLoss>
- electrical defect <ElDef>

SetPosition command with direct ID:

SetPosition with direct ID (type#3)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1	TargetPosition	TagPos [15:8]							
	2		TagPos [7:0]							

Table 36: SetPosition with direct ID

Note: * according to parity calculation
ID [4:0]: Dynamically allocated identifier.

SetPosition command with a general purpose ID:
If <Broad> is set to zero all stepper motors connected to the LIN bus are going to TagPos1.

SetPositon with a general purpose identifier (type#1)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	1	0	ID3	ID2	ID1	ID0
	1	Command	1	CMD [6:0] = 0x0B						
	2	Slave Address	Broad	AD [6:0]						
	3	Target Position 1	TagPos1 [15:8]							
	4		TagPos1 [7:0]							

Table 37: SetPosition with a general purpose ID

Note: * according to parity calculation

SetPosition for 2 motors with indirect ID:

SetPosition for 2 motors with indirect ID (type#4)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	0	1	1	1	1	0	0
	1		AppCMD = 0x80							
	2	Command	1	CMD [6:0] = 0x08						
	3	Slave Address 1	1	AD1 [6:0]						
	4	Target Position 1	TagPos1 [15:8]							
	5		TagPos1 [7:0]							
	6	Slave Address 2	1	AD2 [6:0]						
	7	Target Position 2	TagPos2 [15:8]							
	8		TagPos2 [7:0]							

Table 38: SetPosition for 2 motors with indirect ID

Note: Adn [6:0]: Motor #n physical address
TagPosn [15:0] Signed 16-bit position set-point for motor #n.

7.1.11.12 SetMotorParameter

This command is provided to the circuit by the Master to set the values for the Stepper motor parameters (listed below) in RAM. Note: it is not recommended to change Vmax, Vmin or Acc while a motion is ongoing, otherwise correct positioning is not guaranteed.

- coil peak current value (Irun)
- coil hold current value (Ihold)
- maximum velocity for the Stepper-motor (Vmax)

- minimum velocity for the Stepper-motor (Vmin)
- acceleration shape (AccShape)
- stepping mode (StepMode)
- indicator of the Stepper-motor reference position (Shaft)
- acceleration (deceleration) for the Stepper-motor (Acc)
- secure position for the Stepper-motor (SecPos)

If <Broad> is set to zero all stepper motors connected to the LIN bus will set the parameters in their RAMs.

SetMotorParam with indirect ID (type#4)										
Source	**Byte**	**Content**	**Structure**							
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	0	1	1	1	1	0	0
	1		AppCMD = 0x80							
	2	Command	1	CMD [6:0] = 0x08						
	3	Slave Address	Broad	AD [6:0]						
	4	Irun + Ihold	Irun [3:0]				Ihold [3:0]			
	5	Vmax + Vmin	Vmax [3:0]				Vmin [3:0]			
	6	SecPos + Acc	SecPos [10:8]			Shaft	Acc [3:0]			
	7	Secure Position2	SecPos [7:0]							
	8	Status	1	1	1	Acc Shape	StepMode [1:0]		1	1

Table 39: SetMotorParam with indirect ID

7.1.11.13 SetOTP

This command is provided to the circuit by the Master in order to zap the OTP memory. If <Broad> is set to zero all circuits connected to the LIN bus will zap their OTP memories. Note please refer to Table 52: DC Parameters Supply and Voltage regulator on page 50 that the correct supply voltage is applied to the chip, otherwise the circuit will be damaged.

SetOTPparam with indirect ID (type#4)										
Source	**Byte**	**Content**	**Structure**							
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	0	0	1	1	1	1	0	0
	1		AppCMD = 0x80							
	2	Command	1	CMD [6:0] = 0x0A						
	3	Slave Address	Broad	AD [6:0]						
	4	OTP Address	1	1	1	1	1	OTPA [2:0]		
	5	OTP Data	Data [7 :0]							
	6	Do not care	0xFF							
	7	Do not care	0xFF							
	8	Do not care	0xFF							

Table 40: SetOTPParam with indirect ID

7.1.11.14 SetPosition Short

This command is provided to the circuit by the LIN master to drive one, two or four motors to a given absolute position. This command is valid for half stepping modes (<StepMode> = “00”) and is ignored for other stepping modes. If <Broad> is set to zero all circuits connected to the LIN bus will go to the desired position (only valid for SetPositionShort (1 Motor)).
The physical address is coded on 4 bits, hence SetPositionShort can only be used with a network implementing a maximum of 16 slave nodes. These 4 bits are normally corresponding to the bits PA [3:0] in OTP memory (Address 0x00), while bits AD [6:4] must be at ‘1’. Two different cases must in

fact be considered, depending on the programmed value of bit ADM in the OTP memory. See 6.2.2 OTP Memory Structure on page 24.

ADM	AD[3]	Pin HW0	Pin HW1	Pin HW2	Bit PA0 in OTP memory
0	X	Tied to Vdd		Tied to Vbb	AD[0]
1	0			Tied to Gnd	1
1	1			Tied to Vbat	1

Table 41: ADM bit in SetPositionShort Command

SetPositionShort (1 Motor) with dynamic ID (type#2)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	0	ID4	ID3	ID2	ID1	ID0
	1		Pos [10:8]			Broad	AD [3:0]			
	2	Command	Pos [7:0]							

Table 42: SetPositionShort (1 Motor)

Note: * according to parity calculation
ID [4:0]: Dynamically allocated identifier to SetPosition short command with 2 data bytes.

SetPositionShort (2 Motors) with dynamic ID (type#2)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	1	1	1	1	0	0
	1		Pos1 [10:8]			1	AD1 [3:0]			
	2	Command	Pos1 [7:0]							
	3	Slave Address	Pos2 [10:8]			1	AD2 [3:0]			
	4	OTP Address	Pos2 [7:0]							

Table 43: SetPositionShort (2 Motors)

Note: * according to parity calculation
ID [4:0]: Dynamically allocated identifier to SetPosition short command with 4 data bytes.
ADn [3:0]: Motor #n physical address least significant bits
Posn [10:0]: Unsigned 11-bit position set point for Motor #n

SetPositionShort (4 Motors) with dynamic ID (type#2)									
Source	**Byte**	**Content**	**Structure**						
			bit 7	**bit 6**	**bit 5**	**Bit 4**	**bit 3**	**bit 2**	**bit 1**	**bit 0**
Master	0	Identifier	*	*	1	1	ID3	ID2	ID1	ID0
	1		Pos1 [10:8]			1	AD1 [3:0]			
	2	Command	Pos1 [7:0]							
	3	Slave Address	Pos2 [10:8]			1	AD2 [3:0]			
	4	OTP Address	Pos2 [7:0]							
	5	OTP Data	Pos3			1	AD3 [3:0]			
	6	Do not care	Pos3 [7:0]							
	7	Do not care	Pos4			1	AD4 [3:0]			
	8	Do not care	Pos4 [7:0]							

Table 44: SetPositionShort (4 Motors)

Note: * according to parity calculation
ID [4:0]: Dynamically allocated identifier to SetPosition short command with 8 data bytes.
ADn [3:0]: Motor #n physical address least significant bits
Posn [10:0]: Unsigned 11-bit position set point for Motor #n

7.1.11.15 Sleep Mode

According to LIN specification, rev. 1.2, the sleep and wake up (or standby) mode is implemented. The interface can work in one of the following modes:

- Normal mode: EN = '1', Vdd = high, INH = '1'
- Sleep mode: EN = '0' after the normal mode. The interface consumption is limited to Isleep. Only the receiver is working.
- Standby mode: when, in sleep mode, the LIN pin is entered directly, i.e. when Vbb goes high, INH is immediately activated. The initial state of INH signal is ensured by an internal power-on-reset circuitry inside the wake-up control.

7.1.12 Positioning Task Example

The TMC211 has to perform a positioning task, where the actual position of the stepper motor is unknown. The desired target position is 3000 µsteps away from position 0. See Figure 32: Positioning Example: Initial situation.

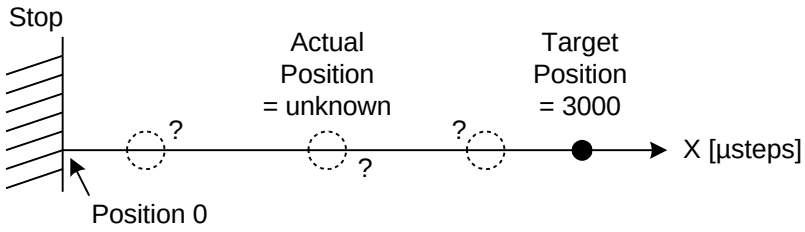


Figure 32: Positioning Example: Initial situation

The following sequence of commands is to send to the slave in order to complete the scenario described above (assumed after power on):

GetFullStatus

The command is used to read the current status of the TMC211. Electrical or environmental problems will be reported, furthermore the circuit leaves the shutdown state and is ready for action.
Furthermore the circuit provides the actual and target position. This information is very important, because if the actual position corresponds to the first target position of the RunInit command the circuit will enter a deadlock state. The master must take care that both positions are containing different values.
See 7.1.11.1 GetFullStatus1 command on page 36 and for deadlock problems see 6.1.11 Reference Search / Position initialization on page 16.

SetMotorParam

In order to drive the stepper motor with a desired motion parameters like torque, velocity, a.s.o.. the SetMotorParam command must issued. See 7.1.11.12 SetMotorParameter on page 42.

RunInit

Hence the actual position is unknown, a position initialization has to be performed. The first motion must drive the stepper motor into the stop for sure. The second motion is a very short motion to bring the motor out of the stop. The actual position is then set to zero automatically after the second motion is finished. See 7.1.11.10 RunInit command on page 41.

After reference search the actual situation looks like as described in Figure 33: Positioning Example: Situation after reference search. Actual position of the stepper motor corresponds to zero, the target position is 3000 µsteps away from the actual position.

Figure 33: Positioning Example: Situation after reference search

Now the positioning command SetPosition can be issued in order to drive the stepper motor to the desired position.

SetPosition

This command will cause the stepper motor to move to the desired target position. See 7.1.11.11 SetPosition on page 41. After the motion has been finished the situation looks like as described in Figure 34: Positioning Example: Motion finished.

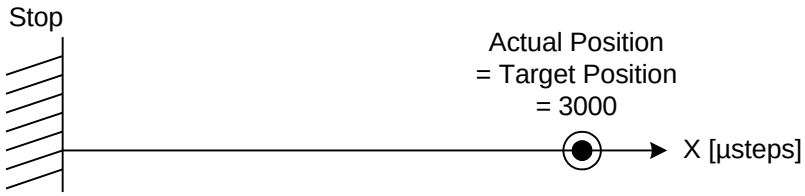


Figure 34: Positioning Example: Motion finished

Afterwards the actual status and position can be verified by using GetFullStatus commands. The master can check if a problem, caused by electrical or temperature problems, occurred. Furthermore the actual position is read.

8 Package Outline

8.1 SOIC-20

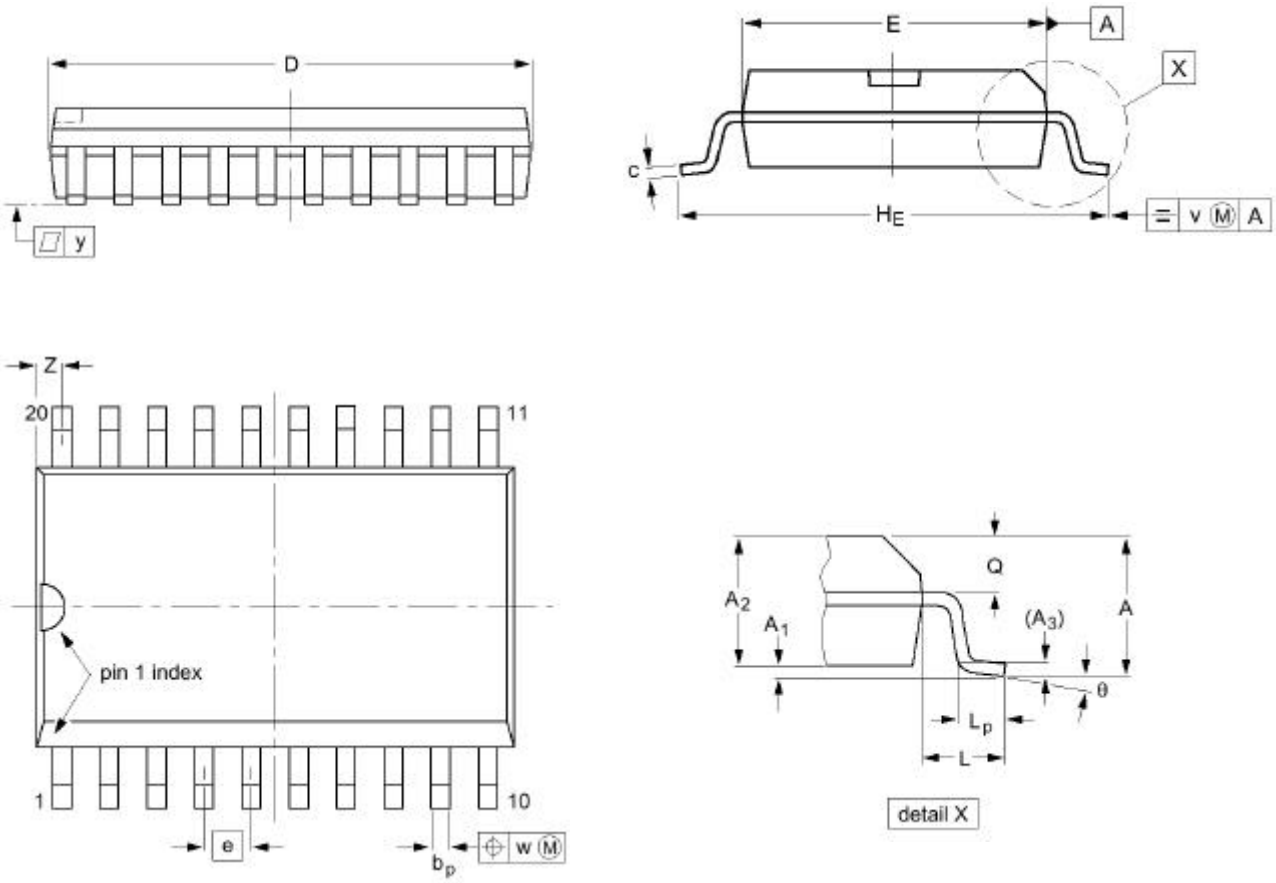


Figure 35: Package Outline SOIC-20

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_E	L	L_p	Q	v	w	y	$Z^{(1)}$	Θ
mm	2.65	0.30 0.10	2.45 2.25	0.25	0.49 0.36	0.32 0.23	13.0 12.6	7.6 7.4	1.27	10.65 10.00	1.4	1.1 0.4	1.1 1.0	0.25	0.25	0.1	0.9 0.4	8° 0°
inches	0.10	0.012 0.004	0.096 0.089	0.01	0.019 0.014	0.013 0.009	0.51 0.49	0.30 0.29	0.050	0.419 0.394	0.055	0.043 0.016	0.043 0.039	0.01	0.01	0.004	0.035 0.016	

Table 45: SOIC-20 Mechanical Data

Note: inch dimensions are derived from the original mm dimensions

8.2 QFN-32

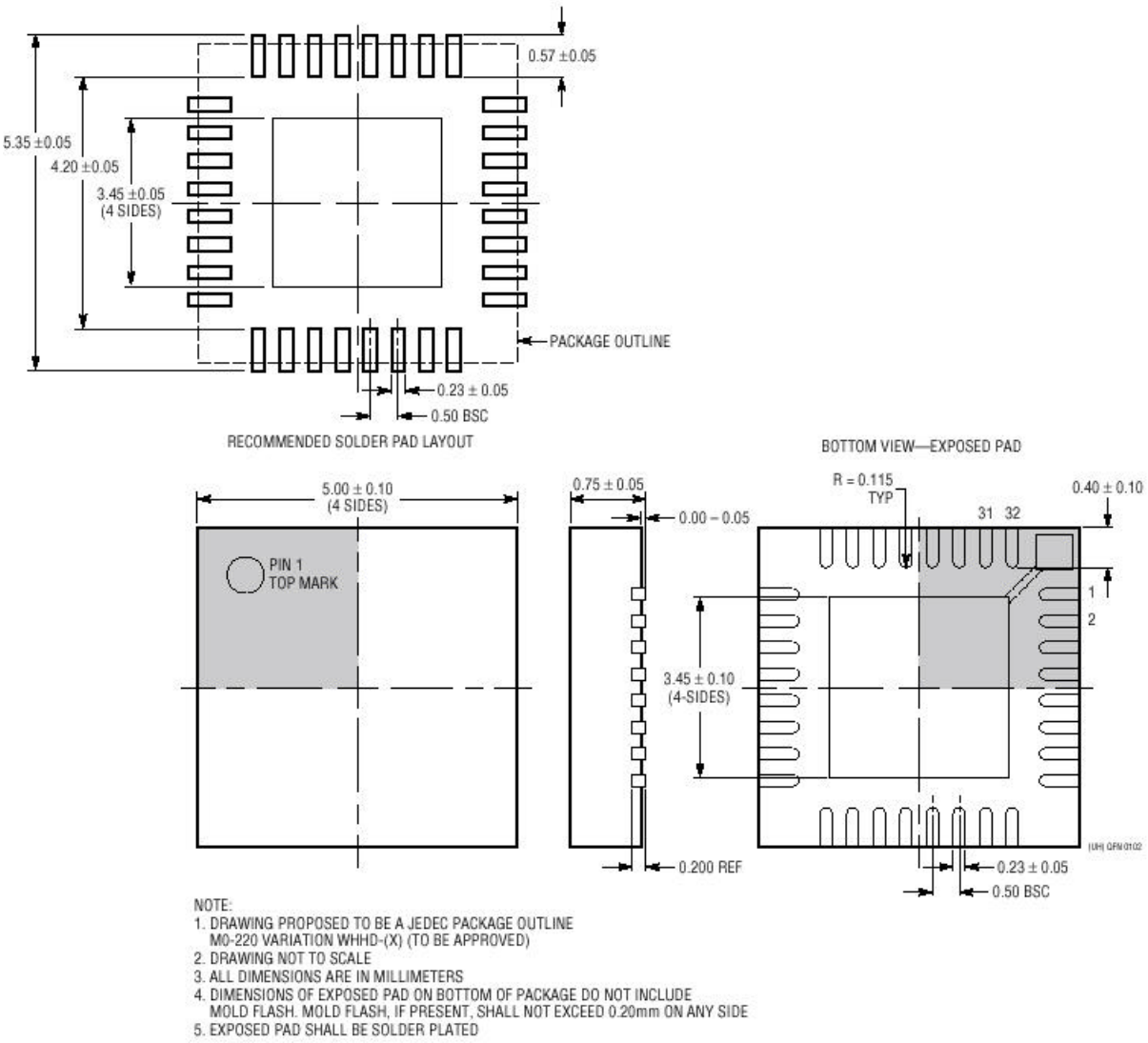


Figure 36: Package Outline QFN-32

9 Electrical Characteristics

9.1 Absolute Maximum Ratings

Parameter		Min	Max	Unit
Vbat	Supply Voltage	-0.3	+35	V
Vlin	Bus input voltage	-80	+80	V
Tamb	Ambient temperature under bias (*)	-50	+150	°C
Tst	Storage temperature	-55	+160	°C
Vesd (**)	Electrostatic discharge voltage on LIN pin	-4	+4	kV
	Electrostatic discharge voltage on other pins	-2	+2	kV

Table 46: Absolute Maximum Ratings

(*) The circuit functionality is not guaranteed
(**) Human body model (100pF via 1.5 KΩ)

9.2 Operating Ranges

Parameter			Min	Max	Unit
Vbat	Supply Voltage		+8	+29	V
Top	Operating temperature range	Vbat <= 18V	-40	+125	°C
		Vbat <= 29V	-40	+85	°C

Table 47: Operating Ranges

9.3 DC Parameters

Motor Driver							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
IMSmax Peak	MOTXP MOTXN MOTYP MOTYN	Max current through motor coil in normal operation			800		mA
IMSmax RMS		Max RMS current through coil in normal operation			570		mA
RDSon		On resistance for each pin (including bond wire)	To be confirmed by characterization			1	Ω
IMSL		Leakage current	HZ Mode, 0V < V(pin) < Vbb	-50		+50	µA

Table 48: DC Parameters Motor Driver

LIN Transmitter							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Ibus_on	LIN	Dominant state, driver on	Vbus = 1.4V	40			mA
Ibus_off		Dominant state, driver off	Vbus = 0V	-1			mA
Ibus_off		Recessive state, driver off	Vbus = Vbat			20	µA
Ibus_lim		Current limitation		50		200	mA
Rslave		Pull-up resistance		20	30	47	kΩ

Table 49: DC Parameters LIN Transmitter

LIN Receiver							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Vbus_dom	LIN	Receiver dominant state		0		0.4	Vbb
Vbus_rec		Receiver recessive state		0.6		1	Vbb
Vbus_hys		Receiver hysteresis		0.05		0.2	Vbb

Table 50: DC Parameters LIN Receiver

Thermal Warning and shutdown							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Ttw		Thermal Warning		138	145	152	°C
Ttsd (*)		Thermal Shutdown			Ttw + 10		°C
Tlow		Low Temperature Warning			Ttw - 155		°C

Table 51: DC Parameters Thermal Warning and shutdown

(*) NO more than 100 cumulated hours in life time above Ttsd

Supply and Voltage regulator							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Vbb	VBB	Nominal operating supply range		6.5		18	V
VbbOTP		Supply Voltage for OTP zapping		8.5		9.5	V
UV1		Low voltage high threshold		8.8	9.4	9.8	V
UV2		Stop voltage low threshold		8.1	8.5	8.9	V
Ibat		Total current consumption	Unloaded Outputs		10		mA
Vdd	VDD	Internal regulated output (**)	8V < Vbb < 18V Cload = 1µF (+100nF cer.)	4.75	5	5.25	V
IddStop		Digital current consumption	Vbb < UV2		2		MA
VddReset		Digital supply reset level (***)				4.4	V
IddLim		Current limitation	Pin shorted to ground			40	MA

Table 52: DC Parameters Supply and Voltage regulator

(*) To be confirmed by measurements
(**) The RAM content will not be altered above this voltage
(***) External resistance value seen from pin SWI or HW2, including 1kΩ series resistor

Switch Input and hardwired address input HW[2]							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Rt_OFF	SWI HW2	Switch OFF resistance (5)	Switch to GND or Vbat	10			k Ω
Rt_ON		Switch ON resistance (5)				2	k Ω
Vbb_sw		Vbb range for guaranteed operation of SWI and HW2		6		18	V
Vmax_sw		Maximum Voltage	T < 1s			40	V
Ilim_sw		Current limitation	Short to GND or Vbat		30		mA

Table 53: DC Parameters Switch Input and hardwired address input

Hardwired address inputs and test pin							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Vlow	HW0 HW1 TST	Input level high		0.7			Vdd
Vhigh		Input level low				0.3	Vdd
HWhyst		Hysteresis		0.075			Vdd

Table 54: DC Parameters and Hardwired address inputs and test pin

Charge Pump							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Vcp	VCP	Output Voltage	Vbb > 15V	Vbb+10	Vbb+12.5	Vbb+15	V
Cbuffer		External Buffer Capacitor	Vbb > 8V	Vbb+5.8 220		 470	V nF
Cpump	CPP CPN	External pump Capacitor		220		470	nF

Table 55: DC Parameters Charge Pump

9.4 AC Parameters

Power-Up							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Tpu		Power-Up time				10	ms

Table 56: AC Parameters Power-Up

Switch Input and hardwired address input HW[2]							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Tsw	SWI HW2	Scan Pulse Period		921	1024	1127	µs
Tsw_on		Scan Pulse Duration			1/16		Tsw

Table 57: AC Parameters Switch Input and hardwired address input

Motor Driver							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Fpwm	MOTxx	PWM frequency		18	20	22	KHz
Tbrise		Turn-On transient time	Between 10% and 90%		350		ns
Tbfall		Turn-Off transient time			250		ns

Table 58: AC Parameters Motor Driver

LIN Transmitter							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
Slope_F/R	LIN	Slope falling (or rising) edge	Between 40% and 60%	0.1		3	V/µs
t_slope_F/R		Slope time falling (or rising) edge	extrapolated	2.6		22.5	µs
T_tr_F		Propagation delay TxD low to bus		0.1	1	4	µs
T_tr_R		Propagation delay TxD high to bus		0.1	1	4	µs
t_slope_Sym		Slope time symmetry	t_slope_F – t_slope_R	-4		4	µs
Tsym_tr		Transmitter delay symmetry	T_tr_F – T_tr_R	-2		2	µs

Table 59: AC Parameters LIN Transmitter

LIN Receiver							
Symbol	**Pin(s)**	**Parameter**	**Test condition**	**Min**	**Type**	**Max**	**Unit**
T_rec_F	LIN	Propagation delay bus dominant to TxD low		0.1	4	6	µs
T_rec_R		Propagation delay bus recessive to TxD high		0.1	4	6	µs
Tsym,Rec		Receiver delay symmetry		-2		2	µs
Twake		Wake-up delay time		50	100	200	µs

Table 60: AC Parameters LIN Receiver

Revision History

Version	Date	Comments
up to 0.90	July 9, 2003	before v. 0.90 changes on unpublished internal versions only

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 Microchips GmbH by request to info@trinamic.com