MONOCHRON

May 17, 2011 20:07

Updating your clock!

The MONOCHRON was explicitly designed to allow users to create their own firmware for the clock using our code base. Once you have built and tested your clock, you can choose one of our existing designs or, of course, write your own!

AVR Programmer or FTDI cable?

There are two ways to update the clock. One is to reprogram the entire chip using an AVR programmer. The other is to use a bootloader that is pre-programmed onto the chip that allows the chip to re-program itself. An AVR programmer is more powerful: you can really mess with anything on the chip and the entire 32K of memory is available. Using the bootloader is safer: there's no way to mess with the fuse settings (which could brick the chip) but you only get 30K of memory since 2K is used by the bootloader. Not a big deal, but if you are making some massive clock which requires tons of flash space, you may need it

For a lot more information about AVR programmers and bootloaders, I strongly recommend reading this short article

Note that to program an AVR you need an AVR programmer, but to upload using the bootloader you need a computer-serial connection (such as an FTDI cable). Unfortunately, they are not the same device so unless you have both, you should pick one to start with. If you're not a microcontroller wiz, I suggest going with the bootloader (FTDI) method. Its as fast (or faster), allows you to debug as well, and theres virtually no way to damage/brick the chip by messing with the fuses. If you're familiar with microcontroller programming, and you have a programmer, then feel free to go that direction.

Installing programming software

The first thing you must do, no matter which way you go, is to install the software for communicating with the monochron!

Unless you've already done some microcontroller hacking, you should install the AVR development system on your computer. For windows, I suggest WinAVR (see here for my tutorial). For Mac, AVRMacPack seems to be the best choice (see here for my tutorial). For linux, you'll have to do some package installing, which depends a bit on your distro: I suggest googling for the best way for your distro, or if you are willing to do it 'from scratch', my tutorial will take you through step-by-step.

Either way, make sure that when you are done, you can open up a command window or terminal, and type in **avrdude** to get the following. If you get a respose that avrdude "**isn't found**" or "**isn't recognized**" go back and make sure you installed the software properly according to the tutorials!

```
C:\WINXP\system32\cmd.exe

C:\>avrdude
Usage: avrdude [options]
Options:
  -p <partno>                Required. Specify AVR device.
  -b <baudrate>              Override RS-232 baud rate.
  -B <bitclock>              Specify JTAG/STK500v2 bit clock period (us).
  -C <config-file>           Specify location of configuration file.
  -c <programmer>            Specify programmer type.
  -D                         Disable auto erase for flash memory
  -i <delay>                 ISP Clock Delay [in microseconds]
  -P <port>                  Specify connection port.
  -F                         Override invalid signature check.
  -e                         Perform a chip erase.
  -O                         Perform RC oscillator calibration (see AVR053).
  -U <memtype>:r|w|v:<filename>[:format]
                             Memory operation specification.
                             Multiple -U options are allowed, each request
                             is performed in the order specified.
  -n                         Do not write anything to the device.
  -V                         Do not verify.
  -u                         Disable safemode, default when running from a scrip
t.
  -s                         Silent safemode operation, will not ask you if
                             fuses should be changed back.
  -t                         Enter terminal mode.
  -E <exitspec>[,<exitspec>] List programmer exit specifications.
  -y                         Count # erase cycles in EEPROM.
  -Y <number>                Initialize erase cycle # in EEPROM.
  -v                         Verbose output. -v -v for more.
  -q                         Quell progress output. -q -q for less.
  -?                         Display this usage.

avrdude project: <URL:http://savannah.nongnu.org/projects/avrdude>

C:\>_
```

Now you can continue!

Installing the FTDI driver

Since its going to be more common, we'll be covering how to use the FTDI adapter first. Nearly all of this tutorial is just getting everything set up and installed, it only has to happen once!

Step #1 is to plug in your FTDI adatper. If you have an FTDI cable proper, there is already a USB A connector on the end

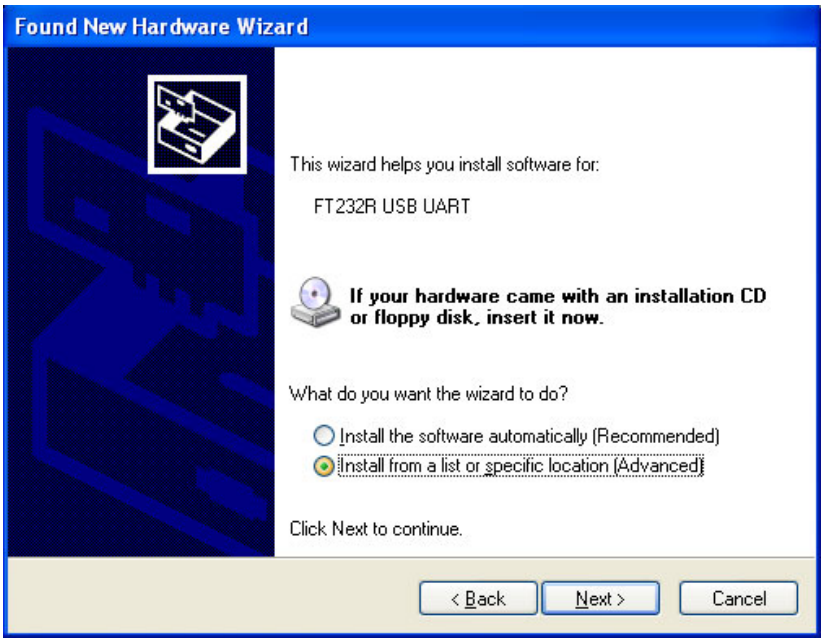
If you have an FTDI adapter, you'll need a standard mini-B cable, pretty much everything uses these so steal your camera's or cell phone's data cable

If you are using windows, you may need to need to download the FTDI driver if you haven't already installed it for another project. If you are using Mac or Linux, the driver is already built in to the operating system (handy!)

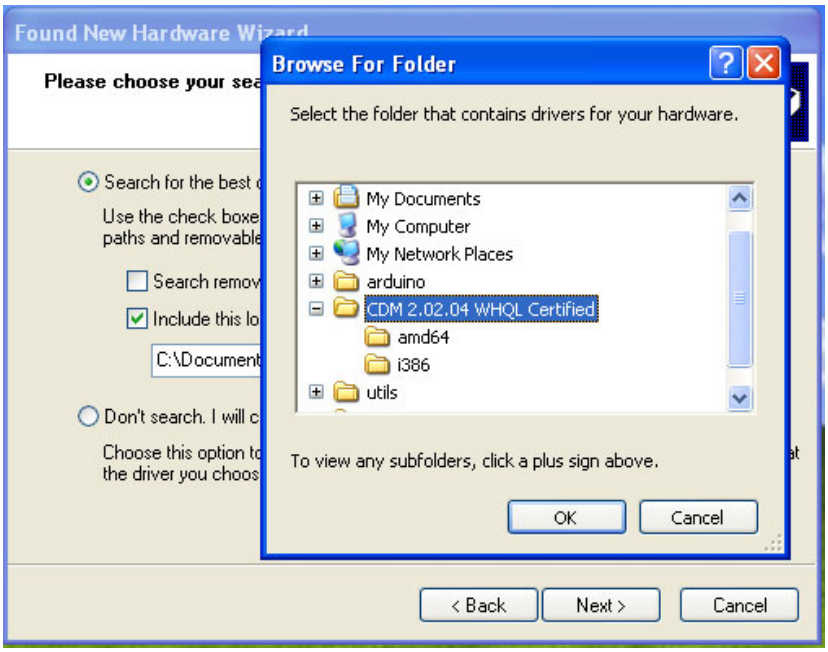
Next up, plug it into your computer! If you are using Windows you may hear a sound from the computer and a little popup bubble in the bottom right corner of the screen that says **Found New Hardware FT232R USB UART**

After a few seconds, the new hardware wizard will start. Select "**No not this time**" and click **Next>**

At the next screen, select **Install from a list or specific location**

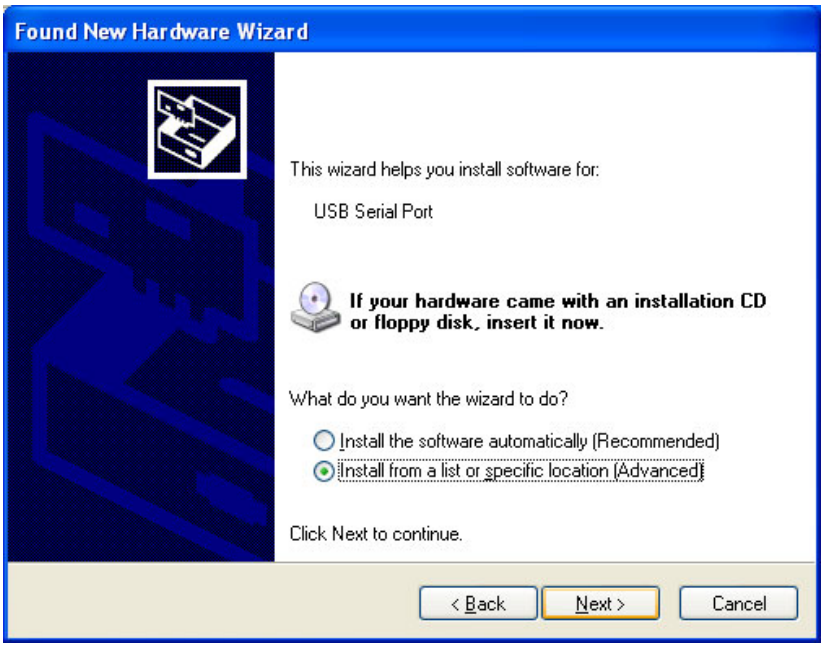


At the next screen make sure **Include this location** is selected and browse to the folder that contains the driver you downloaded. Select the folder and click **OK**

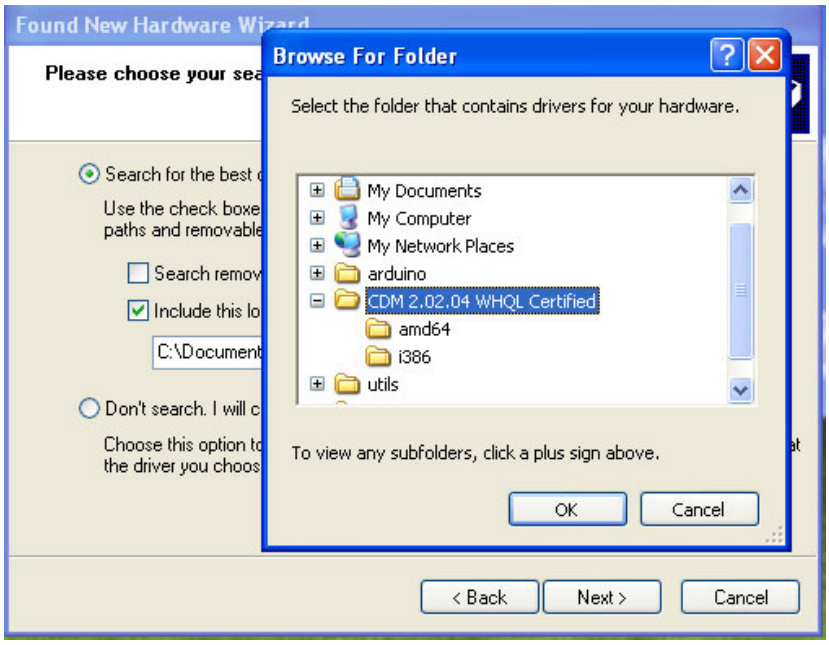


It should copy some files and then come up with this window. Click **Finish**

Almost immediately, another window will pop up, this time it will say **USB Serial Port**. As before, click **Install from a list or specific location**



Browse to the same folder again…



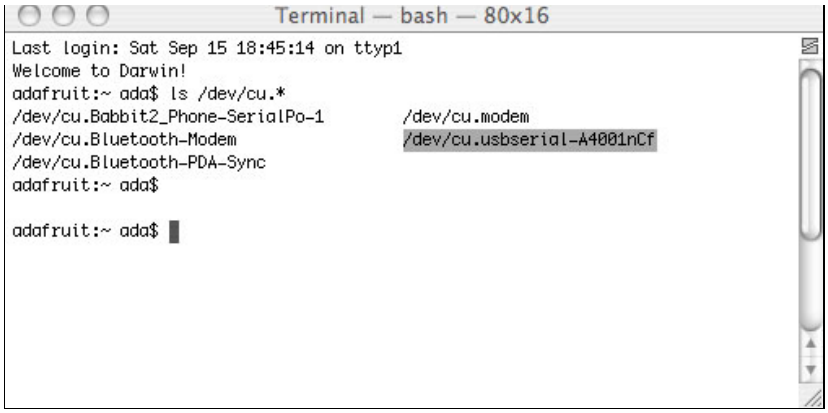
And it should complete successfully!

You may need to reboot the computer.

FTDI name and Set RTS on Close

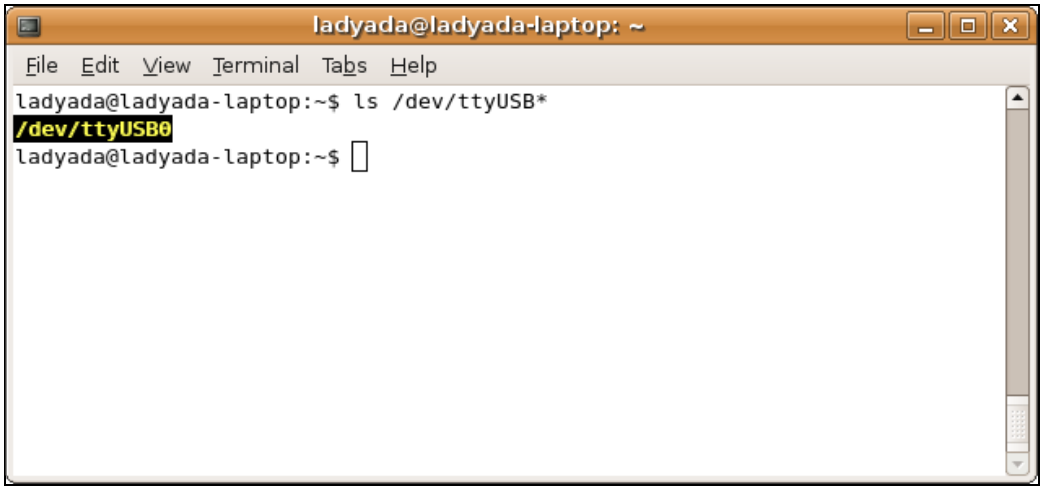
OK now we will verify that the driver installed properly.

Under Mac, in the **Terminal** window, type in **ls /dev/cu.*** which should give the following responses or so

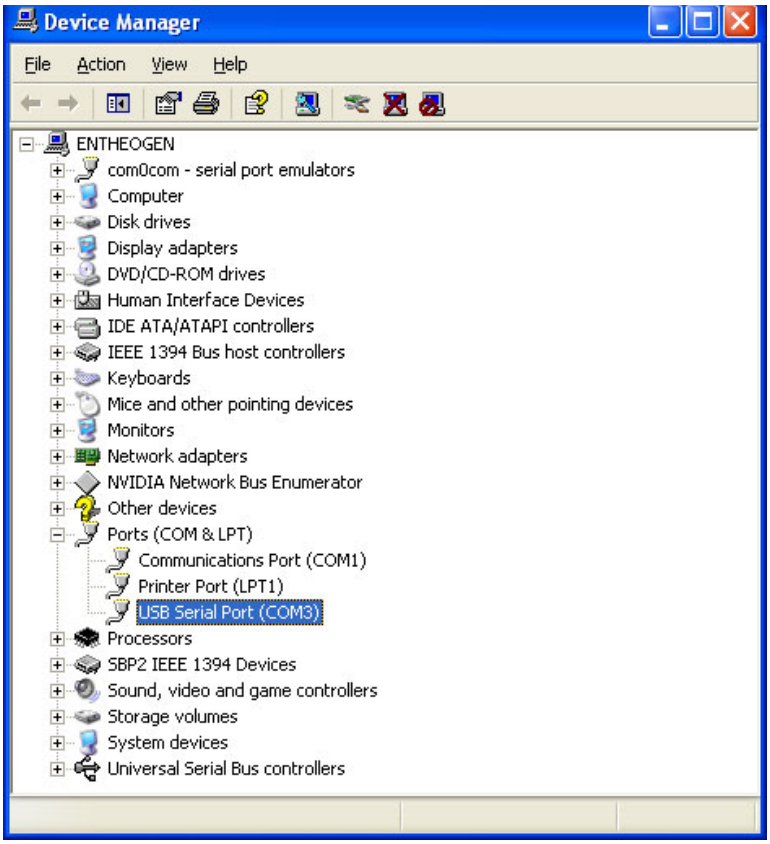


The name we are looking for is **/dev/cu.usbserial-XXXX** where the X's are going to be unique for each cable. Copy and paste the name into a text file so you'll remember it for later.

For Linux/Unix type **ls /dev/ttyUSB*** into a terminal window, you should see a device file called something like ttyUSB0

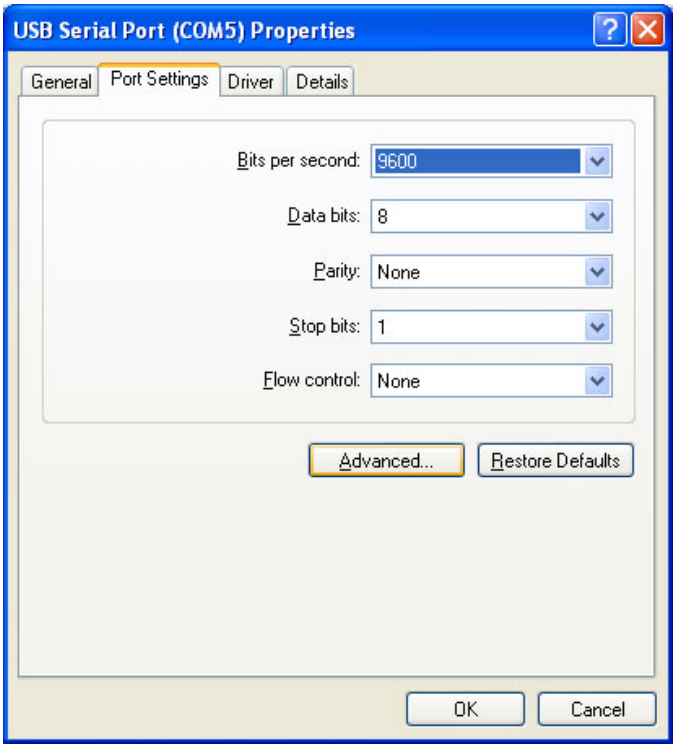


If you are using Windows, go to the **Device Manager** (From the **Start Menu**, select Settings->Control Panel. Double click on **System** and select the **Hardware** tab. Then click on the **Device Manager** button)

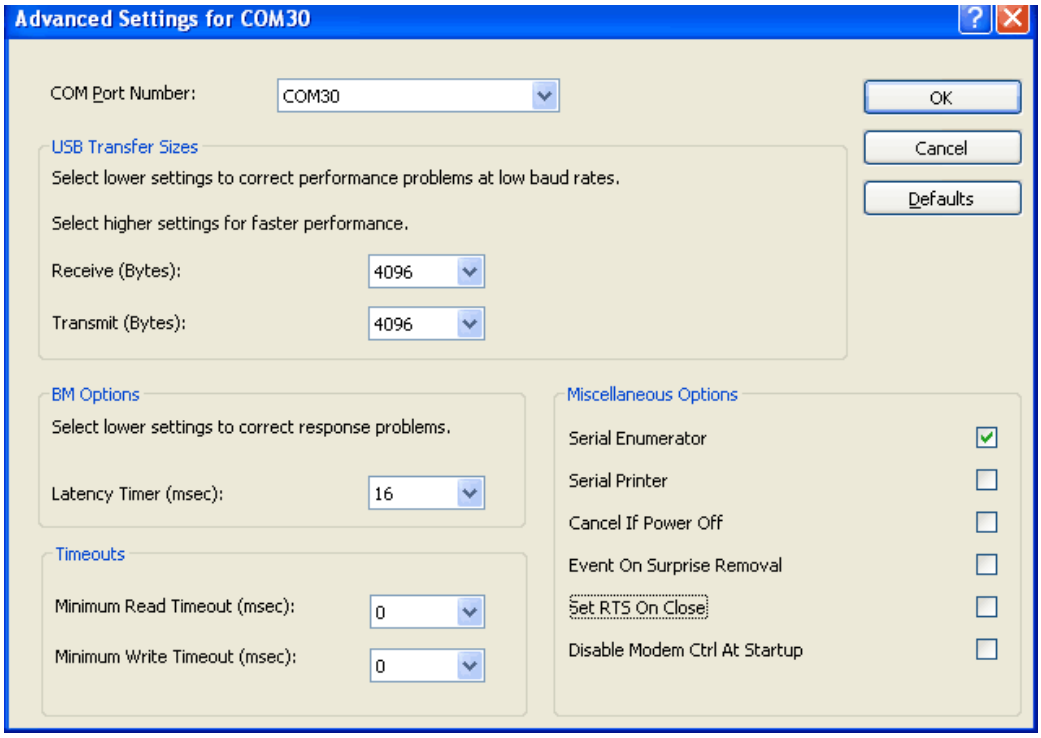


Look for an entry under **Ports (COM & LPT)** that says **USB Serial Port (COM)** the COM number may vary but it should be something like **COM3** or **COM4** the COM number may be as high as **COM99** so just look for the USB serial port. The COM stands for "communication", and each one has a unique number, known as the **COM Port number**. In this case the COM Port number is **COM3**. If you don't see the COM port verify the cable is plugged in, and check that you installed the VCP FTDI driver.

Then right click and select **Properties**



Click on the **Port Settings** tab, and click on **Advanced...**



Make sure **Set RTS On Close** is **not** selected. Then click OK

Whew! OK now you are good to go for the next step

Test!

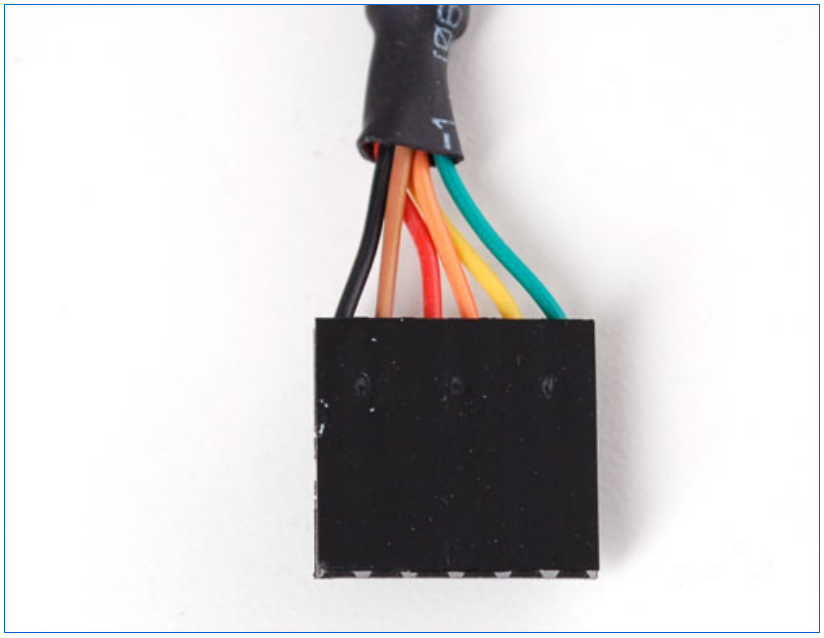
Plug in your clock, and verify its working. Next plug in the FTDI adapter (if its not plugged in yet)

Open up a command window (Windows), or terminal (Mac/Linux/Unix) and type in the following command (dont hit return yet!)

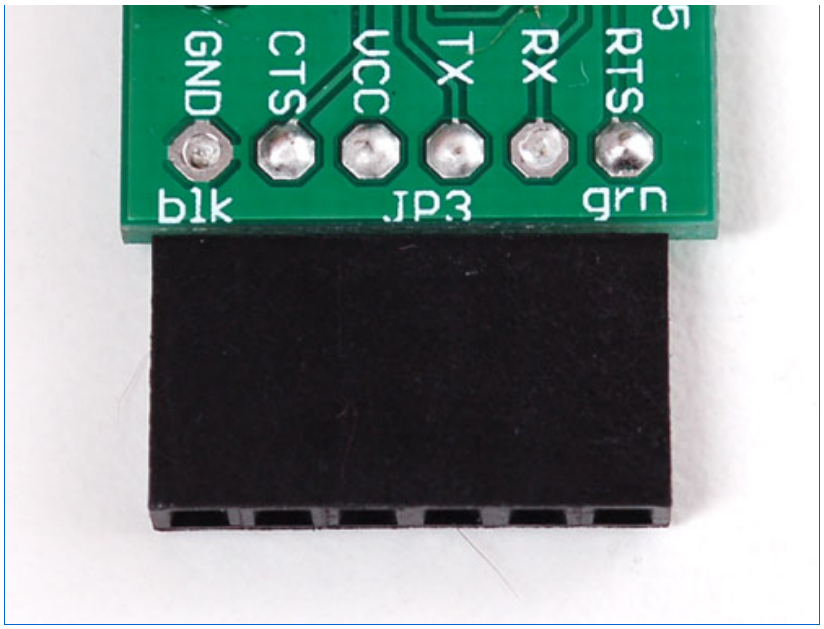
avrdude -c arduino -p m328p -P <COMPORTNAME> -b 57600
*Where **<COMPORTNAME>** is something like **COM3** or **/dev/ttyUSB0***

Remember don't hit return yet!

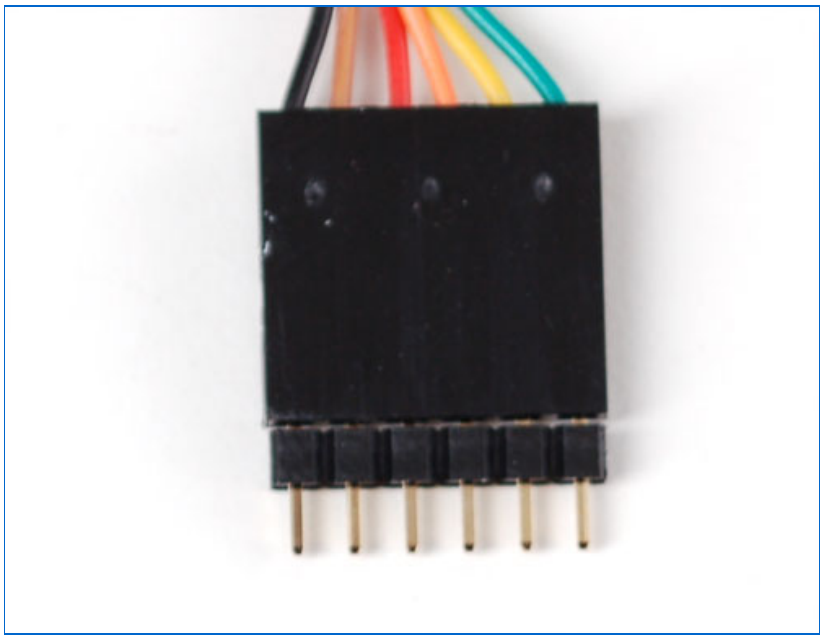
Now look again at your FTDI cable or adapter. If you have a cable you'll notice that one wire is Black.



Likewise on the adapters, there will be text that says BLK (or GND) on one end



Stick a piece of 6 pin header left over from the kit making into the socket so that the shorter pins stick out. The latest kits come with an **extra long** header piece..use that!



Now, plug in the header so that it fits into the back of the clock, so that the BLACK wire lines up with the text that says **blk** You will have to hold the cable so that you end up pressing the header at an angle against the socket, this way you will make contact without having to do any soldering

OK with your other hand, hit the return key. Nothing will happen for a bit, and then you should see something like the following:

```
C:\WINDOWS\system32\cmd.exe
C:\>avrdude -c arduino -P COM30 -p m328p -b 57600
avrdude: AVR device initialized and ready to accept instructions
Reading | ################################################## | 100% 0.02s
avrdude: Device signature = 0x1e950f
avrdude: safemode: Fuses OK
avrdude done.  Thank you.

C:\>_
```

if you get something like

```
C:\WINDOWS\system32\cmd.exe
C:\>avrdude -c arduino -P COM30 -p m328p -b 57600
avrdude: stk500_getsync(): not in sync: resp=0x00

avrdude done.  Thank you.

C:\>_
```

Check that you have the cable in properly and you're holding it right. Then press the up arrow and return to try again.

The most important part is that you will see **AVR device initialized** a bar of **#**'s and then text that says **Device signature** and **Fuses OK** This means you've successfully talked to the bootloader! Yay! Go have a cup of your favorite drink

If you're having problems, go back to the previous step and try to get the clock to reset. If you're having problems still, post up in the forums!

Uploading your favorite clock

Now that you have all that set-up stuff ready, you can get to the fun part. Lets install SevenChron, for example, on our clock.

Go to the clock listing page, and click on the link that says "Code at Github" then click Download Source and save the **zip** or **tar** file onto your computer. Then uncompress it and find the file called **monochron.hex** in the **firmware** subfolder and copy it to your home directory (or to C:\ if you're using Windows).

Back to your command window:

Go to the directory where the file is at. If you're using windows, type in and press return

Go to the directory where the file is at. If you're using windows, type in and press return

cd C:

For mac or linux

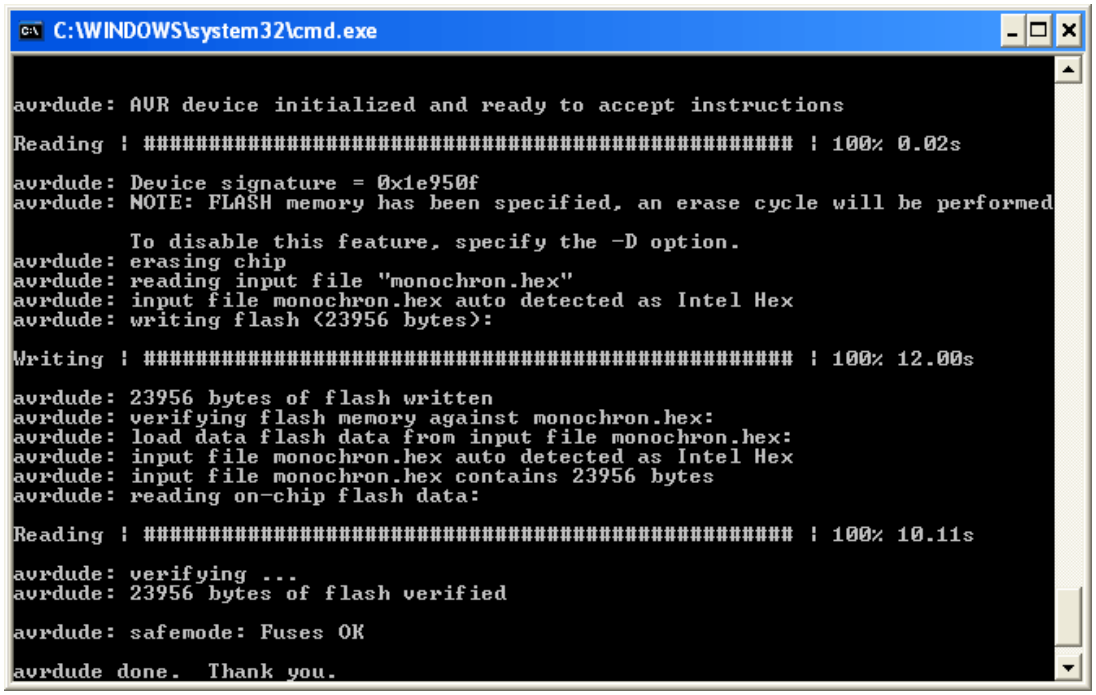
cd ~

Now, we will issue the reprogramming command. Type in the following, dont press return!

For windows: **avrdude -c arduino -p m328p -P <COMPORTNAME> -b 57600 -U flash:w:C:\monochron.hex**

For Mac/Linux: **avrdude -c arduino -p m328p -P <COMPORTNAME> -b 57600 -U flash:w:~/monochron.hex**

Basically, you can hit the up arrow twice and then type in "**-U flash:w:monochron.hex**" at the end. This will **w**rite the **flash** with the new firmware file. Press the FTDI adapter against the clock as before and hit return, keep pressing against the cable for about 30 seconds until the entire process is complete



Thats it! Now every time you want to reprogram the clock, you only have to follow this step.