TMC2208 UART on BigTreeTech/BIQU SKR V1.1, V1.3, and V1.4 Controllers

by Vanessa_E

So.. You got yourself a BigTreeTech/BIQU SKR v1.1, v1.3, or v1.4 controller board, and you want to use TMC2208 driver modules on it, in UART mode, with Marlin? It's not difficult to set up.

Note: this Instructable is meant for the general-use LPC1768-based SKR boards -- it does not apply to the "Pro", "Mini", or any of the other models aimed at specific printer brands or use cases.

Prerequisites:

- You'll need to know how to do a little basic soldering.
- You'll want to have good lighting, and a decent magnifier lamp.
- You'll need some basic skill in using a text editor, and understand how to compile Marlin. Try their official guide. It's aimed at Re-ARM, but the process is the same for SKR boards.
- You'll need to make sure you have the PlatformIO TMCStepper library installed.
- Your PlatformIO framework-arduino-lpc176x platform module must be fairly recent as well.
- You'll need Marlin 2.0.x, preferably the bugfix-2.0.x branch, from their official repository.

You may not necessarily be able to just use any random download/snapshot of the Marlin source tree, so be ready to play with different commits. This will put you on that branch, even with 2.0.4.1-release:

```
git clone https://github.com/MarlinFirmware/Marlin.git
cd Marlin
git checkout bugfix-2.0.x
git reset --hard 484e1a62
```

When you're sure Marlin and PlatformIO are ready, go ahead and load your Marlin configs into your favorite editor or IDE (you'll do the necessary tweaking later in this Instructable).

If you're using the SKR v1.1 and have an LCD connected, please disconnect it prior to following this guide, just to avoid any possible interference. You can plug it back in later.

Supplies:

All users:

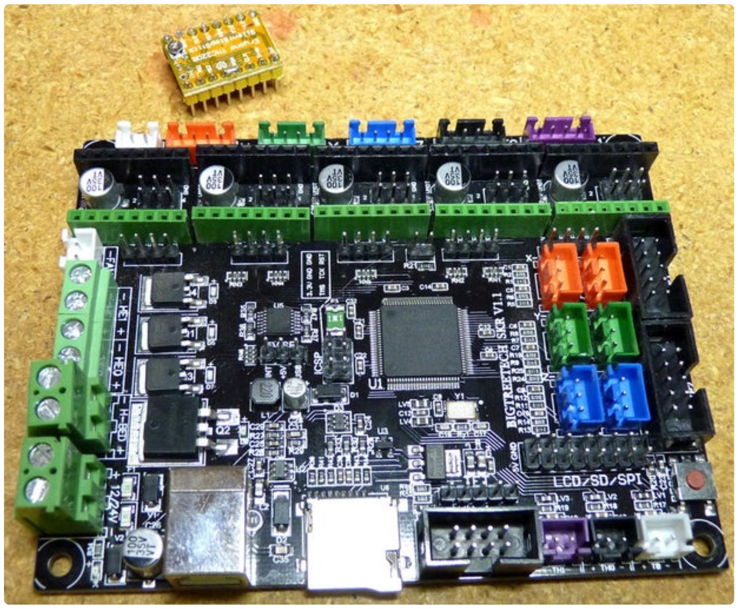
- Solder
- A fine soldering tip would be a very good idea also.

SKR v1.1 users only:

- Some female Dupont connectors (two per driver module). You can find these in a lot of older electronics -- I got mine from old PC cases. You probably have some in your junk box.... er, you DO have a junk box, right?
- If possible, some single-pin Dupont connector housings for the above (one per driver module)
- Some appropriate wire (anything will do as long as it's easy to work with)
- Heat shrink tubing (2 to 3 millimeter size, 2:1 shrink ratio or better, as long as it fits)

SKR v1.3 or v1.4 users only:

- Some shunt jumpers (one per driver module). I refer to the little caps that people routinely call "jumpers", used to temporarily, electrically bridge two pins together. To avoid confusing terms, I'll call these "shunts", as distinct from the pins they go on ("headers", but sometimes also called jumpers), or the pads on your driver modules that you'll be soldering on (yet again, also called jumpers)



Step 1: Modify Your Driver Modules...maybe

In most modern TMC2208 driver modules, there is an empty three-pad jumper present.

In most cases, that jumper is next to the middle two pins on the En/Step/Dir header. One such example is the white FYSETC module pictured above (main image).

In one variant made by Eryone (the amber board above, top-right), this three-pad jumper is not in the middle like usual, but instead is positioned way over near one edge where the potentiometer would normally be found, next to the three diagnostic holes, and aligned parallel to the edge of the driver module (and there's a two-pad jumper on the other end next to a resistor).

In most any kind of electronics, when you see a 3-pad jumper like this, usually you would short the middle pin to one end or the other, if anything at all. In driver modules like these, which pad should be shorted to to the middle technically depends on what SKR version you'll use the module on. But, the way driver modules are used allows us to make it simple: just solder-bridge all three pads together.

This connects the TMC2208 chip's PDN/UART signal to both the PDN pin on the module's header, and to the unused pin next to it. This configuration will work on pretty much all controller boards regardless of make or model, so long as the board still has a standard MS1-3 config jumper block between the driver module's headers, and can be configured or hacked to use UART mode. That includes the SKR v1.3 and v1.4 as well, though this is NOT needed on those.

Some modern modules have a three-pad jumper pre-configured for UART, along with three pins sticking up from the top.

BigTreeTech's v3.0 driver (the black PCB above, bottom-right) is an example of this type. The middle of those pins provides a convenient place to pick up the PDN/UART signal (the others are the NC and Clock pins).

On the bottom, there is the usual three-pad jumper (highlighted with a red ellipse), but one pair of pads is bridged with a 0-ohm link (which resembles a tiny surface-mount resistor, see the green box).

If you have these modules, you need not do any soldering, nor any other modifications. They're already set up and ready to use in UART mode on basically any brand and make of controller that can handle it (either by getting the PDN/UART signal from the aforementioned middle pin, or via already-present circuitry in the controller).

In some modern modules, the 3-pad jumper is pre-configured for UART, but there are no top-side pins.

Such drivers may either come with their PDN/UART jumper already solder-bridged for you, or there may be a thin trace connecting two pads, or they may have a 0-ohm link as with the above BTT modules.

If you'll be using them on an SKR v1.1, just solder-bridge the jumper's middle pad to the unconnected side. If the jumper has a 0-ohm link but it falls off while soldering, let your vacuum have it :-) and just solder-bridge all three pads together. As with the empty-three-pad-jumper kind mentioned above, you can do this soldering regardless of the SKR version you're using, if you want, but it's only needed on the v1.1.

Some older modules just have a two-pad middle jumper.

I have some Eryone driver of this type (the brighter-colored amber board hidden behind the "2 More Images" link above). Incidentally, they seem to be based on some old reference design, as they're basically identical to a number of other brands from the same period.

On these modules, there are at least two sets of jumper pads on the underside: one close the STEP pin, and one close to the middle two pins. There may also be a jumper on the right (assuming you're holding the module with the bottom facing you and the diagnostic holes to the right).

Solder-bridge the middle jumper's pads together.

Some driver modules have the chip and various other parts are on the top of the PCB.

On these modules, like the second black PCB above (hidden behind the "2 More Images" link), the parts aren't on the bottom side like usual. If you hold the module with the chip and such facing you, and the potentiometer and diagnostic holes to your left, there'll be one set of jumper pads on the right, and one set near the middle. There may also be a jumper on the left. There are at least two brands that put the chip and such on top, or used to in the past (Watterott and BigTreeTech prior to v3.0). In these cases, we're interested only in the jumper pads near the middle of the En/Step/Dir header.

As with the others, just use your soldering iron to bridge this middle pair of pads together. This will connect the driver chip's PDN/UART signal to the PDN pin on the driver module's header.

If your drivers have a three-pad jumper:

** If you're using <u>SKR v1.1</u>, jump ahead to "Patch Things Up".**

** If you're using <u>SKR v1.3 or v1.4</u>, jump ahead to "Close Some Circuits".**

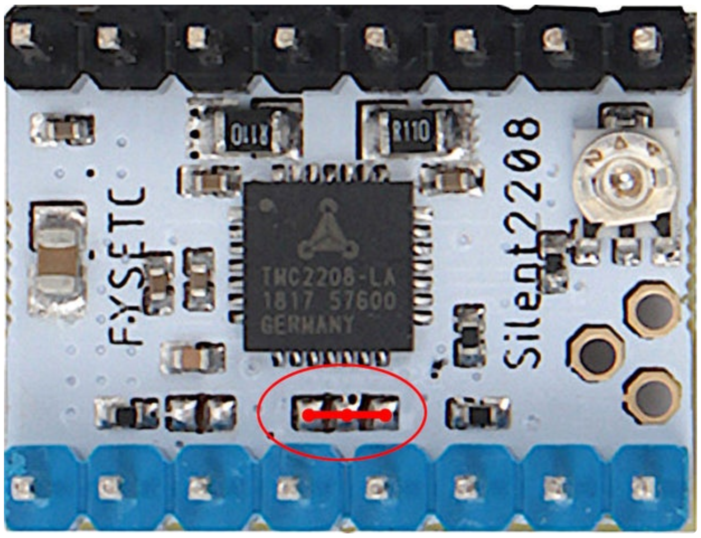
If your driver is one of the older ones with an empty two-pad jumper (parts on bottom or top):
** You have more soldering to do, so continue to the next step, "Modify Your Driver Modules (Part 2)".**

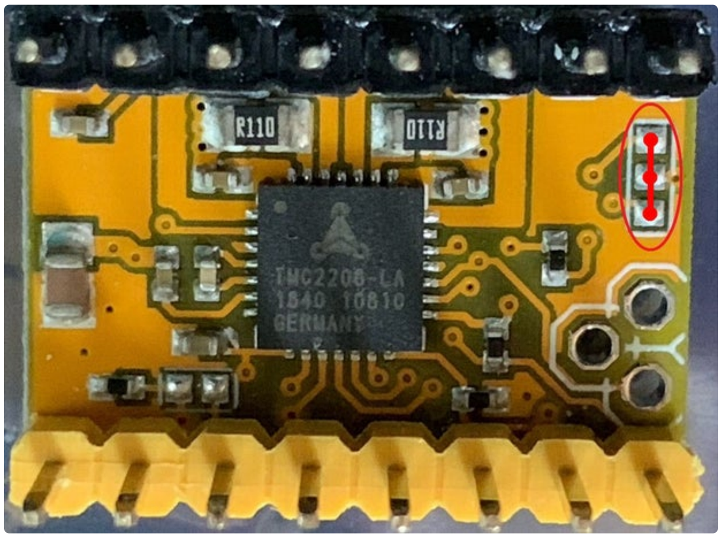
Everyone else:

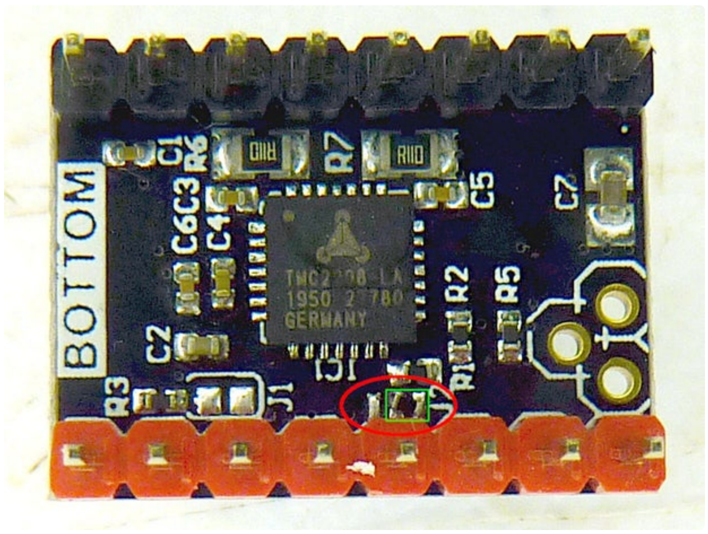
If your drivers have **no** PDN/UART jumpers, and have not been clearly and explicitly declared as being already wired for UART mode, then you must assume that they are the really old kind, and thus are NOT UART compatible.

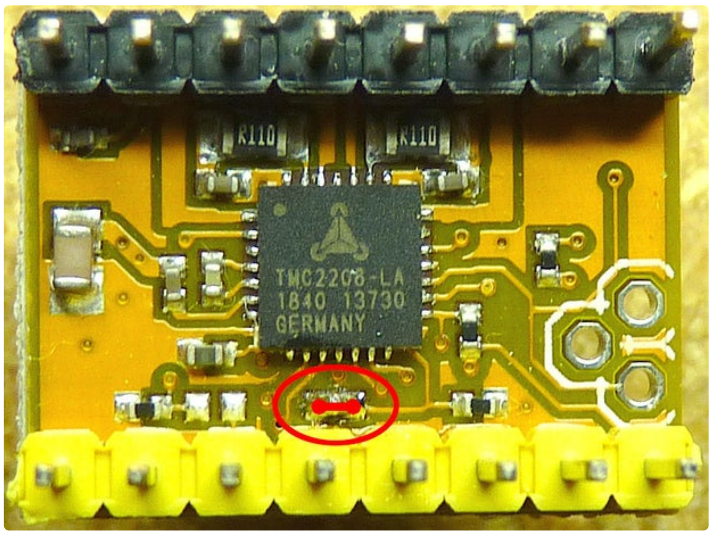
You will not be able to use this old type of driver for this Instructable, unless you have very good soldering skills and dexterity and can safely tap into the PDN/UART pin on the 2208 chip itself.

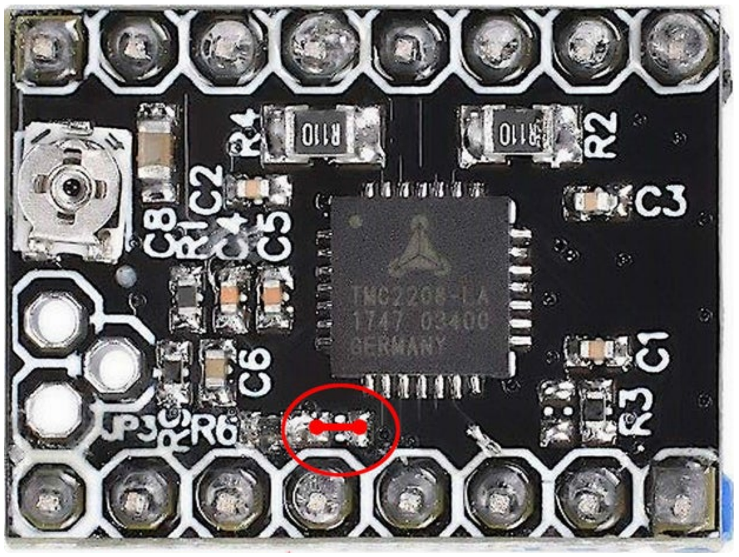
If that's the case, STOP HERE. You'll either have to hack your way into usability, or get new drivers that can be used with this Instructable.









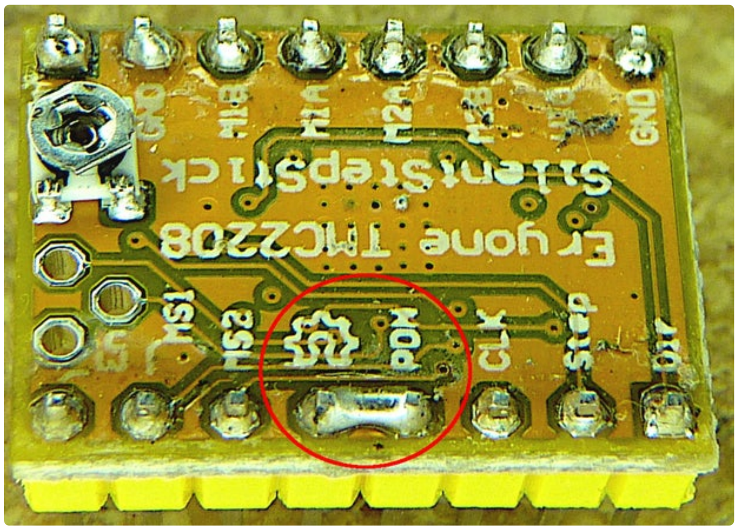


Step 2: Modify Your Driver Modules (part 2)

If your driver module is of the type with two pads on the middle jumper, or it has no jumper at all but comes from the factory claiming to be UART-ready, then you will most likely need to make a solder bridge on the top side of the driver module to connect the PDN pin to the unused pin next to it on the module's header. Some driver modules may not actually <u>need</u> this bridge, but if you've reached this step, just make the bridge anyway -- its presence won't hurt anything, and this eliminates a possible point of confusion.

This unused pin is most likely unmarked, or may be labeled "NC". As you can see on my Eryone modules, the manufacturer put the OpenHardware logo next to it.

We do this because on the controller board, that unused pin happens to be routed to the MS1-3 jumper header under the driver module, making for a convenient place to pick up the UART signal on controller boards that don't have UART support hard-wired in, without having to muck about with cutting traces or flipping pins to the top side of the driver modules.



▯ **If you're using SKR v1.1, proceed to the next step, "Patch Things Up".**

▯ **If you're using <u>SKR v1.3 or v1.4</u>, skip ahead to "Close Some Circuits".**

Step 3: Patch Things Up

For SKR v1.1 users only:

For whichever axis you're working on, remove all shunts from the board's MS1-3 config block.

You'll need to prepare some simple patch wires, with a female Dupont connector on each end.

This is single-wire UART mode, so the patch wires need no resistors or Y-splits (the heat shrink tubing in the middle of my patch wires is just a simple splice, since I re-used wires with Dupont connectors from other random, discarded electronics).

If your driver module is one of the common kind with a two-or three-pad jumper:

Only affix the the plastic connector housing to only one end of the patch wire -- leave the other connector bare.

Carefully bend that bare connector in half "backwards", to create a simple right-angle connection. By backwards I mean that once bent, you could put a really long pin through the connector part, without hitting the crimped-wire end. Add a drop of solder to the inside of the bend (the side that'll face down when plugged in) to reinforce it, and shrink a bit of heat-shrink tubing over it for insulation.

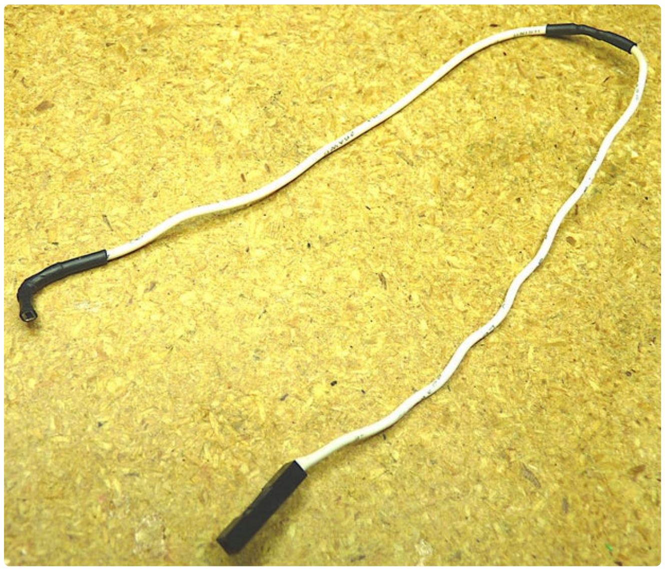
Connect the right-angle end to pin 5 of the config block on the SKR board for the axis you're dealing with (it should be clearly marked on the board next to the pin). This would normally be the MS3 signal line on drivers like the A4988 or DRV8825, but TMC2208 driver modules don't use this connection at all, so the soldering we did on the module routed the TMC2208 chip's PDN/UART signal such that we could pick it up from pin 5.

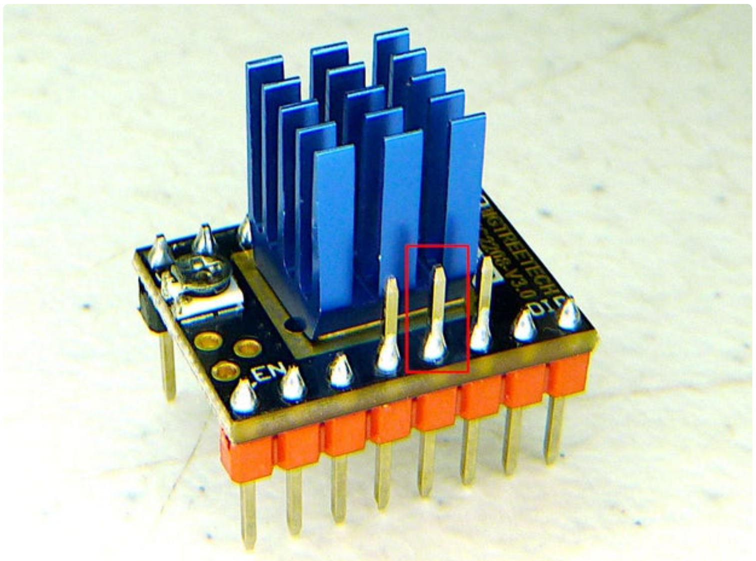
On the other hand, if your driver has the three pins sticking out of the top:

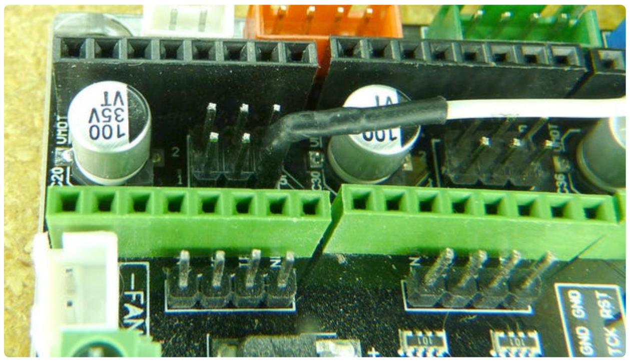
That is, if it's one of BigTreeTech's v3.0 modules with the UART jumper already bridged, or it's one that uses the same design ayway, connect one end of one your patch wire to the middle of the three pins (highlighted with a red box above).

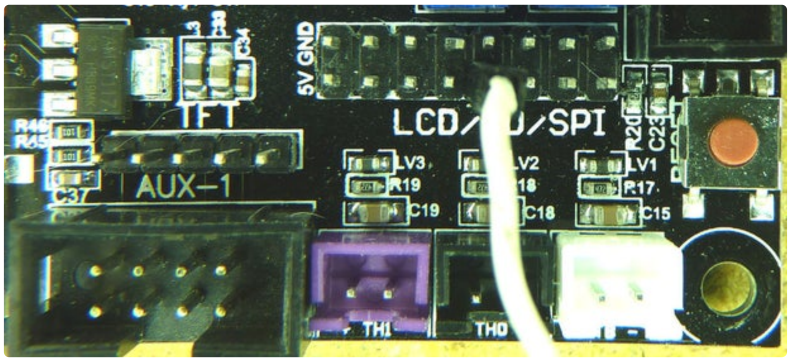
In either case:

Whatever you connected the one end to, connect the other end of the patch wire to an appropriate GPIO pin on the SKR's AUX1 or LCD/SD/SPI header. But just where are these signals? That's explained in the next step.









Step 4: Help Marlin Make the Connection

For SKR v1.1 users only:

In the above image, you'll see areas highlighted in yellow and orange, to point out specific connectors, with a few individual pins highlighted in red, green, blue, and gray. Those highlighted pins are the ones I had in mind for you to use, but there are a few more pins available on the board than you'll need for this project, so you have some leeway here.

Whatever pins you settle on, you will need to define them in Marlin to match where you intend to connect the patch wires on AUX1 and LCD/SD/SPI.

The aforementioned "leeway" can be confusing (it didn't do *me* any favors, to be sure), so to make it simple, I initially went with gloomyandy's pins definitions, reflected in the image above, and put the following into what is now src/pins/pins_BTT_SKR_V1.1.h, just before "LCD Controller" section (it can go anywhere):

```
#if HAS_DRIVER(TMC2208)
  #define X_SERIAL_TX_PIN    P2_06 // "2.6" on LCD/SD/SPI (aka AUX3)
  #define X_SERIAL_RX_PIN    P2_06
  #define Y_SERIAL_TX_PIN    P1_31 // "1.31" on LCD/SD/SPI
  #define Y_SERIAL_RX_PIN    P1_31
  #define Z_SERIAL_TX_PIN    P1_23 // "1.23" on LCD/SD/SPI
  #define Z_SERIAL_RX_PIN    P1_23
  #define E0_SERIAL_TX_PIN   P0_03 // "RX" on AUX1
  #define E0_SERIAL_RX_PIN   P0_03
#endif
```

If you intend to use an LCD such as RepRap Discount Full-Graphic 12864, the above pins will probably not work for you -- they didn't for me once I put a 12864 on the machine. But, since I only intend to use one extruder and I don't need to use "E1" to drive anything else (some people use it to drive a second Z motor), I was able to just use some of that driver slot's unused signals, via the little 4-pin male header next to it. Of course, one must leave the E1 driver slot empty (some people may habitually use it to hold a spare, unused driver module). Here's the configuration I settled on for that arrangement (note that this differs from the above image):

```
#if HAS_DRIVER(TMC2208)
  #define X_SERIAL_TX_PIN    P0_10 // E1 "EN"
  #define X_SERIAL_RX_PIN    P0_10
  #define Y_SERIAL_TX_PIN    P0_01 // E1 "STP"
  #define Y_SERIAL_RX_PIN    P0_01
  #define Z_SERIAL_TX_PIN    P0_00 // E1 "DIR"
  #define Z_SERIAL_RX_PIN    P0_00
  #define E0_SERIAL_TX_PIN   P2_06 // "2.6" on LCD/SD/SPI
  #define E0_SERIAL_RX_PIN   P2_06
#endif
```

If you use this latter configuration, you must also open src/pins/pins_BTT_SKR.h. You've got some tweaking to do there. Look for this section:

```
#ifndef E1_STEP_PIN
  #define E1_STEP_PIN      P0_01
#endif
#ifndef E1_DIR_PIN
  #define E1_DIR_PIN       P0_00
#endif
#ifndef E1_ENABLE_PIN
  #define E1_ENABLE_PIN    P0_10
#endif
```

For old versions of Marlin that are still new enough to support the SKR v1.1, look for these lines instead:

```
#define E1_STEP_PIN        P0_01
#define E1_DIR_PIN         P0_00
#define E1_ENABLE_PIN      P0_10
```

In both cases, simply comment-out the entire section by putting /* above the first line, and */ below the last line, or you can put // at the start of every line.

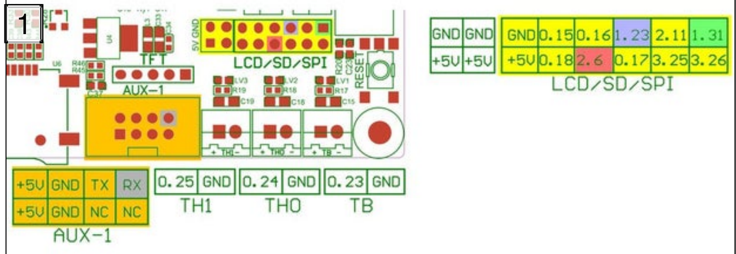
Finally, open Configuration_adv.h and change this setting:

```
#define SDCARD_CONNECTION LCD
```

You should be able to just find and uncomment that line (when commented-out, the default is to disable the LCD's SD card reader).

There may be other pins on the SKR v1.1 that can be used, but I don't know for sure what else you could try, so if the above options won't work, I'll have to leave it as an exercise to the reader to choose appropriate pins to keep the LCD, SD, and 2208's out of each other's way. Comments welcome!

⊠ Skip the next step and jump ahead to "Give Your Board Some Intelligence".



1. undefined

Step 5: Close Some Circuits

For SKR v1.3 or v1.4 users only:

These boards already have the internal circuitry needed to route the UART signals between the LPC1768 and the driver modules, and Marlin already has the proper pins definitions for them. So, the patch wires and pins.h changes mentioned in previous steps (that you hopefully skipped!) are not necessary to use TMC2208's with these boards.

On these boards, the MS1-3/SPI config block consists of four sets of three-pin jumper headers. On both boards, you must clear the config block of all shunts (many boards normally come with a bunch already plugged in), for each driver slot that'll hold a TMC2208 driver module.

Then...

On the v1.3:

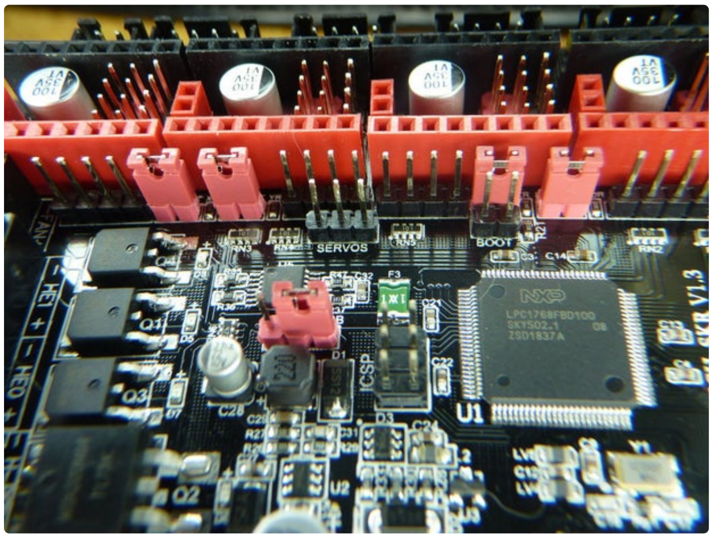
Simply plug shunts into the red UARTX, UARTY, etc. headers located next to (and outside of) the driver module headers on the SKR board. You can see them in the image above, left.

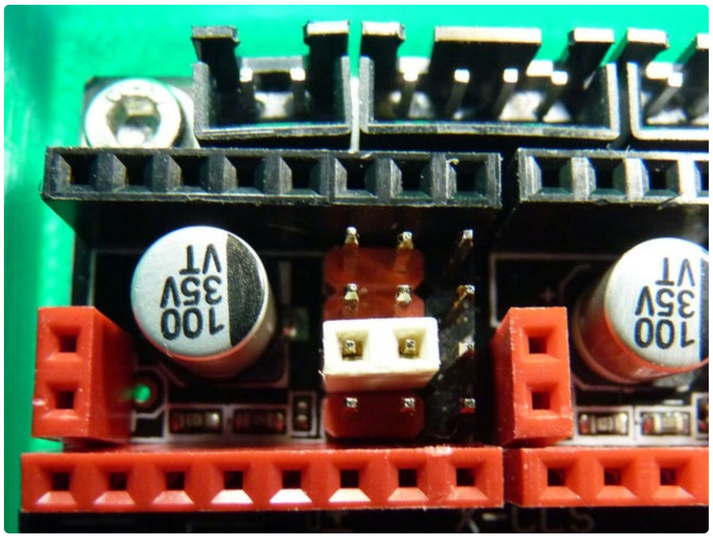
If you have any shunts plugged into the red diagnostic header on the right, next to the two X end stop connectors, remove those, too (this header is not shown in the above photo).

On the v1.4:

On these boards, you'll be using a couple of pins on the config block that you just cleared-out:

Put a single shunt back onto the left two pins of the second row from the red header. The image above, on the right, should make it clear, with the shunt shown in white for clarity.





Step 6: Give Your Board Some Intelligence

All users:

You will of course need to configure Marlin to be able to do its job, if you have not done so already. For brevity, I've omitted most settings that are commented-out or are either specific to my bot alone, or which don't apply to my bot at all. That is, I've tried to include only the stuff that everyone will need.

Start with these basic settings for Configuration.h to get it to at least talk to your PC properly:

```
#define SERIAL_PORT -1
//#define SERIAL_PORT_2 -1

#define BAUDRATE 250000
```

(note that the second serial port is commented-out)

You can use whatever baud rate you want that works with your PC/OS, from the list of suggestions just above that option in the config (not shown here). Faster is generally better, up to a point.

Among those Configuration.h settings, you must set your motherboard type as well. In all cases:

```
#define MOTHERBOARD   BOARD_BTT_SKR_V1_x
```

Where "x" is 1, 3, or 4, matching your board version.

After that, add whatever other configuration you need that's specific to your printer, i.e. thermistor type, temperature limits, motor steps/mm, speed and acceleration limits, build volume, end stops, and so on.

Step 7: Configure Marlin for Your Driver Modules

You'll need to set Marlin up for TMC2208 regular mode (not "STANDALONE") for whichever axes use them (I have all four; by default, Marlin expects A4988 driver modules), with appropriate settings for current, microstepping, stealthChop, spreadCycle, or Hybrid mode, and so on.

For example, in Configuration.h, I have these settings:

```
#define X_DRIVER_TYPE  TMC2208
#define Y_DRIVER_TYPE  TMC2208
#define Z_DRIVER_TYPE  TMC2208
#define E0_DRIVER_TYPE TMC2208
```

...and in Configuration_adv.h, I have these, in various places as appropriate for that config file:

```
#define MINIMUM_STEPPER_PULSE 1   // 0 is not recommended for TMC2208 anymore.
```

```
  #define HOLD_MULTIPLIER    0.5  // Scales down the holding current from run current
  #define INTERPOLATE       true  // Interpolate X/Y/Z_MICROSTEPS to 256

  #if AXIS_IS_TMC(X)
    #define X_CURRENT     600  // (mA) RMS current. Multiply by 1.414 for peak current.
    #define X_MICROSTEPS   16  // 0..256
    #define X_RSENSE     0.11
  #endif

  #if AXIS_IS_TMC(Y)
    #define Y_CURRENT     600
    #define Y_MICROSTEPS   16
    #define Y_RSENSE     0.11
  #endif

  #if AXIS_IS_TMC(Z)
    #define Z_CURRENT     800
    #define Z_MICROSTEPS   16
    #define Z_RSENSE     0.11
  #endif

  #if AXIS_IS_TMC(E0)
    #define E0_CURRENT    500
    #define E0_MICROSTEPS  16
    #define E0_RSENSE    0.11
  #endif
```

```
  #define STEALTHCHOP_XY
  #define STEALTHCHOP_Z
  //#define STEALTHCHOP_E
```

```
#define HYBRID_THRESHOLD
  #define X_HYBRID_THRESHOLD     150  // [mm/s]
  #define Y_HYBRID_THRESHOLD     150
  #define Z_HYBRID_THRESHOLD       3
  #define E0_HYBRID_THRESHOLD     10
```

```
#define TMC_DEBUG
```

Also, if you have SOFTWARE_DRIVER_ENABLE ... well... enabled, disable that option (comment-out the whole line, including the leading #define). It doesn't do what you think it does. :-)

Step 8: Finishing-up

Install the driver modules onto their headers.

If your SKR is already able to talk to the computer (for example, if it's brand new and you're just now opening the packaging), set SKR's 5v power shunt to "USB", hook up your USB cable, and wait for the SD card to show up on your computer, and mount/open it.

If it isn't talking to the computer yet, move its SD card to your computer's own card reader, and mount it.

Backup your board's existing firmware (just copy the SD card's contents to somewhere sane), then compile and upload the firmware to that SD card. When you see firmware.bin appear in the card's folder, unmount the SD card, move it back to the SKR if necessary, then reboot the SKR board, either by pressing its reset button, or by unplugging the USB cable for a few seconds.

Do a quick communications check when it comes back up: use a host program like Pronterface, Repetier-host, etc. to send a few commands like M115 or M119 or others that don't require hooking stuff up to the SKR board, just to make sure the firmware is working. If it all looks good, you should now be done with the basic configuration.

Now power-down the SKR, wire-up your printer's main 12/24v power supply, some motors, end stops, heaters, sensors, etc., move the SKR's 5v shunt to "INT" or "VDD" (the name varies depending on which board you're using), power back up, and check that nothing's gone wonky.

Make sure UART mode is active by sending M122 from your host program. You should see something similar to this at the end of the spew in its debug console:

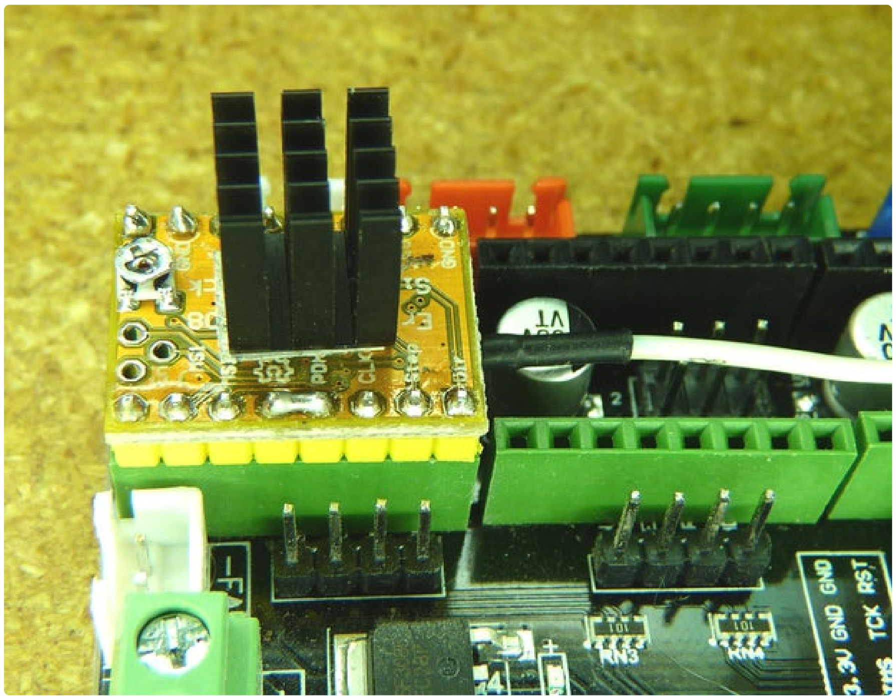
```
Driver registers:
  X 0xC0:09:00:00
  Y 0xC0:0A:00:00
  Z 0xC0:08:00:00
  E 0x80:09:00:00
Testing X connection... OK
Testing Y connection... OK
Testing Z connection... OK
Testing E connection... OK
```

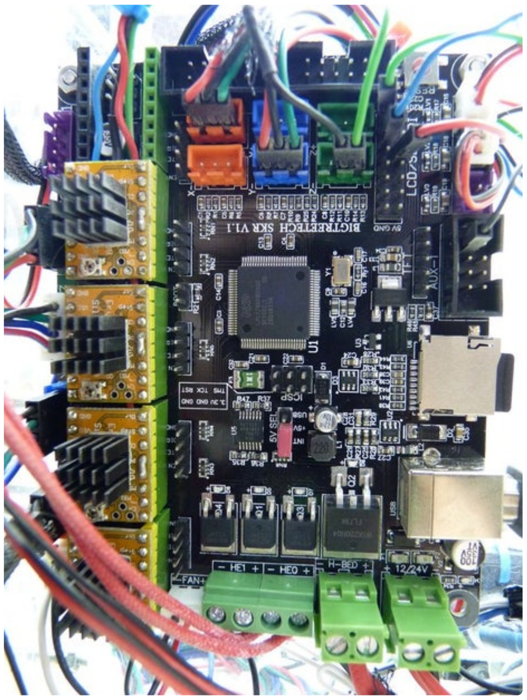
If that came out okay (i.e. nothing says ALL LOW or otherwise implies that there's some fault), check that your motors work, and run in the right directions.

If so, you may now begin routine calibrating and printing.

Enjoy your 32-bit goodness! :-)

As an aside: when you get a chance, you should also adjust the driver modules' potentiometers to set fallback currents, just in case the UART feature craps out some time (e.g. if a connector gets pulled out).





Step 9: Credits and Notes

- This Instructable was inspired by "UART This!" by seansadler.
- jammi clued me in to the UART-on-MS3 method via his RAMPS-oriented imgur guide, sure beats unsoldering and/or flipping pins around!
- The drawing of the pinouts of the SKR v1.1 came from BigTreeTech's public Github repository, modified by me.
- The photo of the TMC2208 module with the three-pad jumper that's next to the diagnostic holes was made by bookzurman, modified by me, and is used with permission.
- The photo of the TMC2208 module with three-pad PDN/UART jumper in the middle was "borrowed" from an Aliexpress listing, enhanced and modified by me.
- The photo of the TMC2208 module with everything on the top side of the PCB was "borrowed" from a Github thread, enhanced and modified by me.
- The clip of a driver module slot on the SKR v1.4 came from BigTreeTech's manual for that board.
- All other photos used in this Instructable are entirely my own work.

Fair Use rationale:

Try as I might, I could not find clean, freely-licensed images of two of the driver module variants (the "borrowed" ones above). Only non-free, low quality sources were to be found. Per the "four fair use factors" set forth in US copyright law:

1. Both images are used in a non-commercial, non-profit manner, purely for educational purposes (though I have no control over what Autodesk does with them outside of this Instructable).

2. The original images were published and deliberately made accessible to anyone on the Internet, without any kind

of account, password, or other obvious means of access control.

3. The images as obtained from the sources above were low-quality. They have been cropped to just the modules themselves, and then scaled, enhanced, and annotated by me. Due to the nature of the technology, using anything less than the full module images currently shown in this Instructable could lead to confusion, defeating the purpose of their use entirely.

4. These images have no obvious possibility to disrupt the market in which they were originally used, and no preference for one brand over any other(s) is intended.

About the connector and wire colors:

Shortly after I got my original SKR v1.1, I had to satisfy my OCD :-) and so I swapped the motor and endstop connectors around so that X is red, Y is green, and Z is blue, to match what you see in most 3d modeling programs, then color-coded my patch wires to match. However, some time after wiring it up, I killed it while probing around on the board (I shorted something out I guess), but since I still have that dead board in my junk box, I just went ahead and used it for most photos, just so that I didn't have to pull my good, working board out of the printer. The white patch wire seen in some photos is just an extra I had lying around.

The working board is shown in the final image since since I needed a "complete, installed" photo for this Instructable, so its wire and connector colors don't match the other photos.

How do you install the TMC2208 library in VSCODE?

So do I need the resistor in on the wire. I see your wire has a shrink tube in the middle of it. Is that the resistor

No, no resistors. This is single-wire UART mode. That heat shrink tube is just covering a plain old splice, since I reused the wires/Dupont connectors from other random bits of electronics. I'm frugal like that. :-)

Exactly what I needed. Thank you!

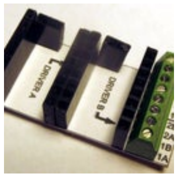
cheers :)

Just want to say thank you for a fantastic post. Helped me a lot!

Hi Vanessa, great instructional post! I wonder if you might be able to help me with an idea I have for dual drivers. I have two steppers on each xyz axis of my large Double H-bot. I would like to have two drivers for each axis (thinking TMC2209 V2.0). If I crated a shield that would take two 2 drivers - the second driver would probably need an extra wired UART pin. Would this be possible with either of the SKR boards or other 32bit board you know of? I look forward to hearing your thoughts.

I'm afraid I can't help you with that. That's quite outside of my experience.

Here is a mockup of my shield FYI



Thanks for the tips!
Any reason why you disabled Stealthchop on the extruder?

When in Hybrid mode, the extruder had a tendency to shift between stealthChop and spreadCycle modes in ways that made it noisy during retractions. Since stealthChop mode can lose steps on fast moves, I figured the safest solution was to just lock it into spreadCycle mode. Besides, it's still respectably quiet in that mode.

Good to know. Thanks for the quick reply!

Overall I think you did an outstanding job on this, I finally got my screen working due to the details you provided. The only problem I had was it would not compile when both serial ports are -1. Also, there is no need to do any soldering on the 2208's V3, just make sure to put the jumpers in for UART mode.

Actually you're supposed to comment-out "SERIAL_PORT_2" entirely. I'll update the Instructable regarding those v3 2208's. Thanks :-)

My error, I missed the comment-out on "SERIAL_PORT_2" entirely. I learned that if the V3 does not say UART on the box, then you must solder the two pads and that will make them UART ready, if it says UART, then nothing need be done to the stepper driver, just add the jumper next to the stepper driver plugs. Again thanks for a great work.

I should have said "in either case just add the jumper next"

For BTT TMC 2208, I have a resistor soldered between 2 of the 3 pads so I assume is in UART mode. is this assumption correct?

The driver is the same as described in the official manual https://github.com/bigtreetech/BIGTREETECH-TMC2208-V3.0/blob/master/TMC2208-V3.0%20manual..pdf . Here on page 6 they mention a pull resistor to be connected between TX and RX. Is this mandatory? I don't have an SKxy board but a trigorilla1.1 but I assume is almost the same.

Will this module work out-of-the box in UART, meaning just connect the TX pin to the board and load the customized firmware? Do I need to de-solder/remove the 3 bottom pin heads (red ones) and let the top 3 ones only?

My understanding is for the V3.0, if the box it came in does not say UART,then you need to solder the tabs in the middle next to the red supports for the pins, if it says UART you need do nothing. you do not need to de-solder/remove the 3 bottom pinheads (red ones) 3.0 is already configured with the exception to soldering the two pads together, a tough job as they are so small. Again if they are UART ready meaning that the box they came in says UART, then just configure the firmware, and there are jumpers that need to be added on the SKR V1.3 to connect the communications lines up, next to the steppers on the main board. If you are using a different board, then I do not know.

For BTT TMC 2208, I have a resistor soldered between 2 of the 3 pads so I assume is in UART mode. is this assumption correct?

For that specific kind of driver module and the SKR v1.3, it should just work as-is, yeah.

Here on page 6 they mention a pull resistor to be connected between TX and RX. Is this mandatory?

No. That's only for older setups. We don't use that on the SKR boards. The v1.1 uses single-wire UART mode, no resistors, while the v1.3 uses two-wire mode and already has the resistors onboard.

Will this module work out-of-the box in UART, meaning just connect the TX pin to the board and load the customized firmware? Do I need to de-solder/remove the 3 bottom pin heads (red ones) and let the top 3 ones only?

That one should work out of the box with the SKR v1.3 with *nothing* connected to the driver. You do not need to do any (de-)soldering. Just ignore the pins sticking up (incidentally, on the SKR v1.1, you could connect your UART patch wire to the middle pin).

Thanks for this info.

I will try to install them on Trigorilla board - is off-topic but is the only thread in which I found people able to give more tech details). Tthus I'm not 100% sure I can let the 2 red pins PDC and UART (the ones facing down) soldered.

That controller board does not have a standard MS1-3 jumper block, and I see no DIP switches or any other such things, so you may indeed have to cut or de-solder the PDN/UART pin from the bottom of the driver module.

Yeah, this is my feeling as well, better cut than cry :-) . Thanks, really helpful!

I see, if you leave port 2 enabled as a 1will that cause a problem. I have so much to learn on this new system. Again thanks

Hi
Thank you so much for sharing your knowledge.
I'm connecting the SKR 1.3 + TMC 2208 big tree v3.0 according to this guide and getting TMC CONNECTION ERROR

Did you remember to solder the jumpers on the drivers? Did you set the shunts on your SKR properly? If you're getting a connection error, you skipped a step.

Your guide doesn't work I soldered all three pads as you said and it broke my stepper driver. Also, I did everything you said to do and it still said TMC Connection Error every single time I followed you guide exactly and it still said error, I soldered all the pads and changed everthing in the software

There's no way that soldering those three pads together will break the driver, unless you got solder somewhere it should not go. Can you show me a close-up, well-focused photo of one of the non-functional drivers?

I bought them on amazon so I just went and bought some 2130 drivers that I know will work with my board.

It turns out I had #define SOFTWARE_DRIVER_ENABLE on in my firmware so none of my drivers are broken althoug my drivers work fine when I am using them in standby mode but whenever I try to use UART my LCD says TMC Connection Error. I am using the same board as you except I am using v2 TMC 2208, I even bought some TMC 2130 and they do the same thing fine in standby, but then the same message in SPI. I have the pins configured correctly on my board and changed my firmware, what is causing this?

If you had SOFTWARE_DRIVER_ENABLE turned on, then I guess you didn't follow the guide as exactly as you thought. ;-) It happens.

As for the connection error with the 2208's, my first guesses are that you connected the UART patch wires wrong (I did more than once), or you didn't put the pins definitions into pins_BIQU_SKR_V1.1.h (I botched this as well at first), or that your LPC1768 framework library is too old.

Are the UART patch wires the jumpers that you put on the board near the stepper drivers, I am running v1.3 so I thought I didn't need any wires running around the board, I thought all I needed was to change jumpers around which I have only on the red pins for UART like the manual says. What do I have to do for the pin definitions I am using the firmware that the manufacturer gave me so I thought that was already done for me. How do I upgrade my LP1768 library?

Thank you for sharing your first instructable! :)

You're quite welcome. I hope it helps a few people get up and running. :-)