TMC211 / TMC222 FAQ list:

TMC211 and TMC222 are identical except for the communications interface. Therefore most questions and answers are valid for both devices. Where differences occur the description applies to the TMC211 while text in parenthesis (TMC222: …) applies to the TMC222.

1. Using the bus interface

Q: How many devices can be operated on the same bus?

A: 128 (TMC222: 32) devices can be discriminated by means of the physical address. However, it depends on some factors if this high number really makes sense. First of all it has to be checked if each device can be serviced under any circumstances in the maximum allowed time taking the bus speed and the individual real-time requirements of each device into account. Second, the idea of reserving address 0 for OTP physical address programming during system installation and defective parts replacement reduces the number to 120 (TMC222: 30). Third, the TMC211 has faster and slower types of the GetActualPos, GetStatus and SetPosition commands. The faster types of course can satisfy harder real-time requirements but can only be addressed to 8 different physical devices. If the faster types shall be used only 8 TMC211 devices can be connected to the same LIN bus.

Q: How to program the OTP physical address bits of a device if there are more devices connected to the same bus?

A: The problem here is that all new devices are shipped with the OTP physical address bits set to zero making it difficult to address just one device with the SetOTP command. Use HW0, HW1 or HW2 (TMC222: HW) input as chip select line to be able to address just one device by SetOTP. If this is impractible since the HW0/HW1/HW2 (TMC222: HW) inputs are hardwired or not controllable for any other reason the only alternative is to assemble and program one device after the other. I.e., assemble only first device and program the desired non-zero address, then assemble the second device and program the desired non-zero address, and so on until all devices are assembled and programmed. This is also a good service concept when replacing defective devices in the field: The idea is that all devices are programmed to different non-zero physical addresses at production/installation time. Once a defective device is being replaced the replacement part can easily be addressed by SetOTP since it is the only part with physical address zero.

2. General problems when getting started

Q: May I draw a small amount of current from the device's VDD-pin in order to supply an external component?

A: Yes, up to 10 mA can be drawn from the VDD pin.

Q (TMC222 only): Is it possible to connect the pullup-resistors required for SDA and SCL to the device's VDD-pin?

A: Yes, it is. However, the total current drawn from the VDD-pin by external components must not exceed 10 mA. I.e., each of the two pullup-resistors must have a minimum resistance of 1 KOhm.

Q: If the power-supply VBAT is 24V, may I still use 16V capacitors for VCP, CPN-CPP?

A: Yes, you can.

Q: What is the maximum series-resistance allowable between SWI and an external switch to GND?

A: Maximum 2 KOhm for ON state, minimum 10 KOhm for OFF state.

Q (TMC222 only): How to interface the TMC222 SDA and SCL lines to a 3.3V system?

A: There are different solutions depending on the maximum SCL clock frequency and the actual supply voltage:

1. SCL clock frequency ≤ 100 KHz: TMC222 SDA and SCL high level input voltages must be at least 3.0V for this frequency range. Pullup-resistors from SCL and SDA to the system's 3.3V supply voltage are sufficient.

2. SCL clock frequency > 100 KHz: TMC222 SDA and SCL high level input voltages must be at least 3.5V for this frequency range. Many 3.3V systems are allowed to run at up to 3.6V. If 3.6V can be guaranteed as the system's power supply voltage pullup-resistors from SDA and SCL to the system's 3.6V supply voltage are sufficient.

3. If solutions 1 and 2 are not possible: Some kind of level shifting is required. One way is to insert non-inverting open-collector TTL buffers as contained in the 7407 device into the SDA and SCL lines. The 7407 needs 5V supply voltage and has 3.3V compliant I/Os since the inputs have TTL voltage thresholds (2V min. high level) and the outputs have open collectors.

- Insert two '07 buffers connected anti-parallel between the SDA pin of the TMC222 and the SDA line of the 3.3V side.
- Connect a pullup-resistor to the SDA line at the 3.3V side
- Connect another pullup-resistor to the SDA line at the 5V (TMC222) side.
- Do the same for the SCL line.

Four '07 buffers are required altogether, i.e. only one single 7407 device is needed.

Q: What is the meaning of ElDef?

A: The ElDef flag ('Electrical Defect') is the logical ORing of the OVC1 and OVC2 flags. OVC1 is set to one in case of an overcurrent (coil short) or open load condition (selected coil current is not reached) for coil 1. OVC2 is the equivalent for coil 2.

Q: What could be the reason for ElDef / OVC1 / OVC2 being set to one?

A: There are a number of possible causes:

- Motor not connected (→ open load)
- Connected motor has shorted coils (→ overcurrent) or broken coils (→ open load)
- Motor coils connected to the wrong device pins
- Selected coil current can not be reached (→ open load) due to high coil impedance or low supply voltage. Solution: Select a lower coil run/hold current or rise the supply voltage. Generally: the calculated voltage required to reach a desired coil current at a given coil

resistance (V = I • R) must be significantly lower than actual supply voltage due to the coil inductivity.

Q: Should the external switch be normally closed or open when the reference position is hit?

A: The SWI input resp. the ESW flag have neither effect on any internal state machine nor on command processing, even not on the RunInit command. ESW must be polled by software using GetActualPos or GetFullStatus commands (TMC222: GetFullStatus1). The software can simply be adapted to whatever state the switch is in when the reference position is hit, i.e. closed or open.

3. Using the device

Q: What is the minimum constant velocity?

A: 3 FS/s (three fullsteps per second). The following settings must be applied:
- Vmax=0 (99 FS/s)
- Vmin=1 (1/32 of Vmax)
- AccShape=1 (Velocity set to Vmin, no acceleration to Vmax).

Q: What is the meaning of the 'Shaft' bit?

A: The Shaft bit determines the rotating direction of the motor, i.e. clockwise or counter-clockwise rotation.

Q: How to generate an interrupt when the target position is reached?

A: This is not possible. The device hasn't any interrupt output at all. Just poll ActPos or Motion[2:0] using an appropriate command.

Q: How can I ensure that I always get consistent data for ActPos and ESW?

A: This is ensured by design. ActPos and ESW are latched synchronously by the same internal signal edge. The update period is about one millisecond (typ. 1024 µs). Each time ActPos and ESW are read by the GetActualPos command the result will be a snapshot of both values taken at the same point in time.
(TMC222: There isn't a single command to read both ActPos and ESW simultaneously. GetFullStatus1 will read ESW whereas GetFullStatus2 will read ActPos. Thus it can not be guaranteed that consistent values are read as long as a motion is in progress.)

Q: How to specify a second target position to go to immediately after a first target position has been reached?

A: This is possible using the RunInit command. Note, that after the second target position has been reached the internal position counter ActPos is reset to zero.

Q: Is it possible to change the stepping mode during a running motion?

A: Yes, it is possible and it has immediate effect on the current motion.

Q: Is it possible to change Vmax during a running motion?

A: Yes, it is, if the new velocity is in the same group as the old one (see datasheet section 'Vmax Parameters'). The Vmax values are divided into four groups:

group A: Vmax index = 0
group B: Vmax index = 1, 2, 3, 4 ,5 or 6
group C: Vmax index = 7, 8, 9, 10 ,11 or 12
group D: Vmax index = 13, 14 or 15

Q: What are the effects of changing Vmax group or Vmin during a running motion?

A: First of all and most important: The architecture of the positioning controller ensures that the position counter is ALWAYS correct under ANY circumstances even when changing Vmax velocity group or Vmin during a running motion. I.e. step loss due to changing Vmin or Vmax during an ongoing motion is always ZERO. What actually can occur is overshooting target position followed by driving back when switching to higher speed or short stop of the motion when switching to lower speed. However, in both cases the desired target position is exactly reached in the end. Furthermore, both effects do not occur if the target position is ahead far enough. E.g. when emulating velocity mode, application software regularly sets the target position to the farthest possible position ahead of the current position (see Q&A 'velocity mode'). If this is done early enough each time before the target position is reached both effects are avoided.

Q: How to operate in velocity mode (continuous motion) rather than positioning mode (ramp mode)?

A: There is no velocity mode. The device was designed primarily for positioning tasks so for each motion there has to be specified a target position by the respective command. However, velocity mode can be emulated by repeating the following two commands again and again:

1. Read ActPos using GetActualPos (TMC222: GetFullStatus2) command
2. Set lower 16 bits of [ActPos+32767] as the next target position using SetPosition command

For real continuous motion this sequence has to be repeated before the current target position has been reached.

Q: How to change velocity during emulated velocity mode?

A: Set AccShape=1. This prevents acceleration from Vmin to Vmax, velocity is at Vmin level all the time. As Vmin is determined by both the Vmax index and the Vmin index much more different velocities can be chosen from compared to only changing at Vmax level. To avoid negative effects which can be caused by switching Vmin or Vmax group during a motion (see Q&A 'effects of changing Vmax…') the target position should always be far ahead of the actual position. This can be ensured by applying the velocity mode emulation command sequence (see Q&A 'velocity mode') as often as possible. In between of the velocity mode emulation commands Vmin index and Vmax index can be switched to generate the desired velocity profile. Please note: Set Acc=15 when simultaneously switching to another Vmax group and Vmin index to avoid out-of-range intermediate velocities. When staying in the same Vmax group, lower Acc values can be used to generate smoother velocity transitions. Background: Even though

AccShape=1, the Acc parameter is effective for velocity changes resulting from Vmin changes during a running motion.

Q: Which units, formats and ranges does position information have?

A: All 16-bit position data fields in commands and responses are coded in two's complement format with bit 0 representing 1/16 micro-steps. Hence a position range of –32768…+32767 in units of 1/16 micro-steps is covered regardless of the selected stepping mode (1/2, 1/4, 1/8 or 1/16 micro-stepping). The difference between the stepping modes is the resolution resp. the position of the LSB in the 16-bit position data field: it's bit 0 for 1/16, bit 1 for 1/8, bit 2 for 1/4 and bit 3 for 1/2 micro-stepping. The position range can be regarded as a circle since position –32768 is just 1/16 micro-step away from position +32767. The device will always take the shortest way from the current to the target position, i.e., if the current position is +32767 and the target position is –32768 just 1/16 micro-step will be executed. 65535 1/16 micro-steps in the opposite direction can be achieved for example by two consecutive SetPosition commands with target positions 0 and –32768.

The 11-bit secure position data field can be treated as the upper 11 MSBs of the 16-bit position data fields described above with the 5 LSBs hardwired to zero. Hence it covers the same position range with a reduced resolution: The position range is –1024…+1023 in units of two full-steps.

The 11-bit position data fields of the TMC211 SetPositionShort commands are coded in two's complement format with bit 0 representing half-steps resulting in a position range of –1024…+1023 half-steps. Hence only a quarter of the range of the other position data fields described above is covered. Note, that SetPositionShort command is valid for half-stepping mode only and is ignored for other stepping modes. Furthermore, SetPositionShort can only be used with a maximum of 16 TMC211 devices connected to the LIN bus.

4. Finding the reference position

Q: How do I find a reference position?

A: The recommended way is to use the RunInit command. Two motions are specified through RunInit. The first motion is to reach the mechanical stop. Its target position should be specified far away enough so that the mechanical stop will be reached from any possible starting position. There is no internal stall detection so that at the end of the first motion the step motor will bounce against the mechanical stop loosing steps until the internal target position is reached. The second motion then can be used either to drive in the opposite direction out of the mechanical stop right into the reference position which is a known number of steps away from the mechanical stop. Or the second motion can slowly drive a few steps in the same direction against the mechanical stop to compensate for the bouncing of the faster first motion and stop as close to the mechanical stop as possible.

Q: Can the SWI input help in finding a reference position?

Not directly. The current state of the SWI input is reflected by the ESW flag which can only be polled using the commands GetActualPos or GetFullStatus (TMC222: GetFullStatus1). The SWI input resp. the ESW flag have neither influence on any internal state machine nor on command processing. The recommended way to find a reference position is to use the RunInit command. Alternatively one could initiate a long distance motion at very low speed using SetPosition and then poll ESW as frequently as possible to be able to stop the motion using HardStop right in the moment the switch position is reached. Then one would reset the internal position counters ActPos and TagPos using the ResetPosition command.

Q: What is the logic of the ESW flag?

A: The ESW flag reflects the state of the SWI input. ESW is set to one if SWI is high or low, i.e. pulled to VBAT or to GND. ESW is set to zero if SWI is left open, i.e. floating. ESW is updated synchronously with ActPos every 1024 µs.

Q: Is it possible to swap the logic of the ESW flag?

A: No, it's not. Actually this is not necessary since the ESW flag must be polled and evaluated by software anyway. The state of ESW has neither effect on any internal state machine nor on command processing.

Q: What else is important for the RunInit command?

A: The first target position of RunInit must be different from the current position before sending RunInit and the second target position must be different from the first one. Otherwise a deadlock situation can occur. During execution of RunInit only Get… commands should be sent to the device.

Q: Does the second motion of RunInit stop when the ESW flag changes, or does it continue into the mechanical stop?

A: Neither nor. The SWI input resp. the ESW flag have neither effect on any internal state machine nor on command processing, i.e. the RunInit command is not influenced by SWI / ESW. The same is true for the mechanical stop: as there isn't any internal stall detection the RunInit command can not detect a mechanical stop. When the mechanical stop is hit the first or second motion of RunInit (or the motion of any other motion command) will be continued until the internal position counter ActPos has reached the target position of this motion. This results in the motor bouncing against the mechanical stop and loosing steps. The intention of the second motion of RunInit is to drive out of the mechanical stop (reached by the first motion) to the desired reference position at a known distance from the mechanical stop or to drive slowly against the mechanical stop again to compensate for the bouncing of the first motion and to come to a standstill as close to the mechanical stop as possible.

Q: Does RunInit reset the position?

A: Yes, it does. After the second motion of RunInit has been finished the internal position counter ActPos is reset to zero.