Instructions for building your monome arc clone

1 *Before you start*

Please note: if you encounter a problem, please **do not post on the monome forums**! Tehn will surely help you, but I don't want my clone to cause them trouble and work. I'm very happy that they make their protocols open and I would like them to continue to do so in the spirit of openness.

For assistance, please write an email to my address instead: mail@johannesneumann.net I will try to help you out.

2 *Required tools*

During assembly and setup you will need the following tools

- AVR ISP programming device (supported by avrdude). I recommend USBasp or any of its clones. Alternatively you can also use STK500v2 clones or any other programmer that supports the ISP protocol. You get something like this on ebay for under 20€.
- Soldering iron, decent(!) flux, tweezers, solder wick
- a windows computer or a windows virtual machine (needed to setup the FTDI chip)
- a multimeter

3 *Software installation*

1. Before you start, please install all the software that's needed to run monome devices. Please look at the instructions provided by http://monome.org
   Even though you might not have a functioning monome at this point, we will need the drivers during the setup process.
2. Install the drivers for your AVR ISP programmer, if you haven't already.
3. Install *WINAVR* from this link. This will install a AVR compiler and (more important) a tool called *avrdude* to upload the software to the ATMega88p microprocessor used in the device.
4. Install *AVR-Burn-O-Mat* from this link. This little tool will make it easier to upload the software to the chip as it provides a nice graphical user interface to *avrdude* installed with WINAVR.
5. Now we need to edit some files. Unfortunately as I am writing this (05.11.2014) it seems that some versions of *avrdude* and *AVR-Burn-O-Mat* don't know the ATMega88p processor yet. So we teach them that it exists.

   First, go to C:\WINAVR.......\bin\avrdude.conf and open it with a text editor (I recommend geany).

   Search for "ATMega88p". If it takes you to something like this ...

   ```
   #------------------------------------------------------------
   # Atmega88p
   #------------------------------------------------------------
   ```

… you don't have to edit anything and the file is fine. Simply go to the next step.

If nothing is found, search for this part:

```
#------------------------------------------------------------
# ATmega88
#------------------------------------------------------------
```

This is the header for the (almost) equal ATMega88. We simply copy the whole passage below this line (down to the next similar looking header) and change a few things. The whole passage you have to copy is roughly 200 lines long.

Duplicate the whole passage and change the lines highlighted with yellow:

```
#------------------------------------------------------------
# ATmega88p
#------------------------------------------------------------

part
    id                = "m88p";
    desc              = "ATMEGA88P";
     has_debugwire = yes;
     flash_instr   = 0xB6, 0x01, 0x11;
     eeprom_instr  = 0xBD, 0xF2, 0xBD, 0xE1, 0xBB, 0xCF, 0xB4, 0x00,
                    0xBE, 0x01, 0xB6, 0x01, 0xBC, 0x00, 0xBB, 0xBF,
                    0x99, 0xF9, 0xBB, 0xAF;
    stk500_devcode    = 0x73;
#    avr910_devcode    = 0x;
    signature         = 0x1e 0x93 0x0f;
    pagel             = 0xd7;
    bs2               = 0xc2;

…
… more stuff follows here …
…
```

Now *avrdude* knows the microprocessor.

6. Now we have to teach *AVR-Burn-O-Mat* as well. Go to the installation directory (mine is: D:\Program Files (x86)\AVR Burn-O-Mat) and open the file *AVR8_Burn_O_Mat_Config.xml.* With a text editor (e.g. geany).

   Again, search for "Atmega88p". If nothing is found, search for "Atmega88". It should take you to this line:

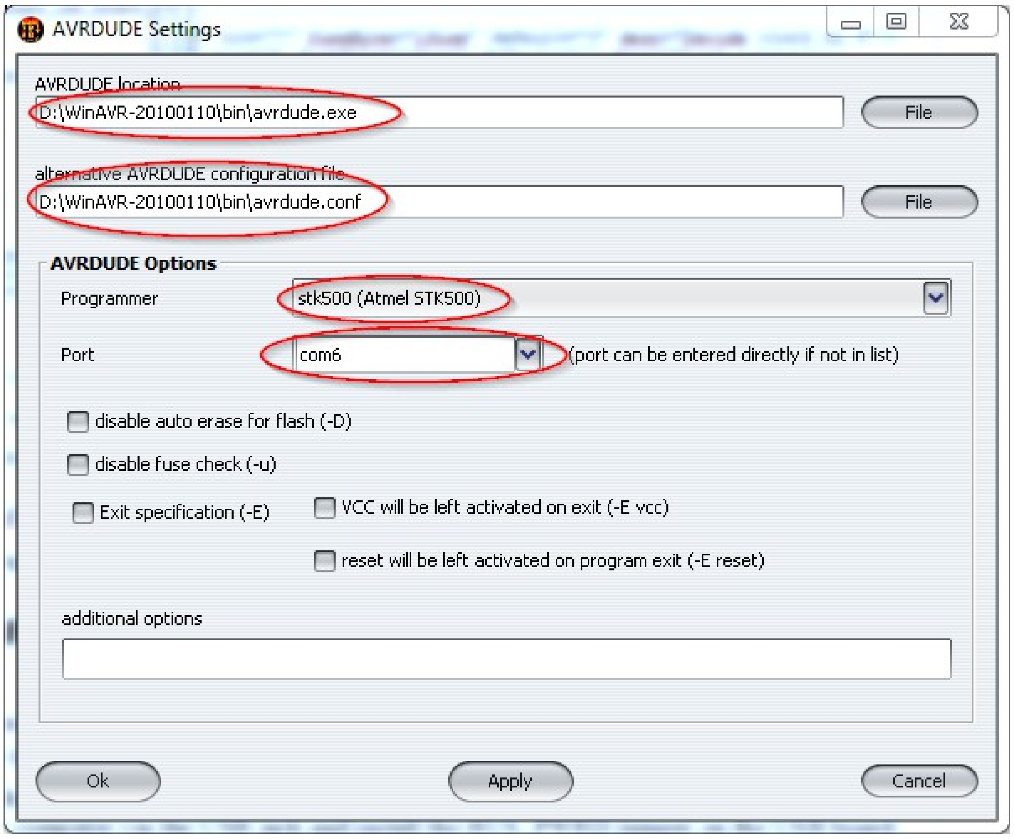
```
<AVR name="m88" caption="ATmega88">
      <Fuse name="UNUSED_E7" bit="7" …
…

…
```

   Copy the whole block below this and make the changes highlighted:

```
<AVR name="m88p" caption="ATmega88p">
      <Fuse name="UNUSED_E7" bit="7" fuseByte="efuse" default="1" desc="unused"
mode="expert"/>
      <Fuse name="UNUSED_E6" bit="6" fuseByte="efuse" default="1" desc="unused"
mode="expert"/>
      <Fuse name="UNUSED_E5" bit="5" fuseByte="efuse" default="1" desc="unused"
mode="expert"/>
      <Fuse name="UNUSED_E4" bit="4" fuseByte="efuse" default="1" desc="unused"
mode="expert"/>
      <Fuse name="UNUSED_E3" bit="3" fuseByte="efuse" default="1" desc="unused"
mode="expert"/>
      <Fuse name="BOOTSZ1" bit="2" fuseByte="efuse" default="0" desc="Select boot size"/>
      <Fuse name="BOOTSZ0" bit="1" fuseByte="efuse" default="0" desc="Select boot size"/>
      <Fuse name="BOOTRST" bit="0" fuseByte="efuse" default="1" desc="Select reset vector"/>
      <Fuse name="RSTDISBL" bit="7" fuseByte="hfuse" default="1" desc="Select if PC6 is I/O
pin or RESET pin" mode="expert"/>
      <Fuse name="DWEN" bit="6" fuseByte="hfuse" default="1" desc="debugWIRE enable"
mode="expert"/>
      <Fuse name="SPIEN" bit="5" fuseByte="hfuse" default="0" desc="Enable Serial Program
and Data Downloading" mode="expert"/>
      <Fuse name="WDTON" bit="4" fuseByte="hfuse" default="1" desc="Watchdog timer always
on"/>
      <Fuse name="EESAVE" bit="3" fuseByte="hfuse" default="1" desc="EEPROM memory is
preserved through the Chip Erase"/>
      <Fuse name="BODLEVEL2" bit="2" fuseByte="hfuse" default="1" desc="Brown out detector
trigger level"/>
      <Fuse name="BODLEVEL1" bit="1" fuseByte="hfuse" default="1" desc="Brown out detector
trigger level"/>
      <Fuse name="BODLEVEL0" bit="0" fuseByte="hfuse" default="1" desc="Brown out detector
trigger level"/>

      <Fuse name="CKDIV8" bit="7" fuseByte="lfuse" default="0" desc="Devide clock by 8"/>
      <Fuse name="CKOUT" bit="6" fuseByte="lfuse" default="1" desc="Clock output"/>
      <Fuse name="SUT1" bit="5" fuseByte="lfuse" default="1" desc="Select start-up time"/>
      <Fuse name="SUT0" bit="4" fuseByte="lfuse" default="0" desc="Select start-up time"/>
      <Fuse name="CKSEL3" bit="3" fuseByte="lfuse" default="0" desc="Select Clock source"/>
      <Fuse name="CKSEL2" bit="2" fuseByte="lfuse" default="0" desc="Select Clock source"/>
      <Fuse name="CKSEL1" bit="1" fuseByte="lfuse" default="1" desc="Select Clock source"/>
      <Fuse name="CKSEL0" bit="0" fuseByte="lfuse" default="0" desc="Select Clock source"/>

      <BrownOutDetection>
        <Fuse name="BODLEVEL2"/>
        <Fuse name="BODLEVEL1"/>
        <Fuse name="BODLEVEL0"/>
        <Setting caption="disabled"           Fuse1="1" Fuse2="1" Fuse3="1"/>
        <Setting caption="1.8V (1.7V - 2.0V)" Fuse1="1" Fuse2="1" Fuse3="0"/>
        <Setting caption="2.7V (2.5V - 2.9V)" Fuse1="1" Fuse2="0" Fuse3="1"/>
        <Setting caption="4.3V (4.1V - 4.5V)" Fuse1="1" Fuse2="0" Fuse3="0"/>
      </BrownOutDetection>

      <OscillatorOptions Type="ATmega48" />

    </AVR>
```

Now *AVR-Burn-O-Mat* can program our processor.

7. Now open *AVR-Burn-O-Mat* and go to the settings page. Check if the directories are correct and enter the connection settings for your programmer.


AVRDUDE Settings

AVRDUDE location
D:\WinAVR-20100110\bin\avrdude.exe

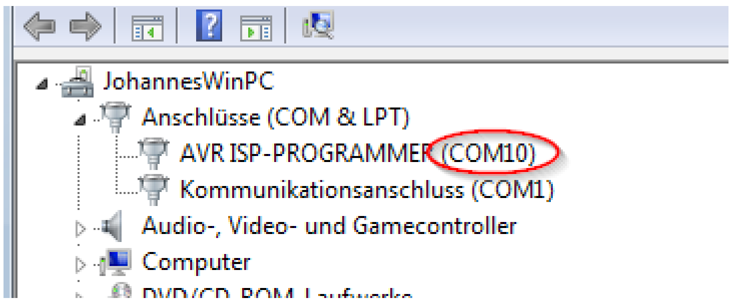
alternative AVRDUDE configuration file
D:\WinAVR-20100110\bin\avrdude.conf

AVRDUDE Options
Programmer: stk500 (Atmel STK500)
Port: com6 (port can be entered directly if not in list)

- [ ] disable auto erase for flash (-D)
- [ ] disable fuse check (-u)
- [ ] Exit specification (-E)
- [ ] VCC will be left activated on exit (-E vcc)
- [ ] reset will be left activated on program exit (-E reset)

additional options

8. I'm using an STK500V2 clone here, but if you click the dropdown menu, it will give you other options, such as USBasp. For most USB programmers, the Port must be set to “usb” or (if they are based on a serial COM interface) the respective COM port. You can open the device manager and plug in your programmer, then it should display the COM port in brackets behind the device name.

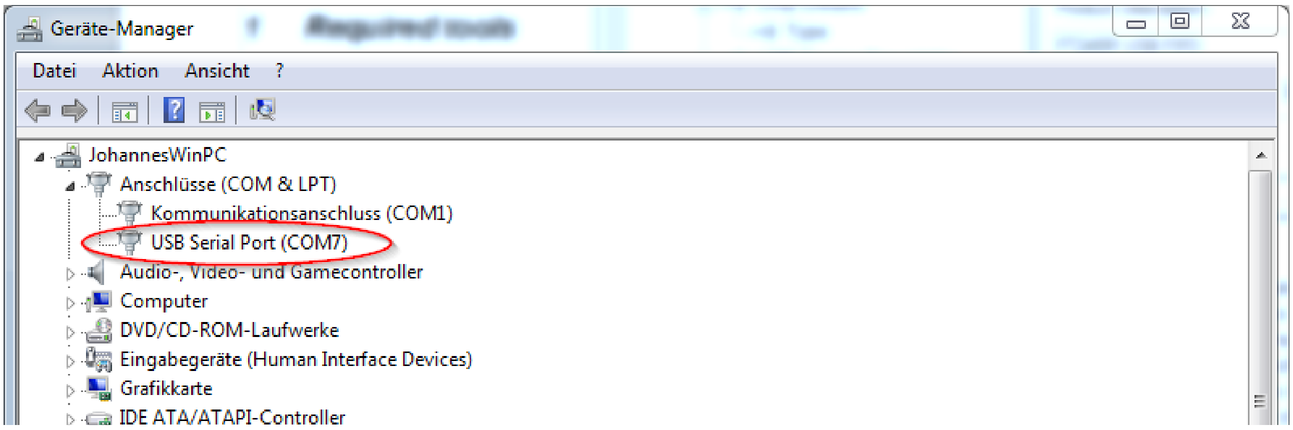

JohannesWinPC
Anschlüsse (COM & LPT)
AVR ISP-PROGRAMMER (COM10)
Kommunikationsanschluss (COM1)
Audio-, Video- und Gamecontroller
Computer
DVD/CD-ROM-Laufwerke

9. Install FT-Prog from this website. We will need it to configure the FTDI USB chip on the board.

4 USB Board assembly

I won't give step-by-step instructions for soldering. If you are not familiar with soldering and electronics in general, please do an easier project first. It can become very frustrating otherwise.

Once you're done assembling the USB board, go to the windows device manager. Connect the USB board to your computer via the USB jack and install the BUS_PWRD jumper on the USB board. You should see the device manager refresh and a new COM port device will appear.

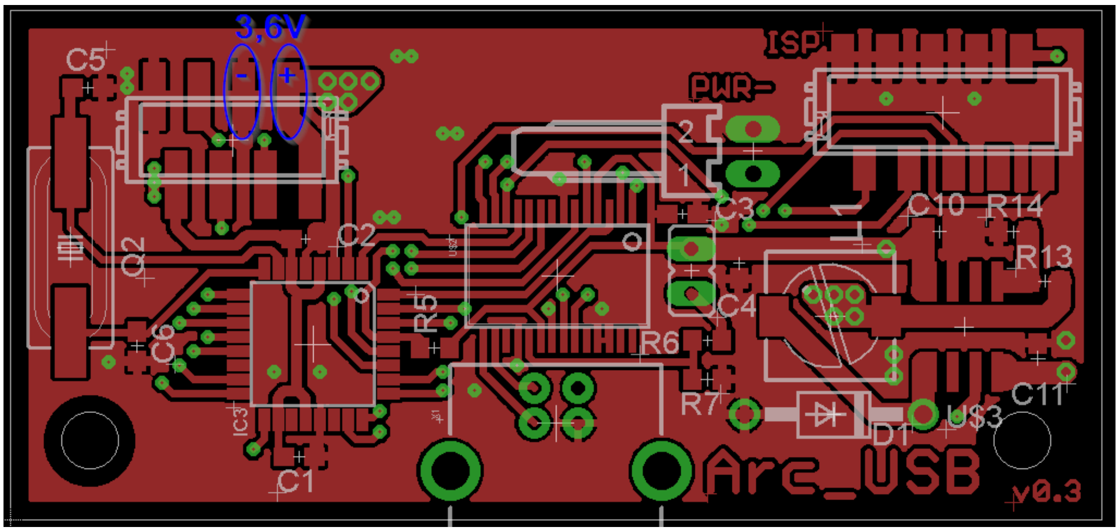

Geräte-Manager

Datei Aktion Ansicht ?

- JohannesWinPC
  - Anschlüsse (COM & LPT)
    - Kommunikationsanschluss (COM1)
    - USB Serial Port (COM7)
  - Audio-, Video- und Gamecontroller
  - Computer
  - DVD/CD-ROM-Laufwerke
  - Eingabegeräte (Human Interface Devices)
  - Grafikkarte
  - IDE ATA/ATAPI-Controller

If it does not appear, please disconnect it immediately and check your soldering!

Now we have to check the voltage of the on-board regulator. Connect a multimeter to the pins highlighted in the following picture:

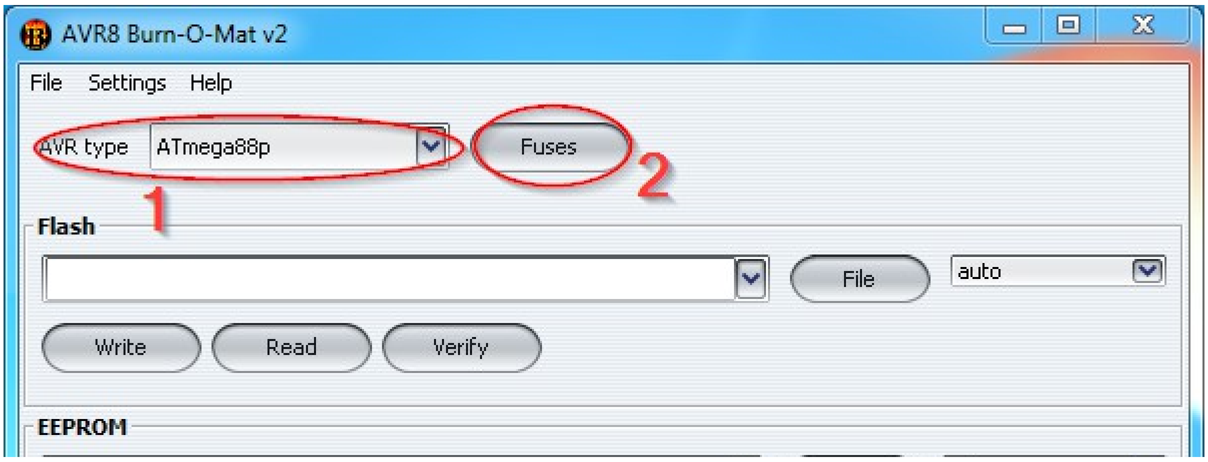


Your meter should read 3,6V (or thereabouts). Fine, it looks like everything is okay.

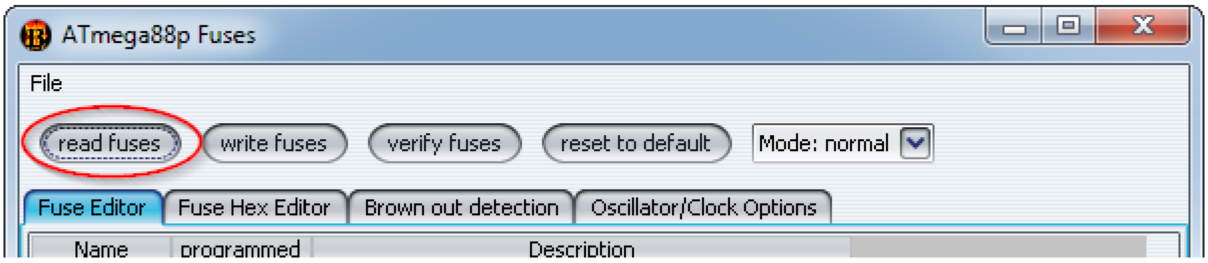
Lets go on and flash the firmware to the chip. Open *AVR-Burn-O-Mat* and connect your programmer to your PC. Then connect your programmer to the 10-pin connector on the USB board. This connector is not the one commonly used for AVR devices, but it uses the same pinout and attaches to the same type of cables. You can simply use your programmers 10-pin cable and crimp micromatch connector to it. This should now be plugged into the USB board. Check the orientation. A good help can be the four ground-pins, which are connected to each other. On both the USB board and the programmer. Pin 1 of the cable is on the opposite side. In the image above, it is the pin on the bottom right of the connector, next to the “R14” label.

Also be sure, that the USB board is plugged into your computer as well and the BUS_PWRD jumper is installed, so the processor receives power.

First, we need to change the fuse-bits in the processor, to use the external 20MHz crystal as a clock source. Select Atmega88p from the “AVR type” menu. Then click on “Fuses”



In the new window, click “read fuses”.

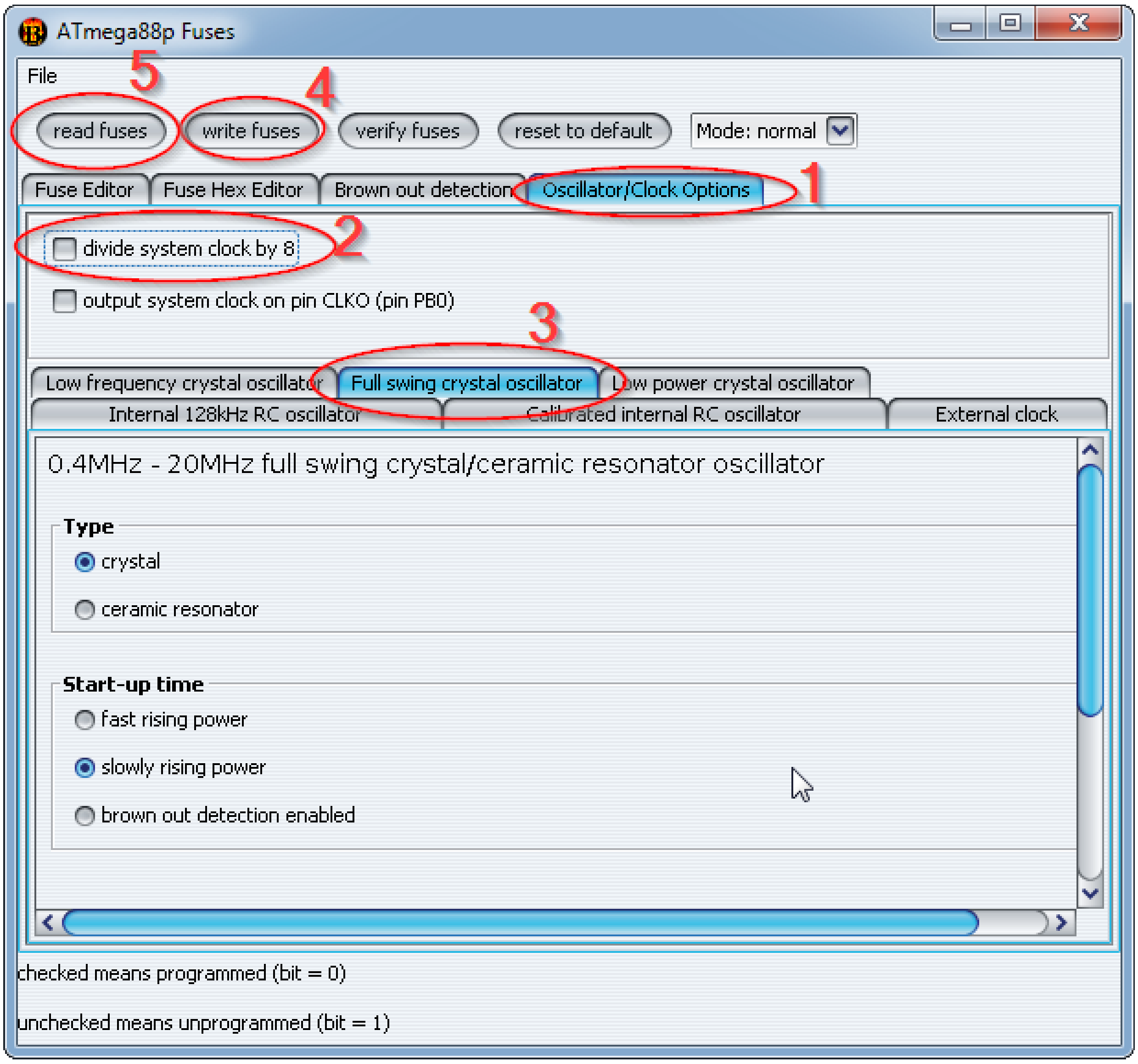


You should see some commands run in the bottom of the main window and eventually a message will pop up, saying: “fuses successfully read”. If not: Check the commands in the main window. They can give you a hint to where the problem is. Usually the programmer settings are wrong (wrong COM port...) or the USB board is not connected properly. Some programmers offer a “low speed” programming mode, this should be used, as the processor is currently still running at its internal low clock speed.

If verything is fine, we can edit the fuse-bits to use the external crystal. Click on the “Oscillator/Clock Options” tab, then un-tick “divide system clock by 8” and select the ”Full swing crystal oscillator” tab.

Double check everything! You might screw your chip up! Never attempt to write the fuses, if you were not able to correctly read them before. You might end up with a chip that has its programming interface disabled and can't be rescued even with professional equipment.

Okay, once you are sure, everything is correct and matches the image below, you can click “write fuses” and again, some commands should run in the main window and a message should say: “Fuses successfully written”. Then click “read fuses” again. If they can be read successfully, you have done everything right.


ATmega88p Fuses

File

5
read fuses
4
write fuses
verify fuses
reset to default
Mode: normal
Fuse Editor
Fuse Hex Editor
Brown out detection
Oscillator/Clock Options
1
2
3
Low frequency crystal oscillator
Full swing crystal oscillator
Low power crystal oscillator
Internal 128kHz RC oscillator
Calibrated internal RC oscillator
External clock

- [ ] divide system clock by 8
- [ ] output system clock on pin CLKO (pin PB0)

0.4MHz - 20MHz full swing crystal/ceramic resonator oscillator

Type

[✓ crystal]
ceramic resonator

Start-up time

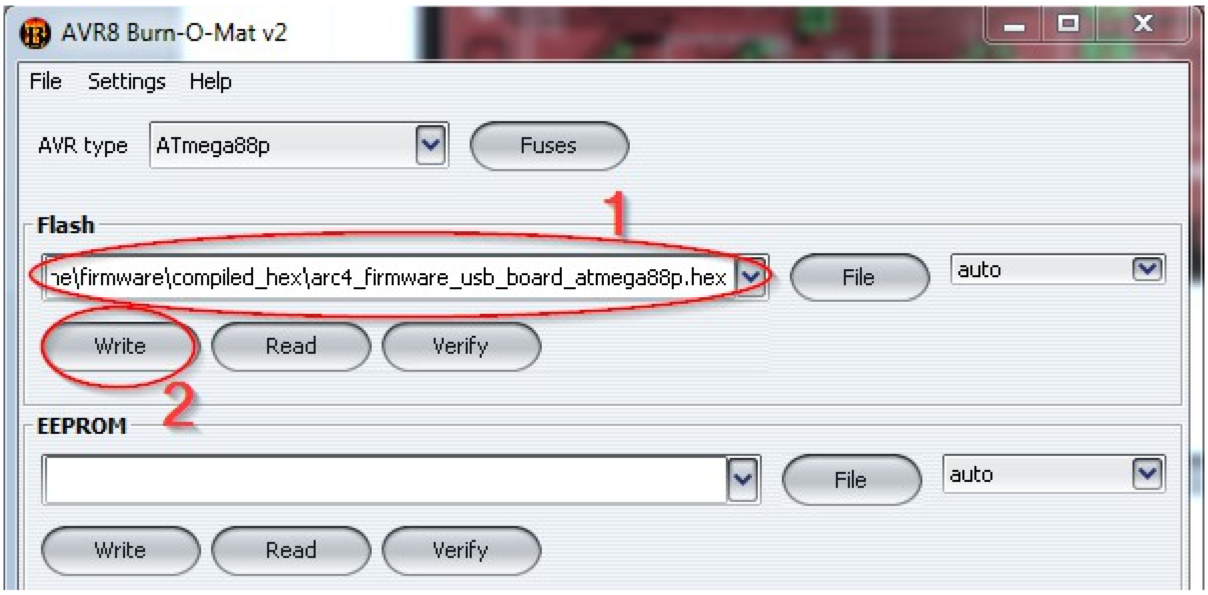
fast rising power
[✓ slowly rising power]
brown out detection enabled

checked means programmed (bit = 0)

unchecked means unprogrammed (bit = 1)

Now we can flash the firmware. Close the fuses window and select the firmware file in the main window. I recommend to use the pre-compiled one in the repository. I want to build an arc 4 here, so I chose the matching firmware. Be sure to use the right file. Using another file will probably make the processor “think” its connected to some hardware which is not there. This can create short circuits and might destroy the chip. So please double check here!

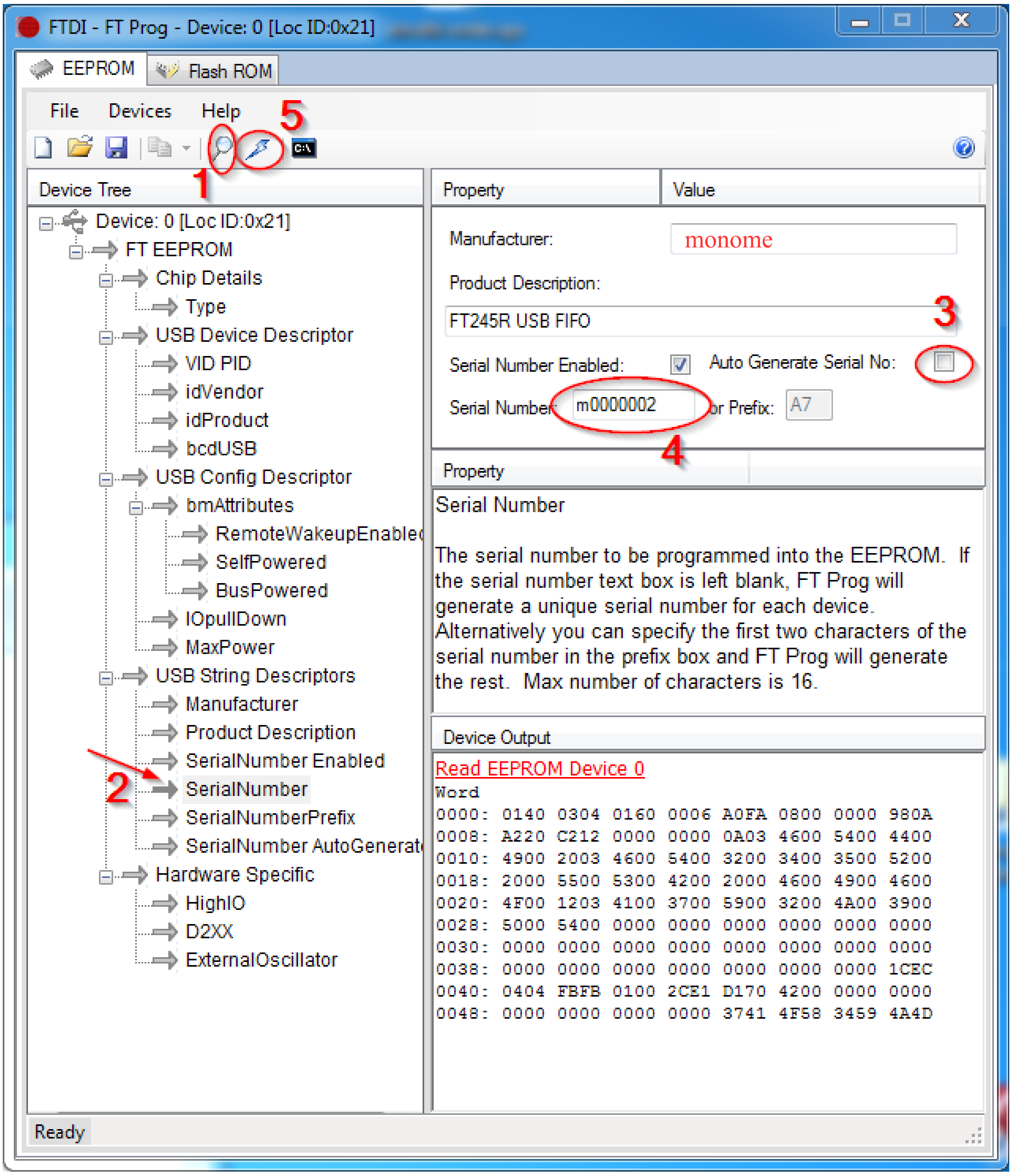
Click “write” and the commands in the bottom of the window should start executing again. Eventually it will say “Flash sucessfully written.”



Nice! Now you can unplug the programmer.

Now we have to configure the USB-chip on the board to be recognized by serialosc so you can use your clone just like a real monome arc.

Start FT_PROG and connect the USB board to your computer via its USB connector. In the FT_PROG window, click the search icon until it displays your device.



Manufacturer: monome

Product Description: FT245R USB FIFO

Serial Number Enabled: ✓ Auto Generate Serial No: ☐

Serial Number: m0000002 or Prefix: A7

Property

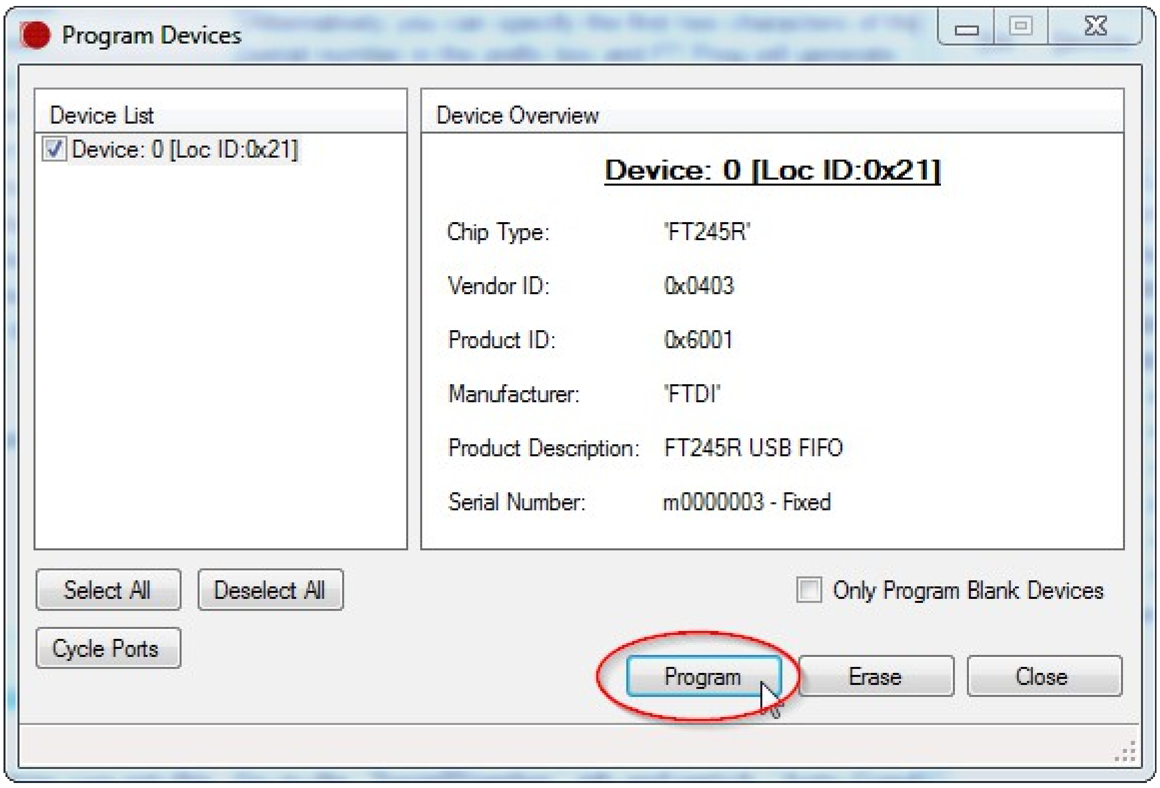
Serial Number

The serial number to be programmed into the EEPROM. If the serial number text box is left blank, FT Prog will generate a unique serial number for each device. Alternatively you can specify the first two characters of the serial number in the prefix box and FT Prog will generate the rest. Max number of characters is 16.

Device Output

```
Read EEPROM Device 0
Word
0000: 0140 0304 0160 0006 A0FA 0800 0000 980A
0008: A220 C212 0000 0000 0A03 4600 5400 4400
0010: 4900 2003 4600 5400 3200 3400 3500 5200
0018: 2000 5500 5300 4200 2000 4600 4900 4600
0020: 4F00 1203 4100 3700 5900 3200 4A00 3900
0028: 5000 5400 0000 0000 0000 0000 0000 0000
0030: 0000 0000 0000 0000 0000 0000 0000 0000
0038: 0000 0000 0000 0000 0000 0000 0000 1CEC
0040: 0404 FBFB 0100 2CE1 D170 4200 0000 0000
0048: 0000 0000 0000 0000 3741 4F58 3459 4A4D
```

Eventually after a few retries you get this. Go to the “SerialNumber” tab and untick “Auto Generate Serial No:”. Now you can edit the “serial number” field. Enter a serial number like in the image above. It should start with an “m” and the numbers after that are irrelevant, but I found that they should be 7 digits long. They represent the “unique number” of your device. If you have other monomes, be sure that you use a different serial number that the one of your other devices. Also change the manufacturer to “monome”, this is important for compatibility with monome eurorack modules and the Aleph. Once you're done, click the flash symbol and program the new settings to the chip.



☑ Device: 0 [Loc ID:0x21]

Device: 0 [Loc ID:0x21]

Chip Type:	'FT245R'
Vendor ID:	0x0403
Product ID:	0x6001
Manufacturer:	'FTDI'
Product Description:	FT245R USB FIFO
Serial Number:	m0000003 - Fixed

Now you can open a monome app and you should see your clone appear in the device list. Wooohoooooo!

If it does not appear, check your serialosc settings.

5 Pot board and ring board assembly

Assemble the ring boards and the pot boards. Start with one pair of them to find errors early.

Use the 8pin micromatch connectors and ribbon cables to connect the pot board to the USB board via their micromatch connectors (Not the one marked with ISP, the smaller one!). Connect the USB board to your computer and check the BUS_PWRD jumper. The USB board should now receive power, and supply the pot board via the ribbon cable that connects the two.

Then attach your programmer to the pot board and repeat the steps done under “usb board assembly”. You should set the fuses the same way and flash the firmware the same way. **But use the pot board firmware file this time!**

Once the pot boards processor is programmed with the right fuses and firmware, you should disconnect the usb cable and set the address selection switch to off/off. This equals to address 0, which is used for the first LED ring in the monome arc clone.

Now attach the led ring.

Connect the USB board to the computer again and the led ring should light up for a short time (part of the startup animation).Open the monome arc test app and check everything.

Then assemble the other boards and connect them to the ribbon cable. Flash them the same way and check them with the test app. Be sure to use the right addresses:

Ring number	Switch 2	Switch 1
0	Off	Off
1	Off	On
2	On	Off
3	On	On

6 That's it, you're done :) Happy music making!