# TMC236 / TMC239 TMC246 / TMC249

**FAQ and application note document**

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

## *TMC236 / TMC239 / TMC246 / TMC249 FAQ and application notes*

## *List of topics*

*(The term TMC236 here always refers to the complete driver family, unless otherwise noted)*

# 1. General problems when getting started

**Q: My TMC236 does not work properly at higher current settings. What can I do?**

A: This often is a problem with the TMC236 detecting a short circuit condition. Try to monitor this in the SPI serial word coming from the TMC236, e.g. using an oscilloscope on the TMC236's SDO line.

The typical causes for wrong detection of a short condition are:

1. The shunt resistor on the high side has got to long/thin PCB traces. These traces can easily raise the value by several 100mOhms.
- Use thick, short and straight traces.
- Make sure, that your sense resistor traces add no substantial resistance to the high side sense resistor.
- Quick workaround: Use lower value for high side shunt or add a voltage divider for VT.

2. During switching of the coil outputs of the TMC236 voltage spikes occur on the sense resistor. This can be caused by long/thin PCB traces from GND to the sense resistor or from the sense resistor to BRA/BRB. There especially should be no parasitic inductivities in these traces.
- Use thick, short and straight traces - a massive grounding is preferable.
- Make sure, that your sense resistor traces add no substantial resistance to the sense resistors.
- Quick workaround: Increase blank time or add RC-filtering for SRA and SRB.

**Q: What can be the problem when I observe reduced motor reliability at lower supply voltages?**

A: This can very well be coupled to the above points.
Generally the motor current in fact is independent of the supply voltage, and the TMC236 with its low internal resistance is a very good driver for lower voltages. But one has to consider, that a lower voltage means that the current needs a longer time to reach the target level. This means that at lower voltage level the maximum motor velocity may have to be reduced.
You can very well monitor this condition watching the open load flags of the TMC236: If they flicker during the motion, the motor velocity might be too high with regard to the actual supply voltage.

**Q: Do I have to make any changes to use the new A-type drivers?**

A: If you use the ENN pin for switching off the driver, make sure that the ENN pin is pulled up externally to at least 2.8V for circuits supplied with 5V. Do not use an open collector output to drive it.
Allow for a small increase in standby current on the VCC supply pins, because VCC feeds the ENN pin overvoltage protection comparator.

**Q: I have replaced a conventional (halfstep) driver by a TMC239. Now, my motor shows more resonances than before. What can be the reason?**

A: Please refer to the chapter on SPI mode programming, for halfstep patterns, if you want to stay with halfstepping. Even for halfstepping it is important to use the mixed decay feature, to get maximum performance.
Check list:

- Did you monitor the short circuit bits? Motor resonances can lead to high peak currents, which could trigger short detection, even if it is not triggered under normal circumstances.
- Are your step-pulses at equivalent distances? If you have got a problem in generating the step patterns, you might want to try the ultra-miniature, low cost, TMC428 SPI stepper motor controller.
- The remaining differences to conventional drivers are: Dramatically lower driver stage resistance – which also reduces resistive damping of the motor. To damp the motor, it is important to use the mixed decay feature properly, i.e. mixed decay switched on during falling slopes of the coil currents. In some cases continuous mixed decay improves performance.
- You could migrate from half-stepping to micro-stepping. This reduces motor resonances especially at low frequencies. At high rotation speeds, not every microstep has to be sent, but it is important to feed a jitter-free signal into the driver (motor), i.e. the microstep current should always correspond to a “snapshot” of the desired current at the time, where sending the telegram is initiated.
- Do you need high velocity? Please see the discussion on reaching maximum RPM.

**Q: The motor makes noise during stand still. What can I do to get it silent?**

A: These are some hints for conditions, which can (but don't need to) lead to audible chopper noise, even during stand-still of the motor:

- Low chopper frequency
- Use of mixed decay mode
- Low motor supply voltage
- High motor coil resistance
- Some microstep positions with very low motor currents on one coil
- Filter elements on SRA and SRB pin required, but not used
- Blank time too low / higher than necessary
- Bad PCB layout: long / thin / bend traces between GND and sense resistors or to BRA/BRB

So, for a given application, the following main points should be checked:

- Increase chopper frequency to 36 kHz.
- If resonable for your motor, it is advised to switch off mixed decay mode during stand-still. (Don't do this, if the exact microstep position is very important).

**Q: What are possible causes for a driver IC failing / becoming too hot?**

A: These are some hints for failure conditions, which can (but don't need to) damage the driver ICs:

- Motor Voltage above 34V (for non-A-types) / above 40V (A-types) or logic supply voltage above 8V
- Pulling motor connector during operation
- Short to GND without high side sense resistor
- Soldering problems (unconnected or shorted pins)
- Motor current set to more than 3A for extended periods of time
- Missing or far too small oscillator capacitor
- High ESD voltages applied to circuit before soldering it to board. We have never seen this happening, but please remark, that the drivers are sensitive to electrostatic (dis)charge, especially before soldering the IC onto the application board.
- Missing GND connection when powering the unit with a laboratory supply
- Missing Rslp resistor (or too high value)
- Missing short to GND resistor (or too high value)

All of these are severe violations of the drivers' specifications. In fact the devices are very rugged and may be operated well beyond spec without problem.

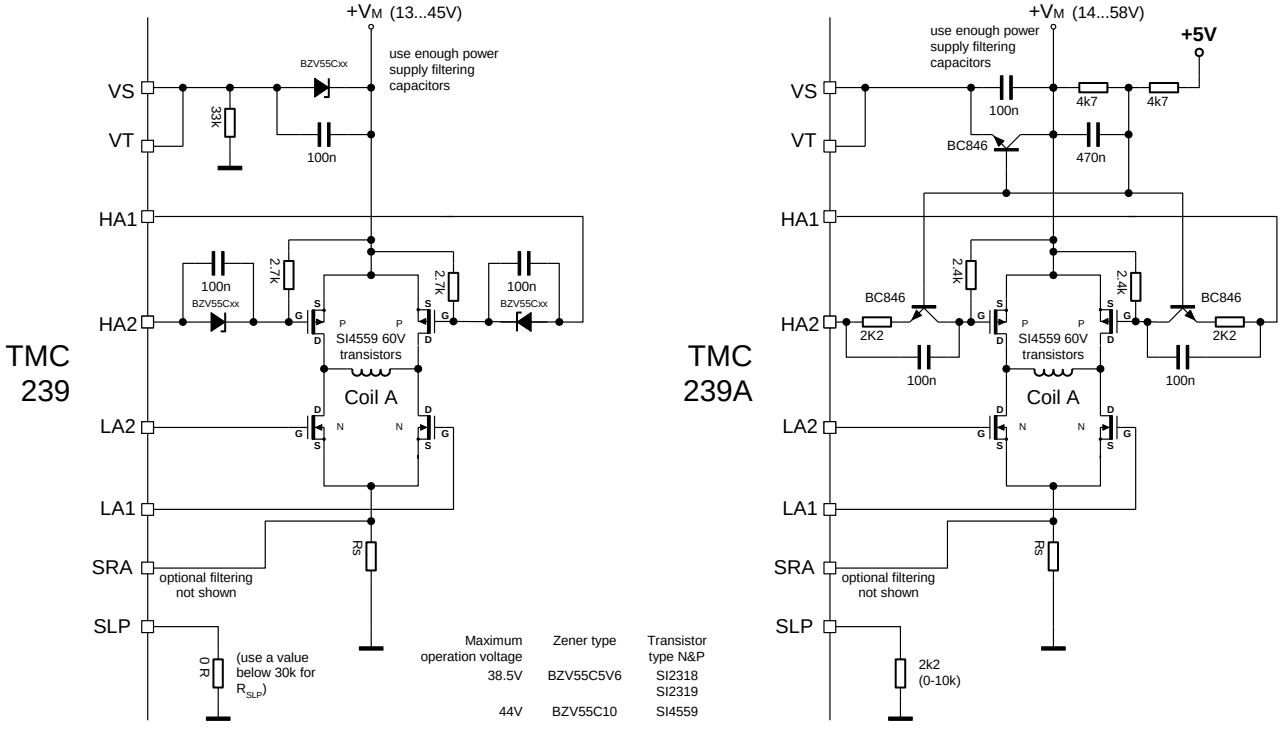
# 2. Higher voltage / higher current applications

### Q: Can the TMC236A / TMC 246A drivers operate at voltages > 34V?

A: For the TMC236A and TMC246A the maximum operation voltage is specified up to 36V motor voltage. Typically the drivers can work at up to 38.5V for long periods of time, but this should not be used in a design, as it might lead to accelerated ageing of the IC. Thus specification is for 34V, which takes into account typical stray of the power supply voltage of +-5%.

### Q: Can the TMC239 / TMC 249 drivers operate at voltages > 28.5V?

A: Yes, the TMC239 and TMC249 can be continuously operated at up to 34V (peak: 36V). There are suitable 40V and 60V P and N-channel transistors available, e.g. the 60V SI4559 (N+P pair). Using zener diodes, the maximum operation voltage of the driver circuit can be extended even beyond 50V (see schematics): The TMC239 supply voltage is reduced via a zener diode, while all P-channel transistor gates are shifted up by the same difference. For voltage differences above 10V, the transistor design (right schematic) should be preferred, because of temperature dependance of zener diode voltage. Please note, that the pull up resistor currents have to be driven by the gate driver outputs, thus setting an upper limit for the slope control resistor. The short to GND detector is not used in this application. The lower operation voltage limit is raised to 7V plus zener voltage. When using the pure zener diode based design, a careful thermal design is necessary – make sure that the zener diodes are well thermally coupled (i.e. placed near to each other). A capacitor bridges the power supply for high dU/dt, thus supply dU/dt has to be limited, especially for power on a power supply slope of maximum 10V/ms is suggested.




Solution for up to 44V supply voltage Solution for up to 58V supply voltage

*Schematic: Operating the TMC239 / TMC249 at up to 58Vpeak supply voltage*

**Q: I want to use the TMC239 with external high voltage, high current drivers for 100V; 4.5A. Where are critical design parameters?**

A: In the TMC239 data sheet you find a circuit using IR2101 power drivers. These drivers can be operated up to 500V. As a transistor you can use any high voltage MOS capable to drive the desired current, e.g. the IRF640.

1. Proper slope control is necessary: Too steep slopes can destroy the transistors because they lead to high currents.
   To control transistor slopes, you can use a gate-resistor for every transistor in the range of 50 to 470 Ohms. The design should result in slopes not faster than 50ns. To ensure a fast transistor switch-off, always use a diode like the 1N4148 to discharge the gate quickly, by bypassing the gate resistor (cathode to driver output, anode to transistor gate)!

2. Break-before-make delay is a very critical design parameter: The transistors of a half bridge are never to conduct at the same time, even not for very short times!
   The TMC239 can not monitor this, when you use external power drivers! It is a good idea to use a driver with dead-time control, like the National LM5102, which is very good for high current applications up to 100V! Program the dead time to get optimum break before make timing! For higher voltages, you can use IR2103 which generates an additional 500ns dead time, which works for most transistor configurations. Using a controlled dead-time driver simplifies the circuit, by eliminating the need for the load capacitors at the TMC239 outputs. But for the IR2103, which has an inverted low side input, you need to provide an additional inverter, e.g. a 2N7002 N-channel MOS with 1K pullup at its drain for the low side control input!

3. It is important to provide for a suitable charge pump circuit, which can stand the desired supply voltage. The additional charge pump sustaining circuit using the square wave oscillator can be comparatively weak, since it does not need to supply much current.

4. For higher currents, a massive ground plane and a compact layout (see datasheet and sample layout) is a MUST.

5. The blank time should be set to a value of at least 1.2µs, because switching spikes are delayed due to the additional drivers.

**Q: I have realized the circuit from the TMC239 / TMC249 datasheet, using external transistor drivers. My switching transistors die sometimes. What is wrong?**

A: (Also refer to the previous question.)

1. Monitor the break-before-make switching using a low capacity oscilloscope probe to measure at the power transistor gates. The transistors never should switch into short circuit. If necessary, increase break-before make time. To do this, you can use LM5102 drivers. These provide up to additional 600ns blank time.

2. Monitor the output slopes: They should be between 80ns and 300ns. Change the gate resistors, if necessary.

# 3. Optimizing motor performance

**Q: What is the optimum supply voltage for my application?**

A: The supply voltage and motor characteristics are related to each other: The supply voltage should always be at least 2 to 4 times higher than the voltage required for the motor to reach the specified coil current. However, it should not be higher than about 25 times this value; otherwise iron losses in the motor become quite high.

*Example*:
A motor is specified with 2.7 Ohms and 750mA coil current.
Thus the motor coil at nominal current has a voltage of 4.0 Ohm * 1.0 A = 4.0 V.
The driver supply voltage for best performance should be at least 2*4.0V = 8.0V, better 16V. A higher supply voltage gives a higher maximum motor velocity.

If the application's supply voltage is fixed or the voltage limit of the driver IC is reached, and you can not fulfil the above formula, or the motor torque at desired velocity is not sufficient, you should choose a motor with a winding designed for a higher current.

**Q: I use a low inductivity motor. The microstepping does not give equal steps, especially at higher supply voltage. How can I improve this?**

A: The problem with low inductivity motors at high supply voltage is, that even a very short chopper ON time leads to a high current flow. Thus, low currents, which are required for the microstepping coils, can hardly be reached. To get good microstep behaviour, switch on mixed decay mode continuously. To avoid audible chopper noise, raise the chopper frequency to 36 kHz. Since the TMC drivers automatically control the length of the mixed decay phase, this has no impact on motor dynamics.

**Q: My motor shows bad microstep behaviour, what can I do?**

A: Check if switching on the mixed decay mode constantly improves microstep behaviour. This often is necessary to get the motor smoothly running at more than 8 microsteps, because the driver can not bring the current to near-zero values, when mixed decay is not used (see next question). Use the new A-types: TMC236A, TMC239A, TMC246A, TMC249A. These have a more exact comparator threshold. Please see next question, also.

**Q: How can I improve the microstepping table to give more equidistant steps? How to eliminate the (smaller) step at the current zero crossing?**

A: Modifying the microstep table can improve motor behaviour a lot. Especially with mixed decay turned on, the motor resulting currents are a bit lower than the DAC control value. This typically results in a reduced zero crossing step, which will lead to motor resonance. As a work-around, try increasing the DAC values which are different from 0 slightly (e.g. by one) for your sine wave calculation.

*Example:* Sine wave with offset:

```
  unsigned char x;                          // table counter for sine wave
  float offset = 0.7;                       // example offset value
  unsigned char amplitude = 15;             // peak value
  unsigned char tablecount = 64;            // number off entries for full wave
  float f;
  int y;

  for (x=0; x < tablecount; x++) {
    f = sin((float)x * 2 * PI / tablecount) * (float(amplitude)-offset+0.49));
    if (y>0) f+=offset;
    if (y<0) f-=offset;
    y = round(f);
    …                                       // store result to table
  }
```

Depending on the motor characteristics, the optimum control wave for the motor will differ a bit from a sine wave. This can improve microstep equidistance, and thus motor noise, up to some degree.

**Q: My motor has a different sound in turn left and turn right. The zero crossing is different.**

A: Please refer to the next question concerning zero cross-over of the current.

**Q: How does the TMC236 driver handle the zero cross-over of the output current? Can it provide a true zero current?**

A: At zero current the coils do not become switched on when you use the SPI interface. With mixed decay switched off, the coils are completely in slow decay, and with mixed decay switched on, the coils are at 50% fast decay and 50% slow decay. This results in one important issue: Since mixed decay only can quickly reduce coil current with the right polarity setting, the polarity change in a microstep drive should always follow the zero current instead of preceding it! This should be remembered, when programming a sequencer for the TMC236, otherwise different motor performance / motor noise in both motor directions can result.

The integrated automatic mixed decay allows a very precise current control even at low currents. When mixed decay is switched off, low current performance is limited: The minimum reachable motor current is proportional to (blank time) * (supply voltage) / (motor resistance).

**Q: What is the transfer function of the driver?**

A: It has a linear transfer function for the DACs, because this is most universal.

# 4. Getting maximum velocity / RPM from your drive

**Q: My application needs very high motor velocities. How can I get the maximum RPM?**

*Comment: The TMC236 family and the controller TMC428 are optimized for motor operation within the nominal velocity range, where the motor current can reach the programmed target current. At higher velocities, the motor torque drops (please see manufacturer data), but it is still possible to use a given motor in this range. However, you should understand that motor operation in the high velocity range, where torque drops to a value below about 60%, is not a desirable operation area. In fact, it makes sense to adapt the application (e.g. gearing) or the motor to operate in its nominal area. Adapting makes especially sense, because the need for operation outside the nominal area means, that you could work with a smaller, less expensive motor, having less torque, but more RPM, and you get a more stable system!*

A: There are two points which allow the optimization of the maximum motor velocity: The current wave used by the motor controller and the driver chopper frequency.

**1. Full stepping is a good solution for high RPM applications!**
Why?
In general, RPM is just limited by the inductivity of the motor and the available supply voltage. So the answer is, use a motor with low inductivity, i.e. high coil current, and maximize supply voltage.
Microstepping does not have a major influence on motor performance at high RPM. As compared to microstepping, fullstepping control allows to get maximum current into the coils. So, when exceeding some velocity, you might reprogram your microstepping table and change into fullstepping. When you use the TMC428, you can realize this on-the-fly, by modifying the driver configuration table to all phase bits constant, or by modifying the sine wave table to a full-stepping table. Additionally, mixed decay should be off. Please be aware, that only one motor attached to the TMC428 should be running in this case. If you have multiple motors attached to the TMC428, which shall run at the same time, it might be better to stay with microstepping.

An important DO NOT for reaching high RPM: Do not set the motor current too high. To keep motor resonance low, it is important that the programmed maximum current can be reached by the application. So you will typically reach a higher velocity, with a lower current setting, when you are at the limits of the motor / supply voltage. To be sure: Check with the oscilloscope at the sense resistor, that the motor driver starts chopping for at least a small part of each (full) step.

**2. Modify the chopper clock to give best possible current regulation!**
*Background:* A stepper motor can be operated outside its normal range at a velocity, where the coil current cannot reach the pre-set nominal motor current. Thus, the motor current wave form does not follow the desired fullstep or microstep pattern any more. At these high motor velocities, the driver can not dampen oscillations of the motor sufficiently, because the driver is on for most of the cycle. Thus motor resonances and oscillations can build up and cause a motor stall. Possible causes for this are:

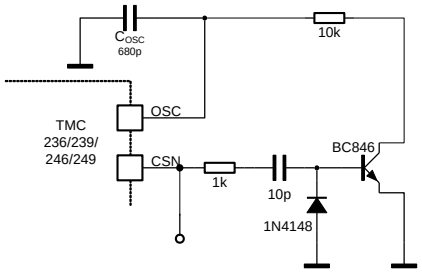
- Limited update rate of the fullstep / sine wave (also refer to 1.)
- A beat between chopper frequency and (full)step frequency

*Solution A:* Avoid the occurrence of any beats in your controller: Update the current wave synchronously to the step rate! The TMC454 generates the chopper clock this way, allowing very high motor velocities. When using the TMC428 controller, use one TMC428 per motor, or bring other motors on the same driver chain to stand still, when one motor has to run at high

velocities. The updating of the other drivers in the chain causes beats between the different update frequencies caused by different step frequencies. While this is only a very small timing difference, and it does not harm at low and medium motor velocity, it may hinder the motor from reaching very high RPM.

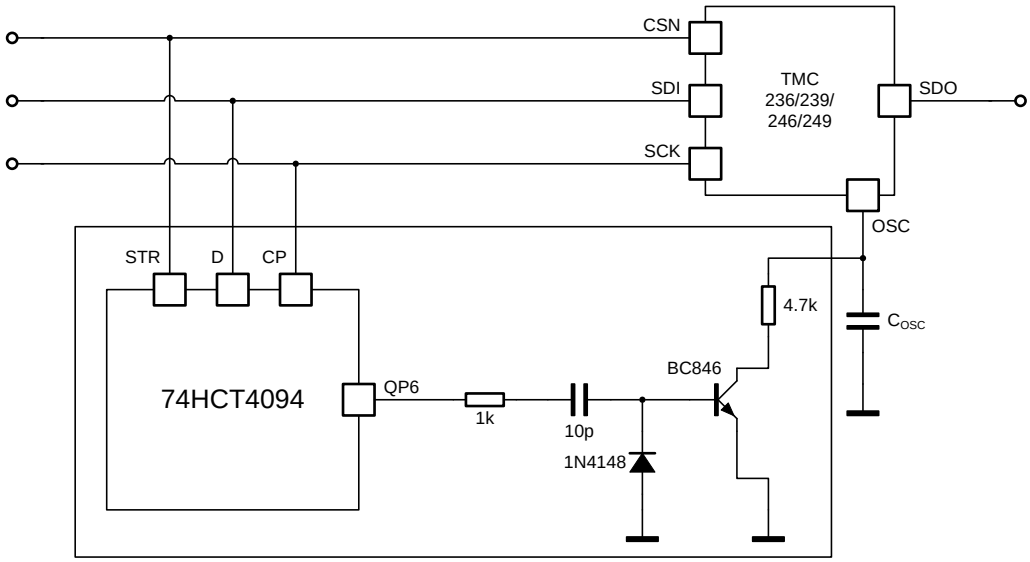
***Solution B:* Synchronize the chopper clock to the steps using TRINAMICs ChopSync™!**
*The following method brings the motor to the highest possible RPM!*

A beat between the step frequency and the chopper frequency may disturb motor operation at velocity ratings above approx. 600 RPM. To avoid this, the application can benefit either of an increased chopper frequency, or the beat can be completely avoided, by generating the chopper clock in a way, that it does not interfere with the fullstep frequency. In order to have a stable chopper cycle sequence for each step, the chopper timer becomes restarted with each step. It can be restarted by discharging the capacitor connected to the OSC pin. This can be achieved by a small circuit, that pulls the OSC pin (and the capacitor) to ground on each rising edge of the chip select signal (CSN pin). A simple circuit can be built as follows:




This circuit works quite well for half- and fullstepping, when only one driver is connected to the SPI bus. It is based on the idea, that each SPI datagram sent to the driver causes a pulse on the CSN line. With each pulse the OSC pin becomes pulled to the lower threshold, causing the oscillator to be synchronized to the CSN pulses. As long, as the SPI frequency is far enough below the chopper clock this works well. For example, up to 6000 RPM can be reached with a standard 1A motor at 24V.

With a microstepping control, this circuit may suffer from too many updates of the SPI chain. Therefore, the following circuit directly accesses one phase bit of the driver:




The circuit now avoids occurrence of a beat between the oscillator frequency and the step frequency, but care has to be taken, that the update rate of the SPI chain is high enough to avoid a beat between the updates and the step frequency. Therefore, the microstep frequency now is limited by the SPI frequency and the length of the SPI telegrams. All telegrams must be

completely transmitted for the driver to make the motor do a step – skipping of datagrams will lead to a jitter of the phase polarity control. The use of only one driver instead of a driver chain ensures the shortest possible telegram length. For example, a TMC428 with only a single driver allows an update rate of about 60 kHz. This means, that a 200 step motor can reach up to 60kHz/16µstep/200steps*60s=1125RPM at a setting of 16 microsteps, 2250 RPM at an 8 microstep setting.
A digital implementation of ChopSync is found in TMC454 and TMC457 ICs! With these, the above circuitry is not required!

*Solution C:*
Interference between the chopper clock and the fullstep frequency can also be reduced or avoided by using a random chopper clock. To generate a random chopper clock, you can use a simple 16 bit CRC generator, and add a random word to your chopper clock timer. The modification rate of the chopper clock should be identical or higher than ¼ of the maximum desired motor full step frequency. At 2000 RPM, 2 kHz random rate are sufficient. The following program extract shows a solution for an Atmel 90CAN128 processor.

```
volatile UINT RandomRegister=1;                    // Attention, do not initialize to zero

SIGNAL(SIG_OVERFLOW0)                              // This procedure is called with 2kHz
{                                                  // It generates a frequency of 31 to 42kHz at
  UCHAR x;                                         //  timer 3 output pin

  x=BYTE1(RandomRegister) & 0x81;                  // Calculate CRC16 random word
  if(x==0x80 || x==0x01)
    RandomRegister=RandomRegister << 1;
  else
    RandomRegister=(RandomRegister << 1)+1;

  OCR3A=(BYTE0(RandomRegister) & 0x3F) + 191;      // Add to Timer frequency word
  if(OCR3A<TCNT3)                                  // Check for timer 3 overflow
  {
    TCCR3C=0x80;
    TCNT3=0x0000;
  }

  TCNT0=224;  //ungefähr 2kHz (bei Vorteiler 256)
}
```

However, the ChopSync™ based solution proves to be much more reliable and does not generate additional audible noise, like a random clock could do.

# 5. Increasing microstep resolution

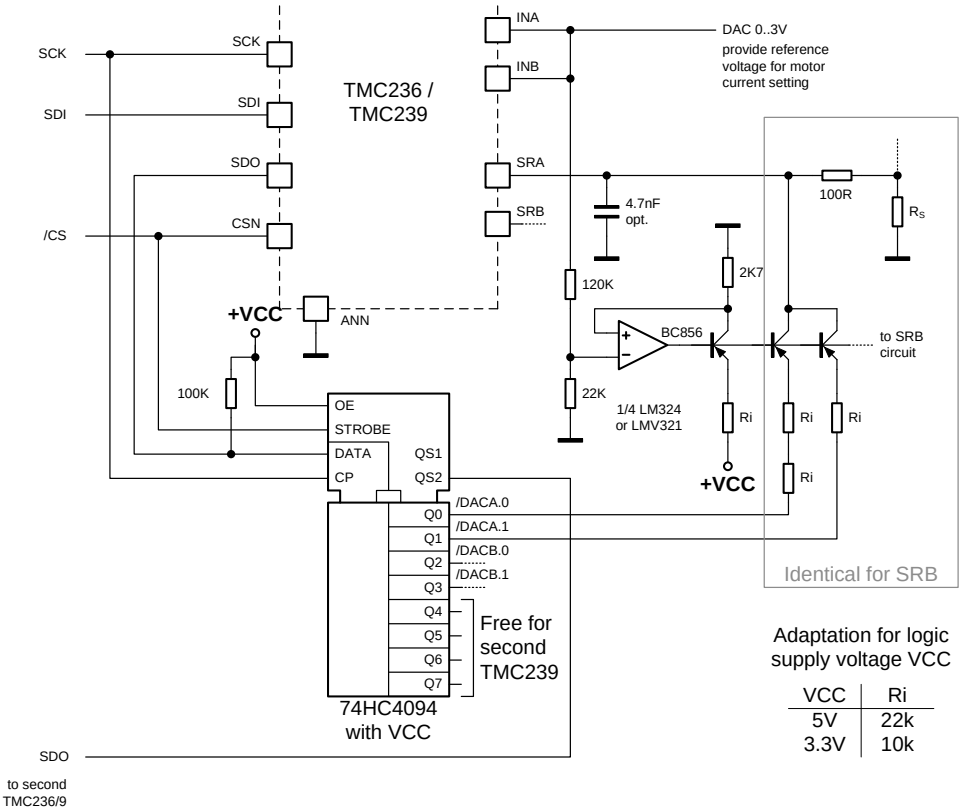
**Q: The TMC236 has 4-bit DACs for 1/16 microstepping, how specifically do you implement an interface for higher resolution?**

A: The TMC236 can realize more than 16 microsteps via TMC428 SPI control: Just program the TMC428 for 32 microstep mode. Due to the combination of two DACs driving the two coils, this results in a resolution somewhere between 20 and 30 microsteps.

The TMC457 directly allows driving the TMC236 with 64 microsteps or even more, using its analog interface.

To get full 64 microsteps using the TMC428, refer to the schematic example in the TMC236 data sheet (Extending the microstep resolution). Please remark, that the **lower two bits are inverted**, and the **values from 0 to 3 give a zero current**. This effectively results in a 60 level current resolution. A suitable microstep table is printed below. The effect of this modified DAC behaviour is, that the TMC428 ramp-phase-dependent current scaling function does not lead to a good result and should not be used! This could be improved by inverting the additional DAC-Bits. Please be aware, that the 4094 cannot be included in SPI busses with multiple /CS lines.

A universal solution based on this, while giving a 64 microstep resolution at any current setting is shown in the attached schematic. It allows the use of an additional microcontroller generated current reference signal. The OPAMP controls a number of four switchable constant current sources, which add two times two bits of DAC resolution to the drivers' internal DACs. By supplying 0 to 3V to the INA/INB input, the motor current can be controlled in a wide range.




*Schematics: 64 bit microstepping with scalable motor current via INA / INB*

For best microstep performance run the motors with mixed decay switched on continuously and 36 kHz chopper.

Microstep table for the printed schematic with inverted LSBs (1/4 wave, like in TMC428):

```
0x00,0x06,0x05,0x04,0x0a,0x09,0x0f,0x0e,0x0c,0x13,0x11,0x10,0x17,0x15,0x14,0x1a,
0x19,0x18,0x1e,0x1d,0x1c,0x22,0x21,0x20,0x27,0x25,0x24,0x2b,0x2a,0x29,0x28,0x2f,
0x2e,0x2d,0x2c,0x33,0x32,0x31,0x30,0x37,0x36,0x35,0x35,0x34,0x3b,0x3a,0x3a,0x39,
0x39,0x38,0x38,0x3f,0x3f,0x3e,0x3e,0x3d,0x3d,0x3d,0x3d,0x3d,0x3c,0x3c,0x3c,0x3c
```

Sample configuration for TMC428 using one driver (TMC428 LSMD=0). The unused 4 bits are included in the datagram:

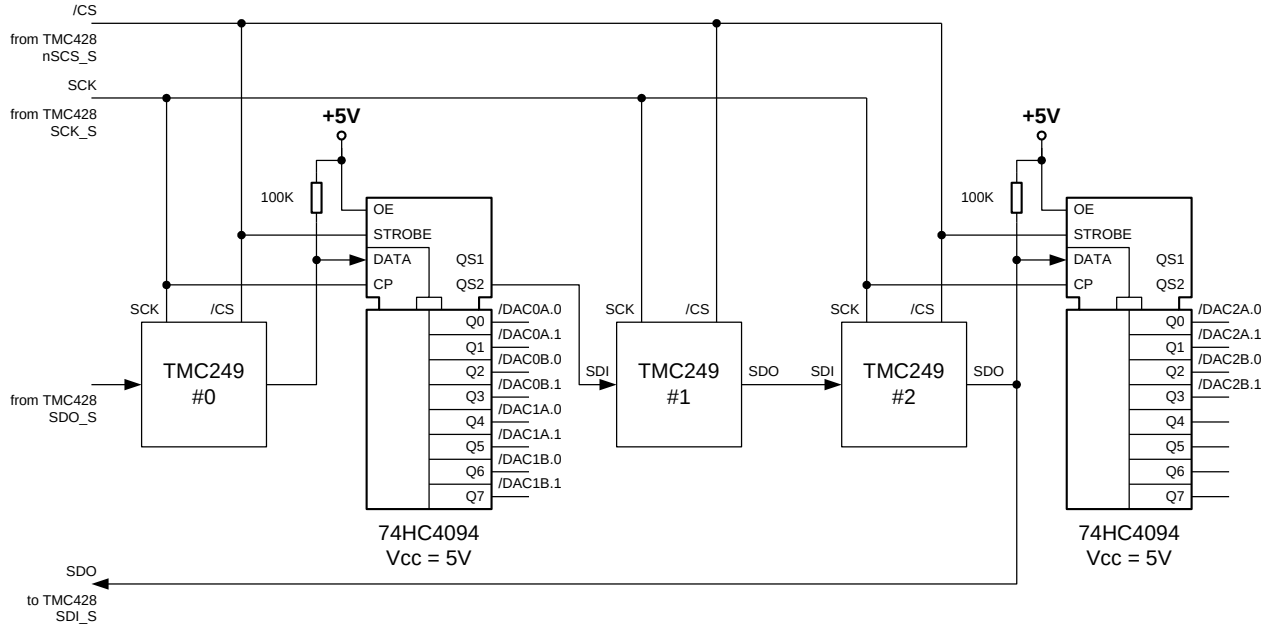
```
0x09,0x08,0x01,0x00,0x11,0x05,0x04,0x03,0x02,0x06,0x11,0x0d,0x0c,0x0b,0x0a,0x2e
```

Sample configuration for TMC428 using two drivers (TMC428 LSMD=1) with one shared shift register, which is located in the middle of the SPI chain:

```
0x11,0x05,0x04,0x03,0x02,0x06,0x11,0x0d,0x0c,0x0b,0x0a,0x0e,0x09,0x08,0x01,0x20,
0x09,0x08,0x01,0x00,0x11,0x05,0x04,0x03,0x02,0x06,0x11,0x0d,0x0c,0x0b,0x0a,0x2e
```

The red entries contain the switch to the next motor bit (NXM).

To attach three drivers with 64 microsteps, just append the configuration table for two drivers to the one for a single driver. The single driver and its shift register are appended to the SPI chain (see schematics). In this case, the complete SPI chain length used is 12+8+12+12+4 = 48 bits, plus four unused bits. The four MSBs of the last shift register remain unused. To get access to the driver error flags, connect the SDO of the last driver in the chain both to the last shift register and to the TMC428 driver chain SPI input. Thus, the TMC428 datagram low and high word registers just see 12+8+12+12 = 44 bits (instead of 52 bits, which would not fit).



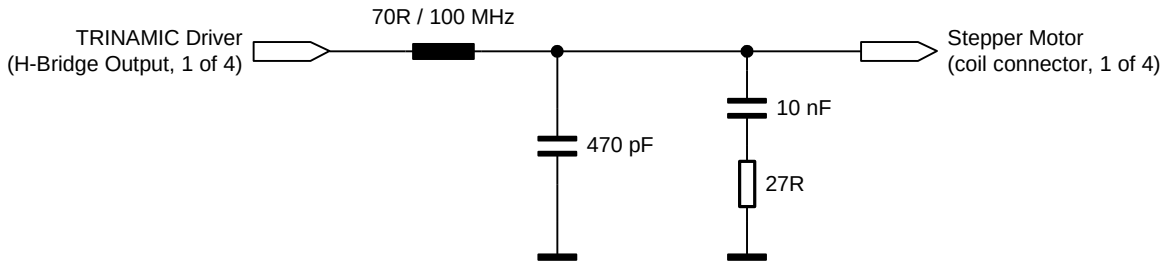

*Schematics: SPI chain with three drivers with 64 microsteps attached to TMC428*

*Attention: While the microstep extension solution based on a 74HC595 shift register as shown in the TMC236 datasheet works with all SPI configurations, the 74HC4094 based solution does not work with multiple CS lines. This is due to the 74HC4094 not having an edge triggered STROBE input.*

# 6. EMC considerations

**Q: I want to add EMC circuitry to my design. Can you give some hints?**

A: Since the TMC239A / TMC249A can support high motor currents, a slow slope setting can not satisfy the need to keep power dissipation low. Further, spikes from switching regulators etc. might still find a path into the motor cables. A LC filtering with dampening by an RC filter for the motor supply lines eliminates these effects, as shown in the sample schematic.



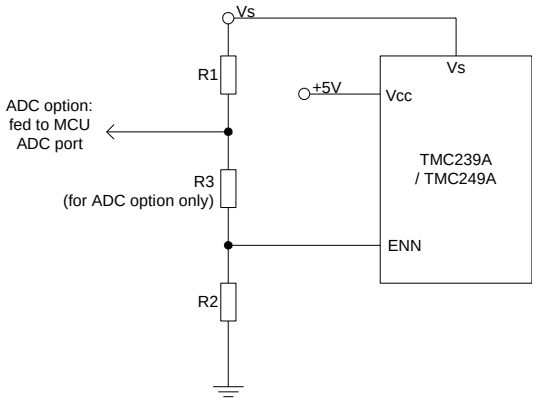

*Schematics: Optional EMC filtering for motor outputs*

# 7. Overvoltage protection

**Q: How can I realize motor supply overvoltage protection for the TMC239A / TMC249A?**

A: Since the TMC239A / TMC249A can be used with MOSFETs, having a lower voltage rating than the maximum power supply voltage of the TMC239A / TMC249A itself, an overvoltage protection may be valuable feature. By switching off the power MOSFETs, the motor outputs are left floating, and thus, the transistor output stage can stand up to two times the transistor $V_{DS}$ voltage. When using 30V MOSFETs, the power stage thus is able to withstand the maximum voltage rating of the TMC239A / TMC249A (40V) while being switched off.

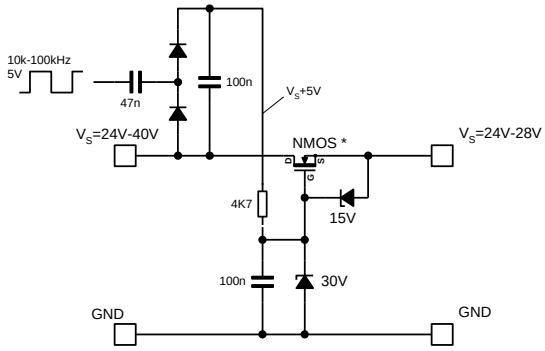
The ENN input of the TMC239A / TMC249A has a dedicated switching threshold of 2.5V reducing the external circuitry to a simple voltage divider as depicted below:




The voltage divider should be dimensioned such that the ENN switching threshold of 2.5V is latest reached at a $V_S$ level of about 30V. This will protect 30V type power MOSFETs attached to the TMC239A / TMC249A. If using a microcontroller's ADC port to measure $V_S$ an additional resistor R3 must be used if the desired ADC voltage range differs from the ENN pin voltage divider range. In this case the extra cost for hardware overvoltage protection is just one resistor. If the ADC input level requirement meets the 2.5V threshold voltage of the driver, then R3 can be omitted and actually no extra parts for hardware overvoltage protection are needed. In this case the ENN input can be fed by the same voltage (and thus use the same voltage divider) as the ADC port does.

**Q: I use an unregulated power supply with an output voltage of 24V. Under low load conditions the voltage may raise above 30V, but the drivers are not allowed to be switched off under these conditions. Is there a simple solution for multiple drivers?**

A: The circuit depicted below is a voltage regulator, which operates at very low drop, if the input voltage is below or equal to 27V. As soon as the input voltage raises, the output is stabilized to the Zener voltage minus about 2 to 4 volts, depending on the NMOS threshold voltage (adapt zener voltage). Choose the NMOS according to the maximum desired current for continuous operation. However, the package should allow for dissipation of the differential voltage for the length of the overvoltage condition, when taking into account the maximum current flowing in this condition. The charge pump clock can identical to the chopper clock for the stepper drivers, using an adequate buffer circuit. If using a 74HC gate, provide an extra protection diode and series resistor to prevent capacitor loading current from destroying the output. The gate protection zener can be eliminated by using a protected MOSFET.




You also might use 40 or 60V MOSFETs, when working with TMC239A or TMC249A.

# 8. Using the SPI interface

### Q: Does the TMC236 update the SPI datagram at once after the rising CSN edge?

A: Yes. You can use this to synchronize the microstep timing via the CSN edge. This allows sending the SPI datagrams without strict real time requirements, while keeping the possibility to synchronously update the driver(s) with their step rate!

### Q: My processor has an 8 bit SPI interface. How can I send 12 bit wide words?

A: You do not need to send 12 bit words. You can send 16 bit wide SPI telegrams. The first 4 bits are just shifted through the TMC236 and you can treat them as don't care. When you send a 16 bit word to the SPI chain, this means that the four upper bits in the processor are don't care, the next 4 bits in this byte represent: MDA, CA3, CA2, CA1. The lower byte represents the SPI bits CA0, PHA, MDB, CB3, CB2, CB1, CB0, PHB.

As most microcontrollers transfer SPI data in a byte-oriented manner (i.e. in units of 8 bits) two successive bytes have to be sent (SDO) and received (SDI). The first four bits sent are don't care, the following 12 bits are interpreted as depicted in the datasheet. The first 12 bits received have the meaning as depicted in the datasheet, the last four bits are don't care. CSN must **not** go high in between the two bytes.

The mixed-decay control bits MDA and MDB can be set active ("1") while the respective phase current is decreasing to allow for faster sinusoidal current waveforms. However, MDA and MDB have to be switched off ("0") before zero crossing of the respective phase current for good load detection results (bits LD2…LD0).

PHA and PHB have to toggled exactly during zero crossing of the respective phase current for smooth current waveforms (see "**Fehler! Verweisquelle konnte nicht gefunden werden.**").

### Q: How to cascade multiple TMC246/249 in an SPI chain?

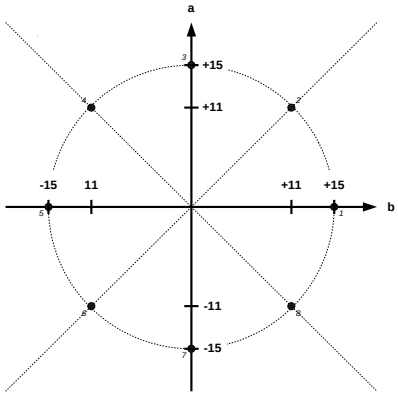
A: It is possible to daisy-chain two or even more TMC246/249 drivers in the SDO-SDI path of the microcontroller. Simply connect SDO of the microcontroller to SDI of the first TMC246/249, SDO of the first TMC246/249 to SDI of the second TMC246/249 and so on. SDO of the last TMC246/249 has to be connected to SDI of the microcontroller. SCK and CSN respectively have to be connected in parallel to all TMC246/249 drivers.

The SPI datagrams for all daisy-chained TMC246/249 simply have to be concatenated **without** any de-activation of CSN in between. The first 12 bits sent and received belong to the last TMC246/249 in the chain whose SDO is connected to SDI of the microcontroller. The last 12 bits sent and received belong to the first TMC246/249 whose SDI is connected to SDO of the microcontroller.

If, for example, two TMC246/249 drivers are daisy-chained, the SPI datagram transfer would comprise 24 bits, i.e. three bytes in total.

**Q: How can I use the TMC236/TMC239 in a half stepping application?**

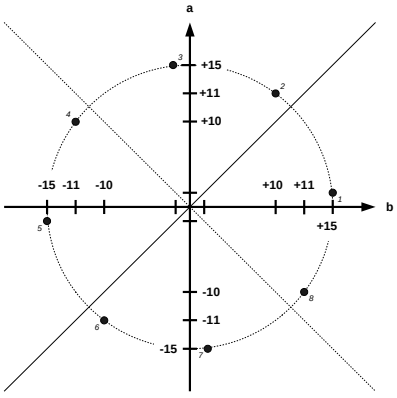
A: In principle, you can drive the coils using a standard half-stepping scheme, as depicted in the left figure. It equals a two microstep sine wave. However, this scheme does not make use of the mixed decay feature, which can dramatically improve performance, even in half stepping applications. To use mixed decay, the modified scheme rotates the fullstep / halfstep positions by some amount, always leading to values different from zero. We can also realize a symmetrical layout of the patterns, which has got an advantage concerning the distribution of power dissipation in driver and motor.




a = +11, +15, +11, 0, -11, -15, -11, 0

b = +11, 0, -11, -15, -11, 0, +11, +15

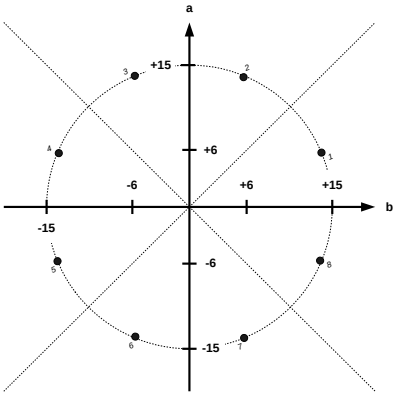
standard




a = +1, +11, +15, +10, -1, -11, -15, -10

b = +15, +10, -1, -11, -15, -10, +1, +11

modified




a = +6, +14, +14, +6, -6, -14, -14, -6

b = +14, +6, -6, -14, -14, -6, +6, +14

modified symmetric

| | a | b | MDA | CA3 | CA2 | CA1 | CA0 | PHA | MDB | CB3 | CB2 | CB1 | CB0 | PHB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | +1 | +15 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| **2** | +11 | +10 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| **3** | +15 | -1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| **4** | +10 | -11 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| **5** | -1 | -15 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| **6** | -11 | -10 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| **7** | -15 | +1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| **8** | -10 | +11 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| | **a** | **b** | **MDA** | **CA3** | **CA2** | **CA1** | **CA0** | **PHA** | **MDB** | **CB3** | **CB2** | **CB1** | **CB0** | **PHB** |
| **8** | -10 | +11 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| **7** | -15 | +1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| **6** | -11 | -10 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| **5** | -1 | -15 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| **4** | +10 | -11 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| **3** | +15 | -1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| **2** | +11 | +10 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| **1** | +1 | +15 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

Improved halfstepping pattern, CW and CCW direction

| | a | b | MDA | CA3 | CA2 | CA1 | CA0 | PHA | MDB | CB3 | CB2 | CB1 | CB0 | PHB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | +6 | +14 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| **2** | +14 | +6 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| **3** | +14 | -6 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| **4** | +6 | -14 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| **5** | -6 | -14 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| **6** | -14 | -6 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| **7** | -14 | +6 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| **8** | -6 | +14 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| | **a** | **b** | **MDA** | **CA3** | **CA2** | **CA1** | **CA0** | **PHA** | **MDB** | **CB3** | **CB2** | **CB1** | **CB0** | **PHB** |
| **8** | -6 | +14 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| **7** | -14 | +6 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| **6** | -14 | -6 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| **5** | -6 | -14 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| **4** | +6 | -14 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| **3** | +14 | -6 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| **2** | +14 | +6 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| **1** | +6 | +14 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |

Improved symmetrical halfstepping pattern, CW and CCW direction

The tables show the resulting SPI datagrams. The mixed-decay-bits (**MDA** resp. **MDB**) are valid for the respective rotation directions 1,2,3,4,5,6,7,8 (CCW) and 8,7,6,5,4,3,2,1 (CW). The split tables for CW and CCW are necessary to ensure mixed decay flags to be set during phases of decreasing phase current with unchanged current direction. When the motor is at rest, the mixed decay bits (**MDA** and **MDB**) should be set to '0', reducing chopper noise and power dissipation.

Concerning readout of the open load bits, these two schemes also improve the result, because the coils are never switched off. However, they should be read out during standstill, e.g. after each motor movement, since the large current steps in halfstepping could lead to wrong results of the open load detection during motor motion.

You still can take advantage of the current control feature in halfstepping, by reducing current (torque) depending on the motor action, e.g. a reduction during stand still.

# 9. Using stallGuard™

**Q: When and how often should I read out the StallGuard value?**

A: The stallGuard is updated once per fullstep, so it is not necessary to readout more often than the actual fullstep rate. In order to avoid reading out a wrong value, be sure **not** to read out the value directly after initiating a new fullstep: Do not use data from SPI datagrams directly following the SPI datagram which has initiated the next fullstep, i.e. which has toggled PHA or PHB bits. The TMC246 / TMC249 updates the stall information within 1 chopper period plus 8 microseconds after toggling of the PHA / PHB bits. A safe point of time to read out the StallGuard bits is for example the SPI datagram, which initiates a phase toggle, or the one preceding it (in microstep applications).

**Q: How do I read out the StallGuard value in a system using a TMC428?**

A: The StallGuard values are read using the TMC428 registers datagram_low_word and datagram_high_word. An update of these registers is forced by writing to one of the registers and polling the CDGW bit, which becomes cleared as soon as the update has happened. In order to detect a stall within the fullstep period where it has occurred, trigger the read out at least once per fullstep. Then isolate the corresponding bits and compare them to a threshold value, which has been determined for the application. However, the StallGuard values might have been read by the TMC428 in a phase, where the values were invalid. To avoid erroneous detection of a stall, trigger the read out procedure a second time, directly after detection of a stall level below your threshold level. Only if the second read out still signals a stall condition, a valid stall has been detected.

**Q: I get sporadic wrong stall detection.**

A: Most probably, your application sometimes reads out the StallGuard value data during the update period inside the driver. At this time, you might see levels which are wrong. Please see the comments on when to read out the StallGuard value.

**Q: How should the stallGuard readout level differ, when I increase mechanical load on the motor?**

A: The value read back from the SPI telegram decreases when the load increases. However, when you use the evaluation board software, the displays in the Windows software show "seven minus value read back", just to make the effect better visible (the visibility in a graph is better when you see a high value when you are having high load). In TMCL, also a high value means high load.

**Q: When I reduce motor current via INA / INB base current setting, why does load measurement give less reliable results?**

A: The load measurement accuracy decreases, when you reduce the motor current by applying an external voltage reference. To make the best use of load measurement, use the internal voltage reference, and adapt the sense resistors set the maximum current to the desired value.

However, if the external voltage reference is used and it is near to 2V, the quality of the load measurement is not influenced.
This is due to the fact, that the eight stall guard levels are referenced to a fixed internal reference of the TMC246/TMC249. So, if you reduce / increase the motor current, e.g. via the current control via INA/INB, or by digitally modifying the motor current, the levels become shifted by the same amount.

**Q: I do not see usable StallGuard readouts. There is no reliable correlation to the load on the motor.**

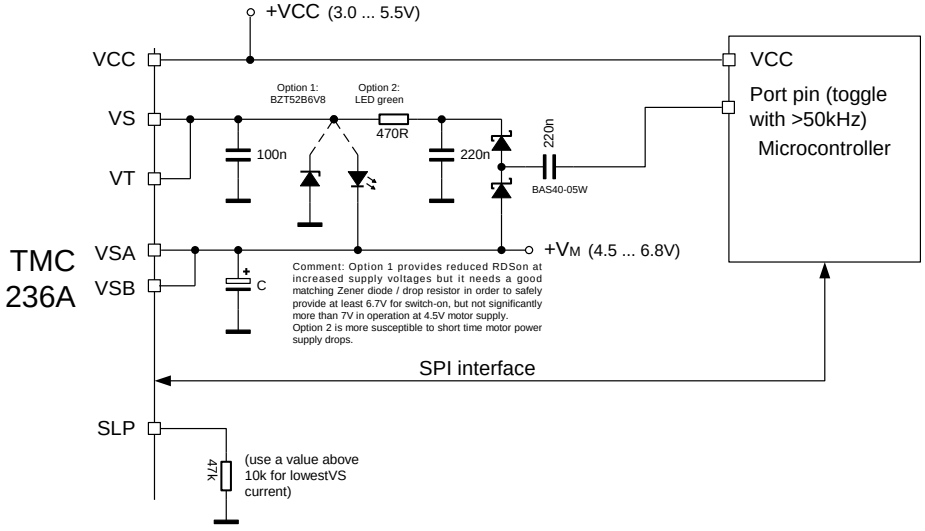
A: The load measurement sees the actual load on the motor, including the load caused by a resonance in the motor itself. So, load measurement results are dependent not only on the external mechanical load, but also on everything that influences the systems resonances: Motor current setting (at a too low current resonance is much higher!), motor velocity and dynamic motor load (flywheel mass also damps or changes resonances). For example, when you see the load indicator increasing, when you load the motor (and it should decrease) this can be, because your motor was in resonance, and you have damped the resonance with your fingers, which in fact reduces (dynamic) motor load! This is somehow tricky.
To get a reliable stall detection for mechanical reference, you should operate the motor at the specified current (by choosing fitting sense resistors) - choose a velocity where resonance is low and the load indicator shows a high value. You might want to do some software filtering, to get more stable values, e.g. find the minimum reading of a few samples of the load indicator for a number of sequential full steps.

# 10. Low voltage operation with TMC236 / 246

**Q: Can I operate the TMC236 with a single 5V supply or with four battery cells?**

A: Yes, this can be done with a simple trick, using an inexpensive low current (1.5mA) charge pump.
*Background:* The TMC236 family has got an undervoltage protection, which disables the circuit below 5.9 supply voltage. To power up the driver, a voltage of minimum 6.7V has to be applied.
*Solution:* The TMC236 VS supply normally is operated with the same supply voltage as the power bridges (VSA and VSB). However, there is no need to stick to this. The bridge voltage may be lower than the supply voltage. This enables us to use a low current 6.8V supply for the TMC236 which ensures power-on of the undervoltage detection, while the motor bridges directly work with the battery voltage. The bridge supply voltage can be up to 2 to 3V below the Vs supply, giving the possibility to operate down to 4.3V or so. Please see the schematic for a suitable circuit. This solution has got the advantage, that the low-side bridge FETs still have their extremely low RDSon, because they still have the full drive voltage. At the same time, the high side FETs are driven with a voltage reduced by the difference between VS and VSA. Since the P-channel FETs used in the TMC236 start conducting at about 2V while they are driven with 6V, they operate, but they have a slightly increased RDSon, which increases as the difference between VS and VSA / VSB goes up. The increase in the RDSon of the high side FETs has got less impact, because most time they are switched off during normal operation.

With a supply voltage of 5V to 6V the driver can still supply the full current without a major increase in thermal dissipation. Since the TMC236 consumes only about 1.5mA at VS, a simple charge pump realized with a microcontroller pin and a dual Schottky diode can supply the required current. The charge pump can be turned off in stand-by periods to save energy. When a RS232 interface circuit like the MAX202 is present, its integrated charge pump will provide enough energy to supply the TMC236 VS. In this case, just an additional 6.8V regulator consisting of a resistor and a zener diode is required.




Alternatively, to also get a higher maximum velocity for the motor, you can use any step up converter able to supply enough current for the stepper motor and supply the whole circuit with an increased voltage. However, this solution adds cost for the step up converter, while the solution depicted above is nearly cost-neutral.

# 11. Driving DC Motors with TMC236 / 239

**Q: How do I get the TMC236 / TMC239 to generate a DC motor PWM?**

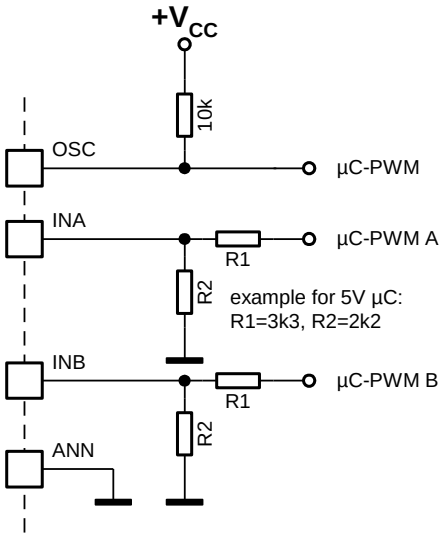
A: The TMC236 uses a constant current chopper principle, which is optimum for driving stepper motors. When operating DC motors with constant current, they provide a roughly constant torque. This in other words means that velocity is very dependent upon mechanical load on the motor. Normally a velocity control is desired, with a torque limit as an add on. The TMC236 can do both! All of its advanced protection and diagnostic features can be used. One TMC236 can control two DC motors!

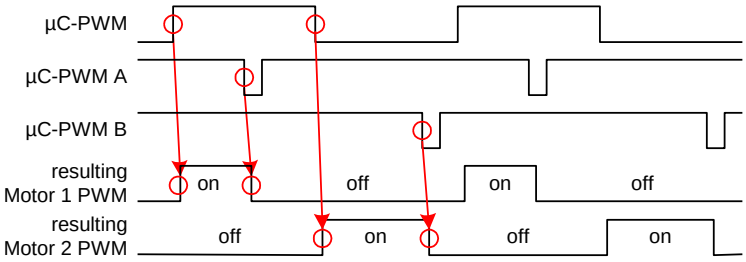
Here are two possibilities for DC motor driving:

1. This solution uses the TMC236 in stand alone mode (SPE tied to GND). The CPU provides a PWM to the TMC236 phase polarity signals (PHA for motor 1 and PHB for motor 2). The PWM frequency is in the range of some kilo Hertz up to a few ten kHz. This PWM directly controls direction and velocity of the motor: A 50% PWM is required for motor stand still (alternatively, disable the driver). A PWM value in the range 50% to 100% makes it turn right, a lower value for turn left. The motor torque can be controlled by providing an analog voltage to INA / INB. The TMC236 integrated chopper only becomes active, when the motor draws too much current. Then it provides for torque control / limit.

   This chopper schemes gives a good motor stiffness and is good for precise positioning control, but it has a higher power dissipation because the chopper is always active in both directions. Thus, the driver supply voltage should not be much higher than the motors' rated voltage.

2. This solution uses the SPI interface of the TMC236 and allows full access to all of its features (see schematic). The CPU provides three PWM signals to the TMC236: It directly controls the chopper clock (OSC) and uses INA / INB to provide a PWM signal for motor 1 and motor 2 (please keep in mind that the voltage for INA / INB should not be higher than 3V). The chopper clock can be in a range of a few kilo Hertz to a few ten kHz.




The waves show motor 1 PWM operating with 25% duty cycle and motor 2 PWM operating with 38% duty cycle.

Here, the PWM signal for motor 1 is related to the rising edge of the OSC pin (see timing diagram), while the PWM signal for motor 2 is related to the falling edge. The motor chopper

is on, beginning from its related OSC edge until the processor switches the related INA / INB input to low. Required low time for INA / INB is 3µs or more. INA / INB shall return to high latest at the next related OSC edge. The resulting chopper on time controls the motor velocity. Please never let the OSC pin PWM output tri-stated. Provide a 10K pullup if this can not be guaranteed.

The motor torque is controlled by the current setting. The current calculation is identical as for a stepper motor: It is given via the digital current control bits in combination with the sense resistors and the INA / INB voltage divider control. You can try with slow or mixed decay setting for this mode. The motor direction is controlled by the SPI PHA / PHB bits. The motor can be stopped by programming the related DAC bits to zero. The three required PWM signals can be generated via a microcontroller with capture / compare unit, or using a single shot timer and interrupt operation.

This chopper scheme is the standard for DC motors. The TMC236 adds torque control and protection and diagnostic features.

# 12. Documentation Revision

| Version | Author | Description |
|---|---|---|
| 2003-2007 | BD | Comments on the application of TMC236, TMC246, TMC239, TMC249 ICs, based on user requests and own developments |
| 2008-01-29 | BD | Added Zero Crossing information / avoiding different results for turn left and turn right |
| 2008-04-22 | BD | Added ChopSync |
| 2008-09-01 | BD | ™ for ChopSync / new release,<br>Added Documentation Revision, new style front page |
| 2009-12-07 | BD | Added microstep schematic values for 3.3V |
| 2016-11-21 | BD | Added SPI How-To |
| | | Info: Revision history will be kept for two years |