motioncontrol-community.org

Categories

- Posts (2)

Archive

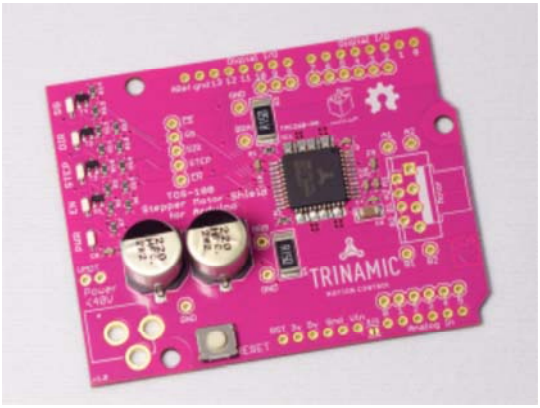
- July 2013

Meta

- Register
- Log in
- Read in RSS

About the TOS-100

The TOS-100 is an Arduino compatible Shield capable of driving one stepper motor up to to 1.7A that utilizes the Trinamic Motion Control TMC260 motor driver chip. For added compatibility with other arduino shields, the TOS-100 allows you to choose nearly any pin for any signal.



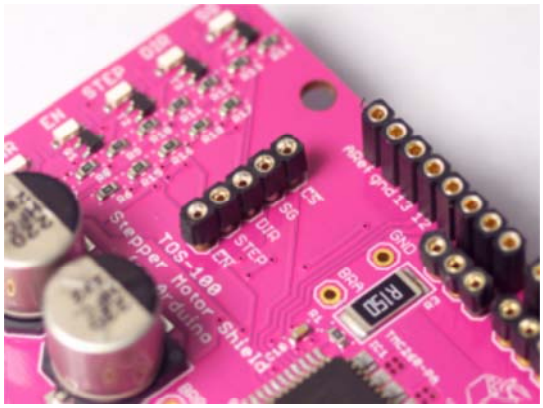
Assembling the TOS-100 Stepper Driver Shield

The TOS-100 shield comes assembled with all SMD parts. The through the hole parts represent various options to assemble the shield and therefore do

not come pre-soldered. To get the shield up and running you have to decide how you would like to assemble the shield and solder the appropriate parts:

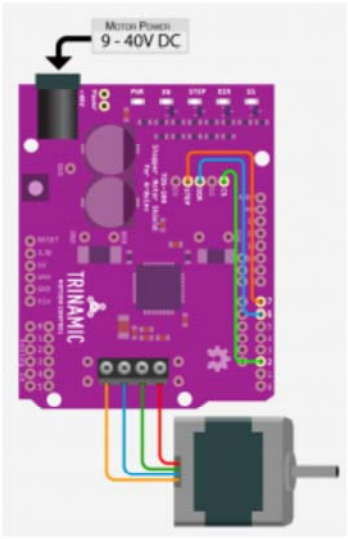
> The stacked headers for Arduino can be replaced by normal headers if you are sure that this is the topmost shield in your stack. In most cases you want to solder the normal stacked headers for flexibility.
>
> The pin connections between the TMC260 driver control pins can be either soldered with wires or with headers for flexibility.
>
> The DC connector can be used to provide external power for your motor (recommended – or use the Input power of the Arduino (not recommended, but the far simpler setup).
>
> The motor connector can either be soldered as screw terminals for a flexible connection or with Molex KK headers (e.g. Molex 22-05-7048) for easier motor connection. Or with anything else that you can fit into the 2,54mm or 3,0 mm connections. Be aware that the connectors or cable should withstand the maximum motor current (1.7A as absolute maximum).

Connecting the TMC260 pins with your Arduino



All control pins of the TMC260, which are not defined by the Arduino standard (e.g. the SPI connection) is not connected on the TOS-100 shield to give your a greater flexibility to suit the shield to your application. The configurable pins are located on the left side of the shield. To connect any pin of the TMC260 you can use the holes next to the Arduino headers. In the above picture the TMC260 pins are soldered with sockets and cables are used for an on-the-fly-wiring. The option to use headers may be the best way if you want to use the shield as an experimental shield in various configurations. If you want to use the shield permanently in one of your project it may be easiest to connect the pins with wires soldered in.
The TMC260 is configured via SPI. The MISO and MOSI pins for SPI are already connected. You need to connect the CS pin to a pin on the Arduino. You can choose any digital output pin you like. Check first for compatibility with other shields you want to use.

The example programs assume that you connect the CS pin to the Arduino digital output pin 2.
The easiest way to drive the stepper motor is to use the step and direction pins. In order to use those pins you need to connect those pins to Arduino digital output pins. The TMC260 stepper motor driver can be used solely over SPI (but this is not yet supported by the provided library).
The example programs assume that the step pin is connected to Arduino digital output 7 and the direction pin is connected to Arduino digital output pin 6.
If you want you can also connect the EN enable pin with an Arduino digital output pin. This is not strictly necessary. The default configuration ensures that the TMC260 stepper driver is permanently enabled and you can disable it in software.
The example programs expect the enable pin to be connected to digital output pin 8 on Arduino, but this connection is optional.



TMC 26x	Arduino Pin
CS	2
DIR	6
STEP	7
EN (optional)	8

(Illustration by bildr.org)

You can also the connect the Stall Guard output pin to an digital input pin on Arduino to react faster on motor stall situations. But you can also access the stall guard value via software. So this connection is optional, but provides a faster reaction on stall situations.

You may have noticed that there are not connections for the digital pins 0 and 1 of the Arduino. Those pins are used as RX/TX for the serial connection – most probably you will need that in your project. They were not made available in order to prevent errors.

Providing power to your TOS-100 Arduino Stepper Driver Shield.

The TOS-100 Arduino Stepper Driver Shield supports two power options:

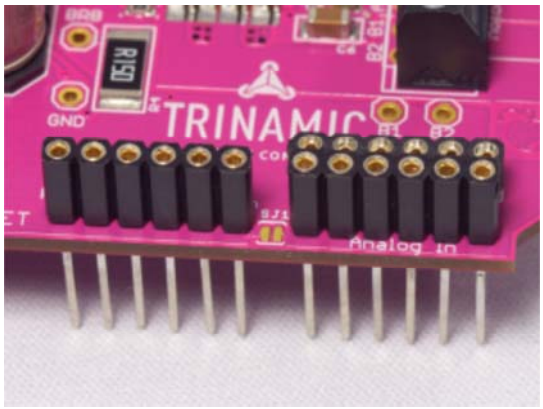
1. You can power it from an external source with 9 to 40V
2. You can use the Arduino input voltage which should be 9-12V

It is recommended to use the external DC connector to provide 12-24V externally (or more or less, depending on your motor specifications, the maxmum rating of the TMC262 is 40V).
As a rule of thumb for selecting the correct input voltage you can use this formula:

*Rcoil * Icoil << Motor Voltage < 25*Rcoil*Icoil*

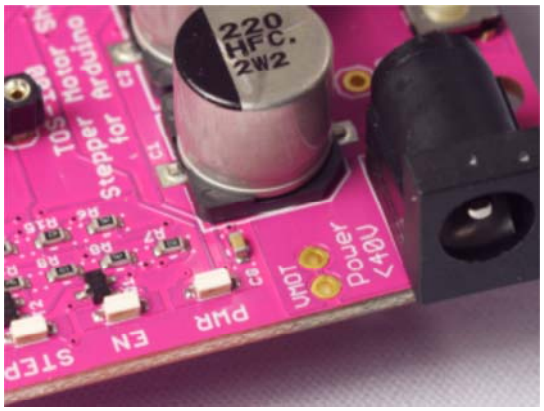
Rcoil and Icoil are the coil resistance and current as given in the datasheet of your motor. The motor voltage must be considerable higher than the the multiplication of these two values. For most Stepper motors something between 12V and 24V is a good value. The TMC262 ensure that your motor only sees the current you specify in software regardless of the voltage supplied.

If you want an extremely easy setup you can add a blob of solder to the solder jumper SJ1, located between the Arduino power and analogue input headers:



This connects the supply power of the TOS-100 to the input voltage of the Arduino. By that you have to power your Arduino externally from 9-12V. The TMC260 chip requires a minimum input voltage of at least 9V and the Arduino regulators have a maximum input voltage of 12V. You can see from those numbers and the recommendations above that this is normally not the optimum power condition for normal stepper motors. Additionally the linear regulator on the Arduino can get considerable hot. So this setup is not recommended, but can make your project significantly easier if 9-12V is enough for the motors you are using. You can solder in the DC barrel connector on the TOS-100 shield and use that to power your motor and your Arduino or use the DC Barrel connector on the Arduino to power your system. Be careful though: The PCB traces on the TOS-100 shield are quite robust to carry a good amount of current and considerable bigger than the VIN connection on the Arduino. So if your motor takes a good amount of power you are advised to use the DC barrel connector on the TOS-100 shield, for small stepper motors the VIN Pin of the Arduino may be sufficient.

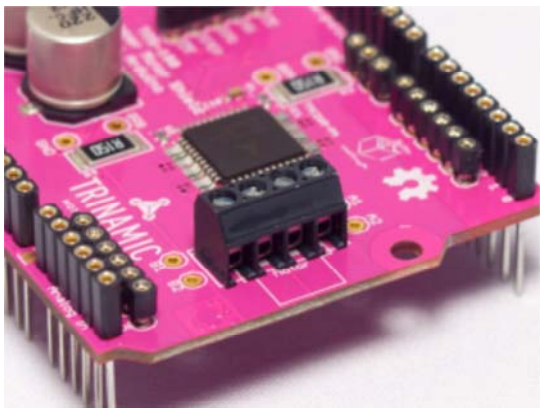
If you want to stack the TOS-100 shields you can provide power from one shield to another without adding a DC barrel connector to each shield. Next to the DC barrel connector there are two small holes for e.g. stackable headers to connect the motor voltage from one shield to the other



Connecting your motor to the TOS-100 Arduino Stepper Shield

The TOS-100 stepper shield provides two general options to connect your stepper motor:

- With provided screw terminals, 3mm pin spacing
- With Molex KK or any other header, 2,54mm (0,1“) pin spacing



For an experimental setup it is recommended to use the provided screw terminals. But you can use anything that fits into the holes as long as it withstands the current for your motor (better be on the safe side and use a connector rated for at least 2A)
You connect one phases of your motor to the ports A1 and A2 and the other phase to the ports B1 and B2.
If you are unsure which wire from your motor belongs to which phase there is an extremely easy way to test it:

If you can turn the motor by hand (i.e. it is not already built into a big machinery where you cannot reach the motor shaft) you should be able to turn your motor by hand. If you now connect two motor wires together two different things can happen:

> You can still turn the motor with the same ease as without connecting the wires. This is a sign that you are connecting two wires of different phase.
>
> When you turn the motor you can feel some dents. The motor provides a step like resistance against turning. This is a sign that you connected two wires of one phase -voila. The other two wires must belong to the other phase.

Alternatively you can of course measure the resistance between the wires. You should read two connections with a resistance pretty close to the specified coil resistance. All other connections should give you an unconnected reading (0L on most multimeters).
If you have found and connected the two motor phases of your stepper motor properly you are ready to go.

Do never ever disconnect the motor wires while the TOS-100 is powered. if you change anything in the motor wiring first disconnect the TOS-100 and Arduino power . Else you most probably damage the TOS-100, your Arduino or even your USB port in a worst case scenario.
If you are driving an unipolar motor in a bipolar way you must not connect the middle tap of the motor and ensure that this wire cannot touch anything (e.g. by insulating it properly). You can recognise the middle tap of your unipolar motor since it has half the resistance than the other two wires of the motor phase.

Installing the TOS-100 driver in Arduino

On Github you can download the TOS-100 Arduino Library. Don‘t be confused it is called TMC26X Arduino Library, since you can use it to drive any TMC260/TMC261/TMC260 chip. Unzip the library. You should get a folder called TMC26XStepper. copy that into the library directory in your Arduino folder. If you do not know how to locate it and how or why you should do it there is a good explanation on the Arduino web site.
After you have installed the library you need to restart the Arduino IDE so that it can find the new driver.

Testing your motor with the TOS-100 Stepper Driver Shield

The TMC26X driver comes with two example programs:

1. The ‚TMC26XExample‘ sketch is very simple example how you can use the TMC26X Library in your Arduino project.
2. The ‚TMC26XMotorTester‘ sketch is sophisticated test and analysis program for your stepper motor and probably the best way to begin.

Open the ‚TMC26XMotorTester' sketch by opening in the menu of Arduino ‚File ->Examples-> TMC26XStepper->TMC26XMotorTester'. If necessary adapt the pin numbers for the CS, DIR and STEP pins. If you have connected the EN pin as well adapt this in the example sketch to your connection.
After that you can upload the sketch to your Arduino.
The TMC260 stepper driver chip needs to be configured via SPI in order to drive a motor. This is done in the setup routine of the TMC26XMotorTest. So when you reset the Arduino you must have the TOS-100 Stepper Driver Shield powered. Or to put it the other way around – if you already have powered your Arduino and apply power to the TOS-100 Stepper Driver shield you must reset your Arduino. Power the TOS-100 Shield, the Arduino and connect it to an USB port on your computer.

You can download precompiled binaries of the client program:

Windows 32bit (64bit is unavaillable due to Processing limitations)

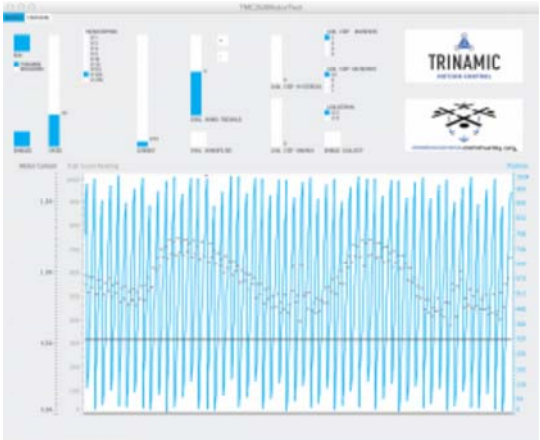
Mac OS X

Linux 32bit & 64bit

Download it, extract the zip archive and start the program. BTW: The program is written in processing and you can find the source code in the example folder.

The processing program should present you a list of available serial ports on your computer. Select the one your Arduino is connected to (if the Arduino is not yet connected, connect it and restart the client program).

If the motor test program has successfully connected to the TMC26XMotorTest on Arduino you will see the main screen for the motor configuration (the lower graph area will be empty – it is just updated if the motor moves):



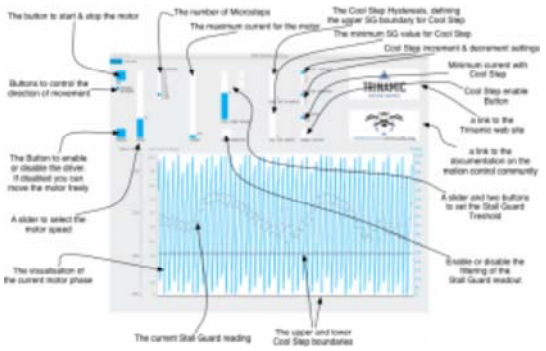
In the main screen you can adjust all basic parameters of the TMC260 driver chip. Let‘s go through them in order to start you motor:
In the middle you have the current slider. This can be adjusted to to provide the correct amount of current for your motor. For starters just ensure that the current is not higher than rated for you motor. If your motor is rated higher than the initial current selection you can increase it.
On the left side you see a slider for adjusting the speed the motor turns (in turns per minute) if it turns. You can leave it for now.
Next to the speed slider there is the selection of the microstepping. 16 is a good starting value for now – so you can leave that.
on the left side of the window you see the big ‚RUN‘ Button. This can be used to start the motor. It is a good idea to try to start the motor by pressing the button.
You motor should now turn slowly.
If not check:

- That the TOS-100 shield is powered with at least 9V
- That the motor is correctly connected to the TOS-100 (see above)
- That the TMC260 pins are connected to Arduino digital output pins and that those pins are specified in the TMC26xMotorTest sketch (and that the edited version is uploaded to Arduino). Especially check the CS and step pin.
- If you have connected the EN pin check that too.

If the motor is running according to the test program the Power LED should be permanently on, the Enabled LED should be on, the Step LED should be kind of dim (Since it is only showing the step impulses).
Now you can explore the several features of the TMC260 Stepper driver, play around with various speeds settings and so.
Keep in mind that for the time being the step impulse generation routine is not the best in the world – so don‘t care so much about the top rotational speed you can achieve. this will change soon. But your deductions that you can reach higher speed at lower microstepping (like 1/4th or 1/2 or even 1/1) is true.

Overview of the Motor Test UI

Here is a short overview of the various settings and readouts on the Motor Tester UI



The control elements of the TOS-100 Motor test client

The most important button is the top left one, to start or stop the motor. The button is blue if the motor is (supposed to be) running, light grey if not.
Below that you find two buttons to toggle the direction of the motor movement. Depending on your motor connection forward is one direction and backward the other.
And below the direction buttons is a big switch to enable or disable the motor driver completely. If the motor driver is disabled the motor can moved freely since the motor driver MOSFETs are disconnected.
The slider next to the motor run button can be used to control the speed of the motor. Depending on the microstepping value you may reach different top speeds, determined by the processing speed of the Arduino.
The radio buttons next to the speed slider select the current microstepping resolution of the motor. 1/256 gives the slowest speed and the smoothes motion. 1/1 is one full step and gives the highest speed.
Left of the current slider are the slider and buttons to set the stall guard threshold. It is a simple rule of thumb to configure it: If the SG readout is too high reduce it, if it is too low increase it. Fine tuning can be easier using the „+“/“-“ buttons than using the slider. With the Stall Guard Filter button you can enable the Stall Guard filter, so that the stall guard value becomes more steady – but contains less information (e.g. if the motor slips).
The cool Step minimum and hysteresis can be used to configure the cool step feature.
The cool step increment and decrement settings can be used to configure how fast the current is reduced and how fast it is increased if the load gets higher.
The minimum Cool Step current radio buttons can be used to select if the current is reduced to half the specified value or a fourth of it.
And last but not least there is a button to enable or disable the cool step feature.
On the right of the panel you will find the logos of Trinamic, linking to the Trinamic web site, e.g. to download the datasheets and the logo of the motion

control community, linking to the documentation (which you are reading right now).

The Readout Panel

The motor tester continually retrieves status data from the TMC260 chip regarding the current motor position, the current load read back by Stall Guard and the current going into the motor. This is displayed in the lower readout panel. The readout panel can be easily used to analyse the motor behaviour. The Stall Guard readout can be used to tune the Stall Guard settings to the current situation. In combination with the Cool Step threshold it is really easy to configure Cool Step and Stall Guard.

Play around with the values of the various configuration parameters until your happy. Later you can use them in your own Arduino sketch (e.g. based on the TMC26XExample sketch).

Further Reading

For further details you may check the API documentation of the TMC26x Arduino Library. To understand more about the inner working of the library and how to adopt it to your own project check the source code of the library on Github. If you are interested in the schematic and layout of the TOS-100 Shield (e.g. to use it for you personal projects) check the discussion of the TOS-100 Shield Layout & Schematic.

Reusing the Library, Schematic or PCB Layout for your Project

The TMC26X Arduino Library, the TOS-100 schematic and PCB Layout are issued under permissive open source licenses:
MIT License for the TMC26X Library
Creative Commons Attribution for the schematic and PCB layout
This allows you to use the source code, schematic and PCB layout as it is or in a modified form in your project or product without any restrictions. Your project or product does not need to be open source.