The problem

Linear advance in Marlin firmware is somewhat incompatible with TMC2208 drivers in StealthChop mode (the motor will stop turning).
Unfortunately, Creality wires the TMC2208's in the version 4.2.2 board in "standalone" mode, which is 16 microsteps, StealthChop mode.

OPTIONS FOR TMC220X DEVICES, ONLY:

MS1/MS2: CONFIGURATION OF MICROSTEP RESOLUTION FOR STEP INPUT (TMC220X)		
MS2	**MS1**	**Microstep Setting**
GND	GND	8 microsteps
GND	VCC_IO	2 microsteps (half step)
VCC_IO	GND	4 microsteps (quarter step)
VCC_IO	VCC_IO	16 microsteps

The solution

Switch the TMC2208 on the Extruder axis to SpreadCycle mode.
Again, unfortunately, the only way to do this on the TMC2208 is by programming the internal registers through the UART. Creality does not do this…

Warning

Some extremely fine soldering will be needed, together with compiling the Marlin firmware from scratch.
I've used one of the more accessible pins on the STM32 processor so it isn't that bad, but it's still very easy to bridge the adjacent pins if you're not careful.
If you are not comfortable with these, either get a friend to help or look for a board which doesn't use the TMC2208. You're pretty much on your own here...

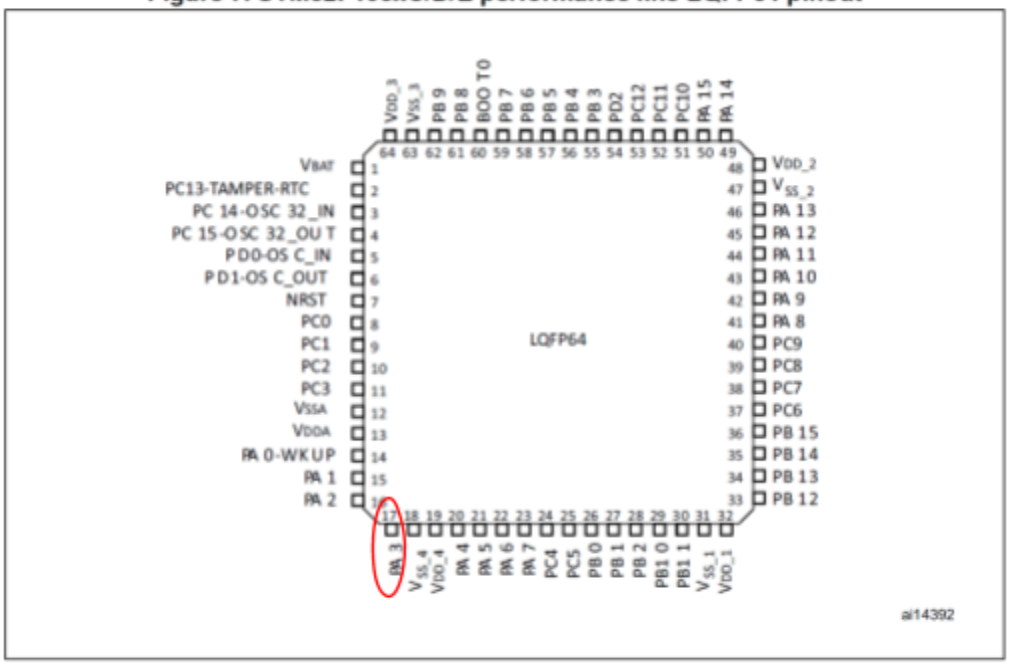
Hardware modifications (or, the easy part)

You will need:

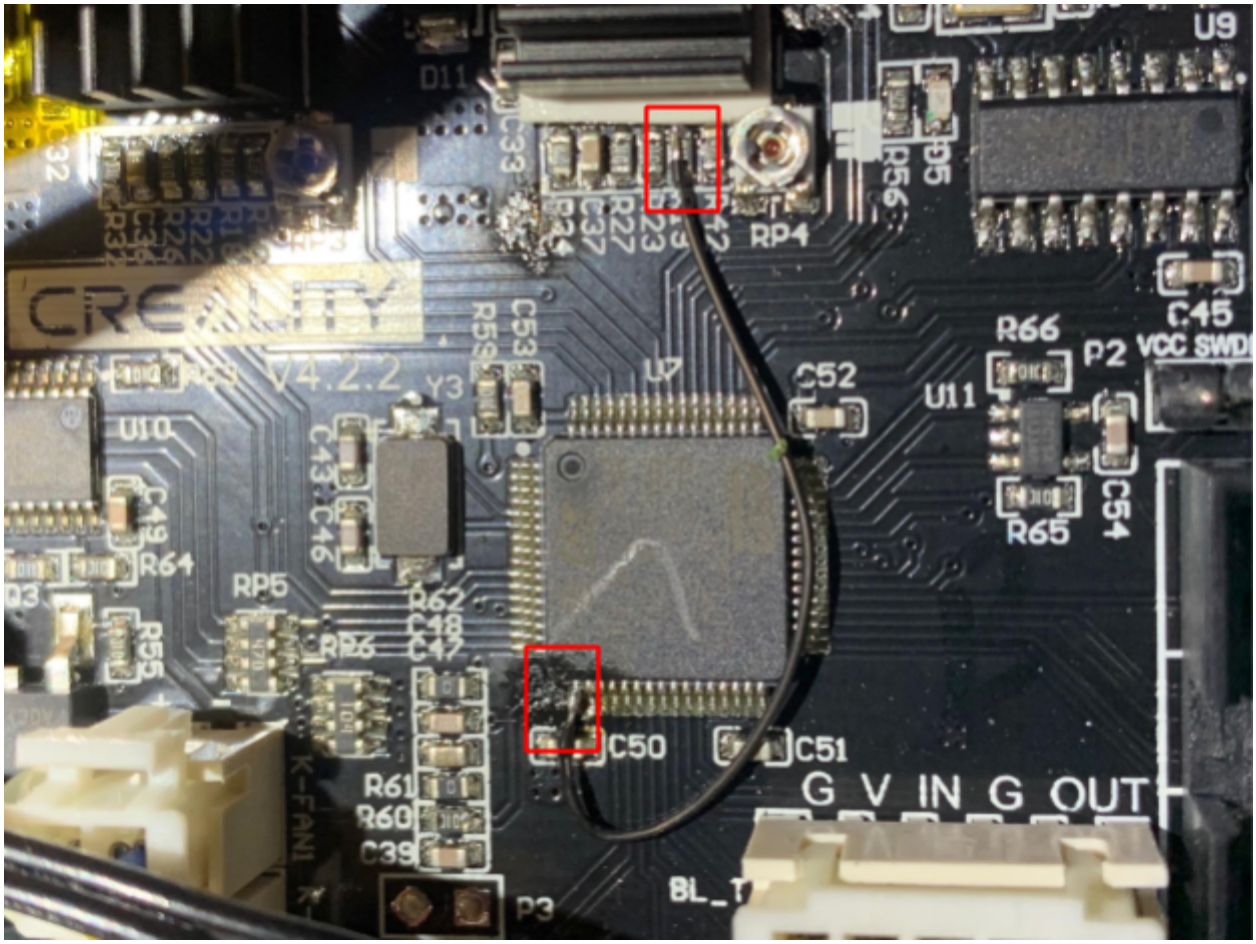
- A fine tipped soldering iron with steady hands
- Some very thin wire (30 gauge wire wrap wire works well)
- Fine solder so you can control the amount that goes down
- To be able to disassemble the motherboard from the printer
- Some thermal adhesive/tape if you remove the heatsink from the TMC

We need to solder a wire from pin PA3 of the processor to the UART pin on the TMC2208.
Luckily, this is on the edge of the package so it's easy to solder to.
Also fortunately for us, the UART pin on the TMC is pulled down via a resistor on the PCB. So instead of soldering directly to the TMC, we can solder on the resistor pin instead.

Figure 7. STM32F103xC/D/E performance line LQFP64 pinout



1. The above figure shows the package top view.



You'll need to solder a wire from the end of R19 nearer to the TMC2208, to pin 17 of the STM32 controller. If you have trouble accessing R19, you may have to remove the heatsink, but make sure you have some way of re-attaching it. Use a magnifying glass to check for shorts!
That's all for the hardware modifications, now to the fun part!

Firmware modifications

Again, you have to be comfortable with compiling Marlin from source. This is as good a time as any to learn, that way you can customize your machine *just the way you like it*.
I will not go into how to configure Marlin for your particular machine, you can use one of the configuration files included with the Marlin firmware as a start.

For the Creality machines, you need to change a line in "platformio.ini": Make sure "default_envs" is set to STM32F103RET6_creality, otherwise the code will not compile.

```
[platformio]
src_dir      = Marlin
boards_dir   = buildroot/share/PlatformIO/boards
default_envs = STM32F103RET6_creality
include_dir  = Marlin
```

Top hint: Do try building one "stock" firmware for your machine before you do any modifications!! That way you have a "known good" starting point to fall back on. If/when you start getting compiler errors after messing about with all the configuration files, you'll want a "clean" code that you *know* will work.

What we need to do:

- Set up a software serial port on the STM32
- Configure Marlin to communicate with the TMC2208
- Disable S curve interpolation (I found it made no difference whether it was enabled or not, but disable it for a start)
- Enable Linear Advance
- Disable StealthChop on the extruder axis

Step 1: Set up a software serial port on the STM32

Go to the "Libraries" tab in PlatformIO and search for SoftwareSerialM. Install it.

Import the Marlin directory into Visual Studio Code (if you haven't already) and open up "pins_CREALITY_v4.h" in /src/pins/stm32f1.

Add these lines to the bottom:

```
#if HAS_TMC_UART

 #define E0_SERIAL_TX_PIN                  PA3
 #define E0_SERIAL_RX_PIN                  PA3
 #define TMC_BAUD_RATE 19200

#endif
```

This will set up a serial port on pin PA3, with the baud rate at 19200 (for reliable communications).

Step 2: Configure Marlin to communicate with the TMC2208

Go to “configuration.h” and look for “#define E0_DRIVER_TYPE”. It should now read “TMC2208_STANDALONE”. Change it to “TMC2208” (without the quotes).

```
#define X_DRIVER_TYPE TMC2208_STANDALONE
#define Y_DRIVER_TYPE TMC2208_STANDALONE
#define Z_DRIVER_TYPE TMC2208_STANDALONE
//#define X2_DRIVER_TYPE A4988
//#define Y2_DRIVER_TYPE A4988
//#define Z2_DRIVER_TYPE A4988
//#define Z3_DRIVER_TYPE A4988
//#define Z4_DRIVER_TYPE A4988
#define E0_DRIVER_TYPE TMC2208
```

Step 3: Disable S curve interpolation

Comment out the line “#define S_CURVE_ACCELERATION” (place two slashes in front of it so that it reads //#define S_CURVE_ACCELERATION).

```
/**
 * S-Curve Acceleration
 *
 * This option eliminates vibration during printing by fitting a Bézier
 * curve to move acceleration, producing much smoother direction changes.
 *
 * See
https://github.com/synthetos/TinyG/wiki/Jerk-Controlled-Motion-Explained
 */
#define S_CURVE_ACCELERATION
```

Step 4. Enable Linear Advance

Go to configuration_adv.h and search for “lin_advance”.
Uncomment out the “//” in front of it.

```
#define LIN_ADVANCE
#if ENABLED(LIN_ADVANCE)
  //#define EXTRA_LIN_ADVANCE_K // Enable for second linear advance
constants
  #define LIN_ADVANCE_K 0.07    // Unit: mm compression per 1mm/s extruder
speed
  //#define LA_DEBUG            // If enabled, this will generate debug
information output over USB.
  #define EXPERIMENTAL_SCURVE // Enable this option to permit S-Curve
Acceleration
#endif
```

Step 5: Disable StealthChop on the extruder axis

Comment out #define STEALTHCHOP_E on the extruder axis.

```
  /**
   * TMC2130, TMC2160, TMC2208, TMC2209, TMC5130 and TMC5160 only
   * Use Trinamic's ultra quiet stepping mode.
   * When disabled, Marlin will use spreadCycle stepping mode.
   */
  #define STEALTHCHOP_XY
  #define STEALTHCHOP_Z
  //#define STEALTHCHOP_E
```

You’re done! Build the firmware and flash it into your board.

Testing (or how do we know your motherboard isn’t totally screwed)

To test whether the TMC2208 is working, issue a M122 command in your choice of terminal. I use Octoprint.

It should spit out some information about the TMC2208, like so:

```
Send: N75624 M122*32
Recv:       E
Recv: Enabled           true
Recv: Set current  800
Recv: RMS current  760
Recv: MAX current  1072
Recv: Run current  17/31
Recv: Hold current 8/31
Recv: CS actual    0/31
Recv: PWM scale
Recv: vsense       0=.325
Recv: stealthChop  false
Recv: msteps       16
Recv: tstep        11942
Recv: PWM thresh.
Recv: [mm/s]
Recv: OT prewarn   false
Recv: pwm scale sum      0
Recv: pwm scale auto     0
Recv: pwm offset auto    0
Recv: pwm grad auto      14
Recv: off time     0
Recv: blank time   16
Recv: hysteresis
Recv:  -end        -3
Recv:  -start           1
Recv: Stallguard thrs
Recv: uStep count  988
Recv: DRVSTATUS    E
Recv: sg_result
Recv: stst         *
Recv: olb
Recv: ola
Recv: s2gb
Recv: s2ga
Recv: otpw
Recv: ot
Recv: 157C
Recv: 150C
Recv: 143C


Recv: 120C
Recv: s2vsa
Recv: s2vsb
Recv: Driver registers:
```

```
Recv:        E      0x00:11:00:00
Recv:
Recv:
Recv: Testing E connection... OK
```

You want to make sure the line “stealthChop” reads FALSE!
Now, when you enable the motors you should hear a very faint hissing sound from the extruder motor. Don’t worry, this is normal!

If not, manually disable StealthChop by issuing a M569 S0 E and issue M122 again. Repeat until it shows “FALSE”.

I found that it sometimes still doesn’t disable Stealthchop even though it says FALSE… but sending the M569 S0 E command a few times (like 8-10 times) is enough to make sure it gets through. I put it in the “issue before print” script in Octopi, otherwise you can put it in the startup script in your slicer. Make doubly and triply sure it is disabled!

If “Testing E connection” shows “ALL LOW”, try a few more times. It doesn’t always work, for reasons I don’t want to spend time finding out.

The default current of 800mA should be enough, if it’s too high you run the risk of the TMC2208 overheating and skipping steps. SpreadCycle has more torque than StealthChop, so it doesn’t need as much current.

Calibrate the flow by following the instructions on the Marlin website. Hopefully we’re done!