
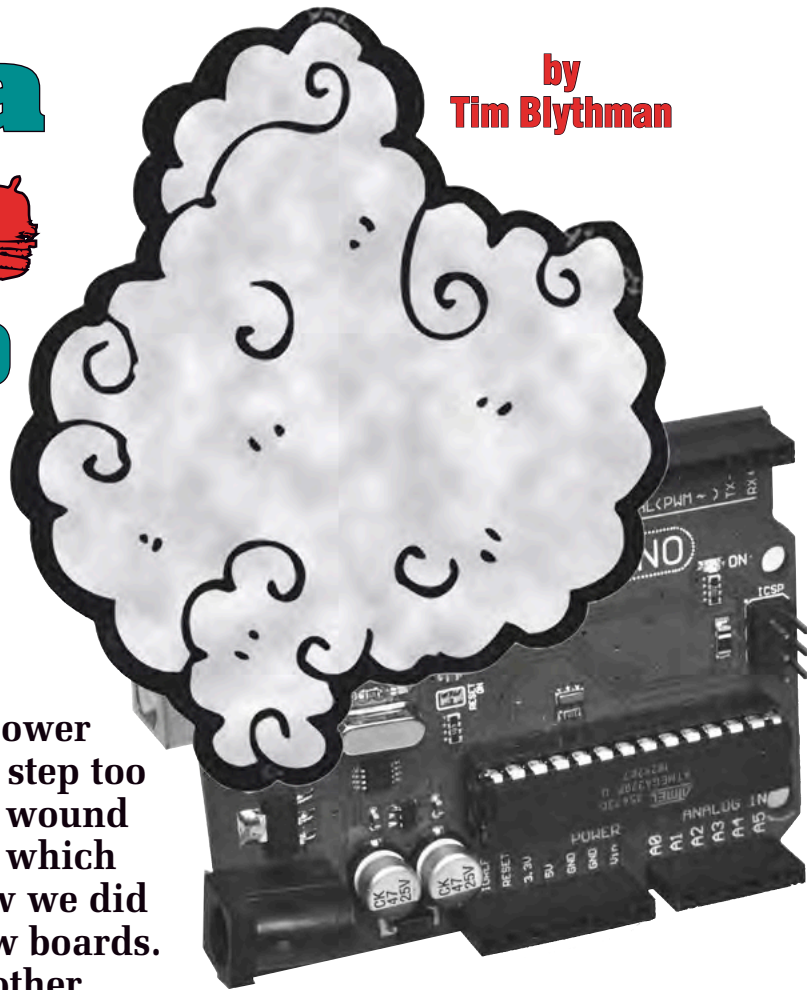


# Fixing a **RUSTED** Arduino Uno



by  
**Tim Blythman**



The Arduino Uno is a hardy beast, but occasionally we manage to let the magic smoke out. Perhaps our attempt to harness the power of lightning to run an Arduino was a step too far . . . who is to say? Regardless, we wound up with a few poor Arduino victims which needed to be resurrected. Here is how we did it, for less than the cost of buying new boards. These techniques should work with other Arduino boards, too.

**T**he Arduino Uno (and its various clones) has been designed to be resilient in the face of poor treatment by both beginners and experienced users. The ruggedness of the ATmega328 microcontroller is a major factor in this.

Despite this, we managed to break a few Unos. Most of these have been due to excessive voltages being applied to the DC jack or VIN input.

Let's look at the damage caused and how we can fix it. If you have an Arduino to fix, we're assuming that you have some experience with Arduino boards and the Arduino integrated development environment (IDE) software.

While there is no doubt that some Arduino-compatible boards are very cheap, almost to the point of being disposable, it can still be worthwhile to repair them. Below, we discuss three components that are likely to fail and how to replace them.

## Clones and DC regulators

Some Uno clones use a different 5V

regulator from the original, and these cannot withstand as high an input voltage. This stung us twice before we figured out what was going on.

Genuine Arduino Uno boards have an NCP1117 regulator, capable of handling up to 18V, while some clones use the AMS1117 instead, which is only good up to 15V. If (like us) you apply more than 15V to a clone, this voltage can find its way to places it shouldn't, like the USB port of a connected laptop. This can also burn out the regulator.

Replacing that regulator can not only fix the board, but you can replace it with a proper NCP1117 or equivalent, giving you the full 18V input range.

Note also that the original Uno, and most clones, have an ATmega16u2 microcontroller as their USB-serial converter IC. This chip can also be damaged as it is connected to the 'outside world'.

Some clones use a CH340 instead, and this could potentially also be damaged.

We haven't managed to blow up any ATmega328s (yet!), but we did have one that appeared to have a damaged ADC pin and as a result, was giving erroneous (and frustrating!) readings. If it does fail, this IC is easy to replace, as it is usually socketed.

One way to quickly check that the ATmega328 is functional is by pressing the reset button and watching the onboard LED. It flashes twice when the Arduino bootloader starts up. If you don't see this flash, either the micro is not getting power, it hasn't been programmed, or it is faulty.

Clones of the Arduino Mega and Leonardo often feature similar parts to those described above, so the following advice is pertinent to these boards, if not relevant to all components.

## Things that go pop

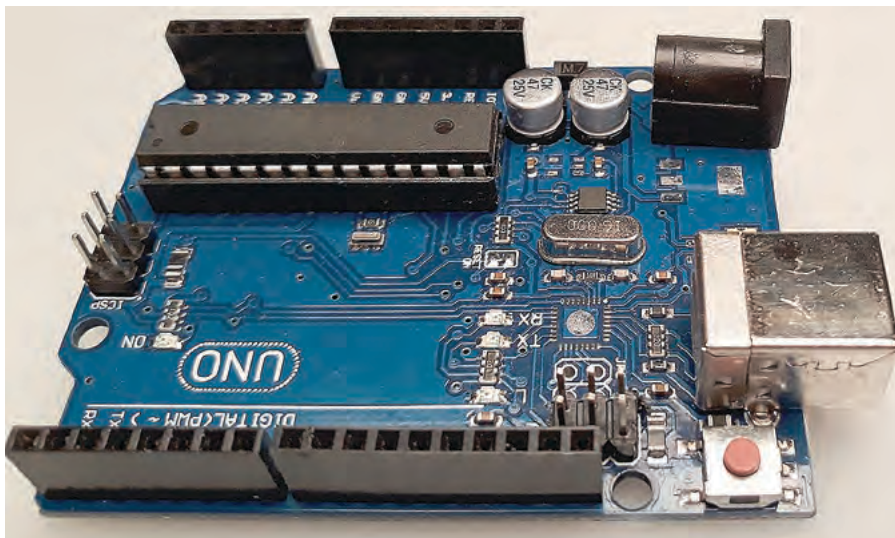
The most likely component to require replacement on a dead Uno board is the main voltage regulator.

Referring to the official schematic for the most common "R3" variant shown in Fig.1, this part is labelled









**Fig.2: one of our boards after removing the defective parts and cleaning the pads. We've also removed the residual flux; the result is almost like a brand-new board.**

If you have a problem with this IC, this part is available over the counter (Jaycar Cat ZZ8727, Altronics Cat Z5126 or Z5125 without the bootloader).

For the other parts, you will probably have to order from a larger supplier like Digi-Key or Mouser.

For U1, we ordered an NC-P1117LPST50 regulator. The part we ordered also had a T3G suffix, but this only refers to how the part is supplied (tape and reel in this case).

For U2, we ordered the LP2985-33DBV. The part we used also had an "R" at the end, again indicating that it is supplied on tape and reel.

U3 is an ATmega16u2 in a 32 pin VQFN package, with a part code of ATmega16u2-MU. Again, this had an "R" suffix to indicate tape and reel.

As mentioned earlier, depending on how you plan to use your Arduino, you could just remove a damaged 3.3V regulator and not replace it if you don't

need the 3.3V rail.

## Equipment needed

U3 comes in a QFN package, which is short for Quad Flat No-leads. It is very hard to solder or desolder without SMD-specific gear. We used a hot air rework station (available quite cheaply online) and solder paste, as well as the tools noted below.

Removing U1 and U2 is difficult without a hot air station, but possible. Replacements can be fitted with a temperature-adjustable soldering iron, although you may need a fine tip. Tweezers, flux paste and solder braid (solder wick) are also very helpful.

A magnifying glass will make working with these small parts easier. Even a mobile phone camera with digital zoom can let you get in close enough to inspect your work.

Note that flux generates a bit of smoke when heat is applied. Use a fume extraction hood or work in a location with excellent ventilation. We set up a small 12V computer fan to suck the fumes away. It probably isn't good for the fan in the long run, but it is better for our lungs.

Flux removal solution is useful for cleaning up afterwards, as the generous use of flux makes the process much easier. Isopropyl alcohol or acetone can be used if you don't have a dedicated flux removal solution. Take care, as many of these compounds are quite flammable.

## Remove the old chips

Naturally, the first step in replacing

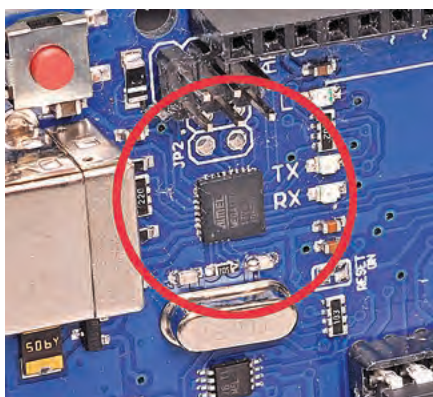
the defective ICs is to remove the old ones. If you have access to a hot air station, then it will be easy.

Grasp the defective part using tweezers with one hand and lift the board by a few millimetres, holding onto the part to be removed only. If you lift it too high, solder is likely to splash around. Aim the hot air at the part, and after around 20 seconds, the solder will melt and the weight of the board will pull the two apart. If you smell burning or see charring, the air is too hot, and the board may be damaged.

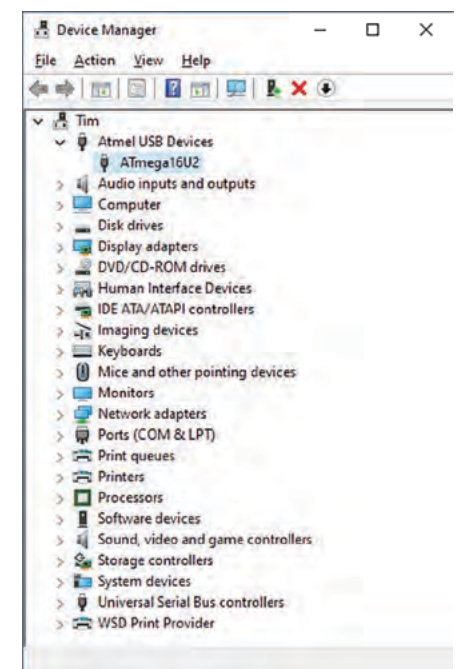
If you don't have a hot air station, you'll need to melt the solder on all the pins together, so they all come away at the same time. One way to do this is to build up a large blob of solder around the part, covering all the pins on both sides. Or if you're fast, you can alternately heat the two sides of the chip and rely on residual heat to keep one side molten while you lift the part off.

Alternatively, you can cut the pins off while the component is still soldered to the board; then desolder the pins individually. But it's easy to damage the PCB tracks when cutting the pins on such small parts, and this is not possible for U3 as it has no pins.

Once the defective components are gone, clean the pads using the flux paste and solder wick. Apply flux to



**Fig.3: if you apply just the right amount of solder to the QFN pins, with plenty of flux, you should get nice clean joints like these.**



**Fig.4: If the ATmega16u2 chip is soldered correctly, Windows Device Manager should show it as a connected device when the board is plugged in.**

the pads and rest the end of the braid on the pad. Press down on the braid with the iron and gently slide it to the side. The less residual solder left behind, the better the final result will be. We were able to get the pads nearly looking like they had never been soldered (see Fig.2).

## Fitting the replacements

For U1 and U2, apply flux to the pads and rest the parts on the pads. These parts have a different number of pins on each side, so the correct orientation should be obvious. The flux may help to keep them in place, but it's best to also hold them with tweezers.

Apply some more flux to the top of the pins. Clean the tip of your iron, add some solder and apply the tip to one of the pins. For U1, try one of the small pins, as this will be less affected by the large copper track below. The flux will draw solder from the tip and onto the pin.

If necessary, use the tweezers to adjust the position of the part, ensuring it is lined up with the pads and flat against the PCB. Once this is done, solder the remaining pins, turning up the heat for the large tab on U1.

If you get a solder bridge, ensure all the pins are soldered down before attempting to correct it. This will prevent the part from moving. Apply flux, then the braid followed by the iron and gently pull away.

## Fitting U3

The QFN part, U3, is a bit trickier to replace; but without much prior experience with QFN, we ached it two times in a row. The pads are so far recessed that it is really difficult getting solder onto them. We tried loading up our iron with solder to get close to the pins, but it didn't work. You may have better luck trying this technique with a very fine-tipped iron.

So we had to use solder paste and hot air. If you have access to a solder stencil to suit a QFN32 part, use it, but this isn't a requirement.

Start by applying a generous amount of flux paste to all the pads, including the large central tab. Squeeze out a small amount of solder paste and mix it into the flux paste along each side of the IC. It should go right into the corners. The amount of paste needed is minimal, perhaps what you could pick up on the tip (not the head!) of a pin for each of the four sides.

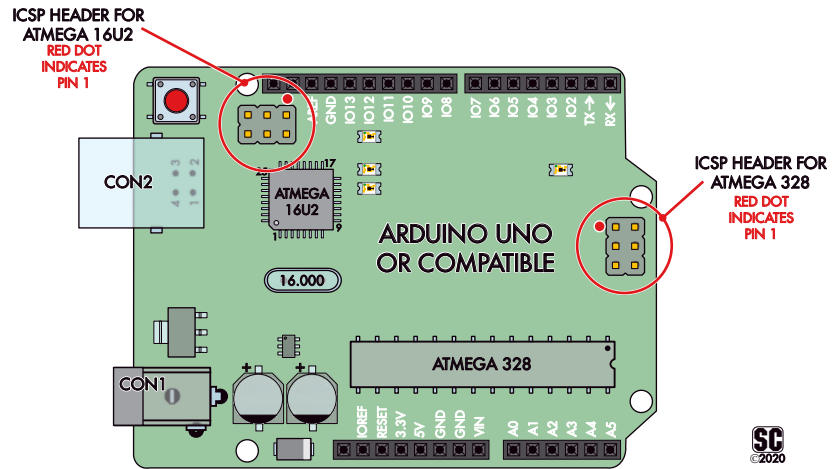


Fig.5: all Arduino Uno boards should have two six-pin in-circuit serial (ISP) programming headers, as shown here; one for each onboard micro.

Sit the part on top, ensuring that the pin 1 marking lines up with that on the PCB. If you have trouble seeing it, position the 'Atmel' text on top of the chip to be closest to the USB socket. Ensure that the IC is located centrally on the footprint and hold it there with tweezers.

Apply heat with the hot air gun directly to the top of the chip; you don't want the air to move the flux or solder paste too much. The flux should soften and flow, and eventually, the solder paste will coalesce towards the pins. You need to ensure there are no grey smears of solder paste left, although there may be silvery balls floating around. This is fine, as they can be picked off later to avoid short circuits.

Once you are sure that U3 has been soldered in place, clean it up by loading the tip of a fine-tipped soldering iron with a small ball of solder. Apply fresh flux paste to the pins and gently drag the tip along one edge at a time. If you have the right amount of solder, a nice-looking fillet should be left behind.

If you get bridges between pins, try again with less solder on the tip to help

remove the excess. The combination of surface tension and flux should leave a clearly visible fillet of solder to each pad (see Fig.3 – close-up of QFN pins).

## Testing

Before cleaning up the board, you can test that U3 is soldered correctly by trying to connect the Uno to a computer. While the ATmega16u2 does not have any firmware loaded initially, these chips come loaded with a "DFU" (device firmware upgrade) bootloader which means that a Windows computer will recognise that a device is connected (see Fig.4).

If you see a similar device appear, then the ATmega16u2 is communicating correctly, and you can clean any excess flux off the PCB. A fine brush (like an old toothbrush) is handy for cleaning among the pins. Note: do not use a toothbrush for brushing teeth after this!

If it doesn't appear in Device Manager, you need to resolder the chip and try again.

## Loading the firmware

As we mentioned a little earlier, the

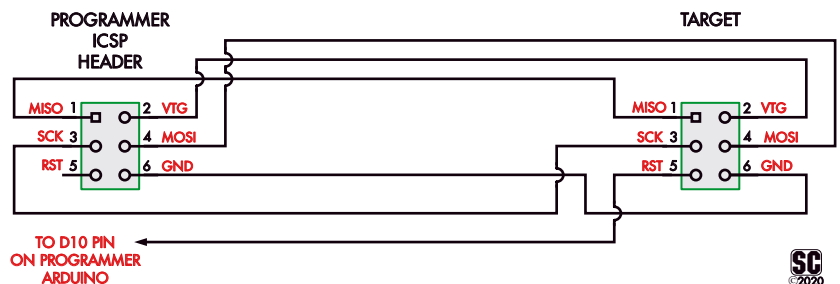


Fig.6: This view of our ISP jumper wire is shown from above (as it would look plugged into the top of the board). The stray male jumper goes to a dedicated pin on the programmer board (pin D10 by default) while the other five pins simply go to the corresponding pin on the programmer ISP headers.



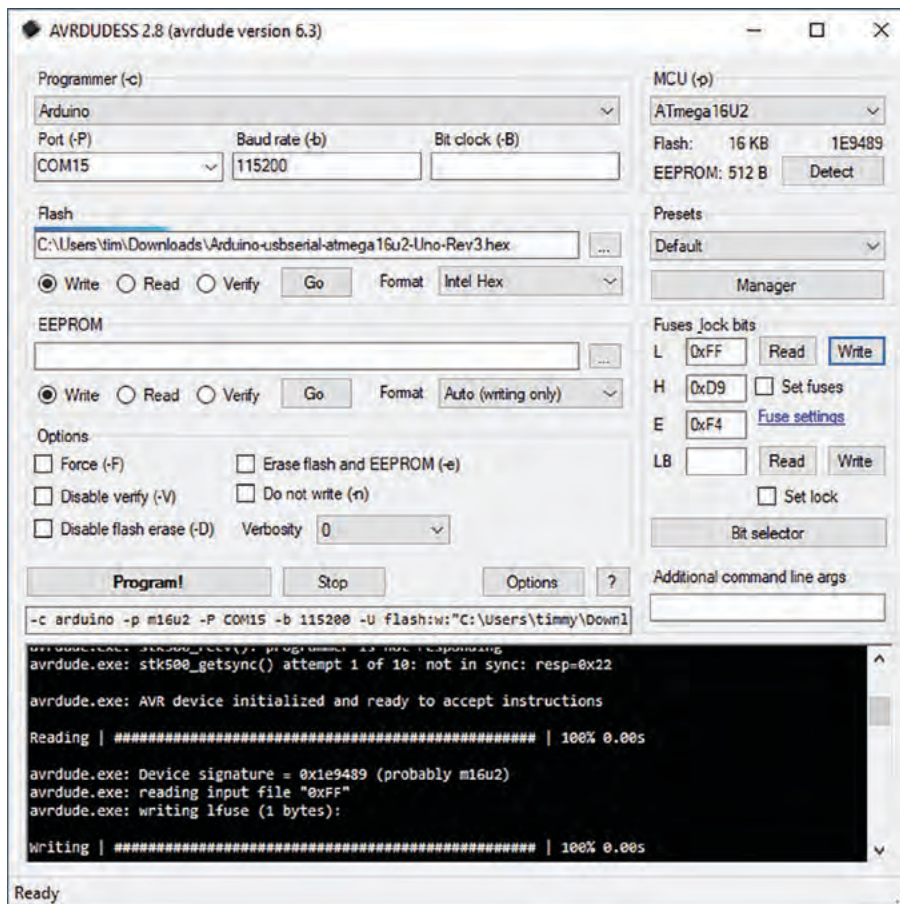


Fig.7: here are the required AVRDUDESS programming settings for the ATmega16u2. The port at top left should be the serial port of the programmer Arduino.

ATmega16u2 needs firmware to be loaded to operate as USB-serial converter. While the DFU bootloader can be used to upload firmware (using the Atmel Flip software), we found that it did not properly set the configuration fuses, meaning that it did not operate at the correct baud rate.

So we'll describe a more general method. This doesn't use the DFU bootloader, but does require a small amount of extra hardware. This method can also be used to load the Arduino bootloader onto a blank ATmega328 chip.

To do this, we use an ISP programmer, which plugs into the 3x2 pin ISP header. The Uno board has two ISP headers, one for the ATmega328 and one for the ATmega16u2 (see Fig.5). The process to program both is practically the same, but the firmware image is different.

These chips can be programmed by using another Arduino board. Any 5V Arduino board with an ISP header should be usable, such as the Uno, Mega and Leonardo (and their clones). A sketch to do this is included with

the Arduino IDE software download. The only extra hardware needed is a simple jumper cable to connect the programmer to the target board (see Fig.6).

Make up the cable as shown. You can use a set of individual jumper leads with DuPont headers on each end (packs of these are available from Jaycar & Altronics). Alternatively, do what we did and solder a length of ribbon cable to a pair of 2x3 female headers, with heatsink tubing used to protect the solder joints.

From the Arduino IDE, open the ArduinoISP sketch from the following menu item: File -> Examples -> 11.ArduinoISP -> ArduinoISP. If you can't find it, try upgrading to the latest version of the IDE. Select the correct board (for use as the programmer) and serial port and upload the sketch.

## Programmer software

You also need to load appropriate software onto your PC, to upload the firmware image and fuse settings to the Arduino programmer. Luckily, such a program is also included with the Ar-

duino IDE, and it is called AVRDUDE, the utility that performs the uploading of sketches to the boards. By the way, AVRDUDE is short for "AVR Downloader/Uploader".

To make things easier, we will use AVRDUDESS, a graphical interface for AVRDUDE. You have to download this separately, from: [siliconchip.com.au/link/aaxh](http://siliconchip.com.au/link/aaxh)

As AVRDUDE will have been installed along with the Arduino IDE, once installed, AVRDUDESS should automatically detect its presence. With AVRDUDESS running, you need to adjust its settings to be like those shown in Fig.7. Be careful here since selecting the wrong Fuse byte values (L/H/E at right) can 'brick' the chip!

From the top, set the Programmer to "Arduino" and ensure the port and baud rate match the Arduino you are using as a programmer. The baud rate should be 19,200 as this is the default for the ArduinoISP sketch (the code snippet shown in Fig.8 is where to change this if you need to).

Connect the target end of the programmer to the target board at the ATmega16u2 ISP header, ensuring that the pin 1 designations line up, as shown in Fig.6.

The power LED on the target board should light up as the programming cable provides power. If it does not, check the wiring.

We occasionally found that connecting the target board caused the USB connection to the programmer to drop out. Try unplugging and replugging the USB cable in this case.

To do a quick connectivity check, press the "Detect" button at the top right of the AVRDUDESS window. After a short delay, you should see the message in the lower window:

**Detected 1e9489 = ATmega16U2**

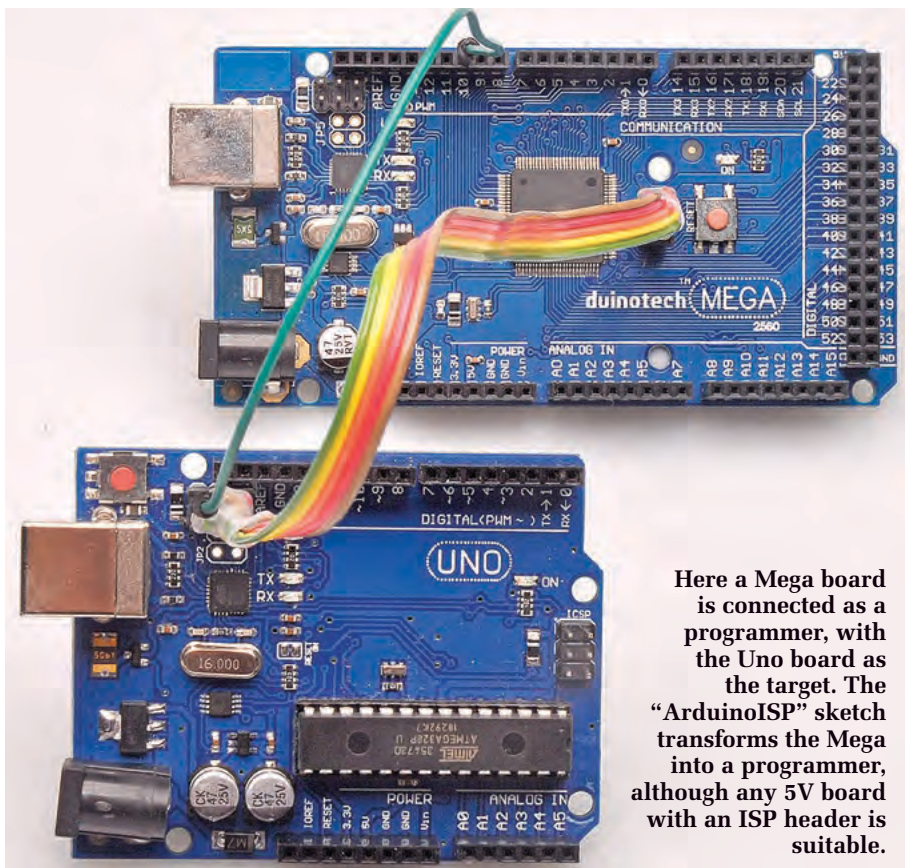
And the MCU selection at top right should match. If you see:

**ERROR: Unknown signature 000000**

```
// Configure the baud rate:

// #define BAUDRATE 19200
#define BAUDRATE 115200
// #define BAUDRATE 1000000
```

Fig.8: this small fragment of the ArduinoISP sketch is where the serial port baud rate is set.



Here a Mega board is connected as a programmer, with the Uno board as the target. The "ArduinoISP" sketch transforms the Mega into a programmer, although any 5V board with an ISP header is suitable.

Then the target processor is not being detected. Check your connections and try again.

To upload the firmware, select the "Write" radio button under the "Flash" heading at upper left and then select the firmware file.

You will have a copy of it hiding somewhere in your Arduino IDE folder (on our system, it was in C:\Program Files (x86)\Arduino\hardware\arduino\avr\firmwares\atmega-xxu2\arduino-usbserial\Arduino-usbserial-atmega16u2-Uno-Rev3.hex).

If you are updating the firmware on the ATmega16u2 installed on a Mega board, you need to use the version with "Mega" in the name instead of "Uno".

To make your life easier, we have included the current version of both files in a download associated with this article on the SILICON CHIP website.

Having selected the file, click "Go" under the Flash section. You should see messages like this in your output window:

**avrdude.exe: verifying ...**  
**avrdude.exe: 4034 bytes of flash verified**  
**avrdude.exe done. Thank you.**

This means that firmware has up-

loaded correctly. Once that's done, under the section labelled "Fuses lock bits" at right, click "Read".

The L, H and E (low, high and extended fuses) values should read 0xFF, 0xD9 and 0xF4 respectively, just like our screenshot. We read these from another working Uno.

If not, change them to match, then click the "Write" button in the same section. We only had to change the low fuse byte on our chip. Once this has completed, the ATmega16u2 is correctly programmed.

You can now unplug the programming cable from the target Uno and connect it to a computer via its USB cable. The ATmega16u2 chip should now show up as a USB Serial Device.

## Reprogramming the ATmega328

You can also use this approach to install or repair the bootloader firmware on the ATmega328. This is necessary, for example, after plugging a new, blank ATmega328 chip into the Uno board.

The arrangement is the same as shown above, except that you connect to the other ICSP header on the target board.

The required file is called "optiboot\_atmega328.hex". Optiboot is the name of the bootloader firmware. We have included this in our .ZIP download to make your life easier.

Once the boards are connected, click the "Detect" button to identify (or manually select) the MCU, write the HEX file to flash and then change the fuse bits.

In this case, they should be 0xFF, 0xDE and 0xFD for the low, high and extended fuse bits respectively. We used AVRDUDESS to read these from another Uno to confirm that they were correct.

Similar firmware files exist for the Leonardo (ATmega32u4) and Mega (ATmega2560) boards and their main processors.

By the way, it's also possible to use an ISP programmer to upload sketch files directly to the ATmega328 on an Uno, bypassing the USB-serial connection.

The connections are the same as for writing the bootloader to the ATmega328 chip. From the Tools menu in the Arduino IDE, select Programmer -> Arduino as ISP. To upload the sketch, press Ctrl-Shift-U or select the Sketch -> Upload Using Programmer menu option.

Note that doing this will corrupt the bootloader settings, so if you want to use the USB-serial link for uploading in the future, you will have to re-instate this using AVRDUDESS, as described above.

## Pre-built ISP programmers

If you don't have a separate Arduino board, or find the above procedure awkward, you can purchase a dedicated Atmel in-circuit serial programmer like Jaycar Cat XC4627.

This comes with a 10-pin cable, but a 10-pin to 6-pin adapter is also available (Cat XC4613). Or use Altronics Cat Z6540, which has sockets for both 10-pin and 6-pin cables.

These programmers may need their own drivers installed, and will have a different programmer type, rather than "Arduino as ISP".

## Conclusion

We used the process described here to resurrect two Uno boards with around \$10 of parts and some time. And we learnt quite a bit about the Arduino system in the process; hopefully, so will you.

SC