

Firmware Build Notes

WARNING: THE INSTRUCTIONS BELOW HAVE THE POTENTIAL TO CORRUPT YOUR ARDUINO IDE INSTALLATION. PROCEED AT YOUR OWN PERIL! OF COURSE, IF YOU REALLY MESS THINGS UP YOU CAN ALWAYS UNINSTALL AND REINSTALL THE IDE.

NOTE: These instructions were developed and tested using the Arduino 1.6.4 IDE. I have no idea whether they will work for other versions of the IDE.

The Frequency Reference firmware is built using the Arduino IDE, however there are a couple of complications. These complications arise because Arduino generally expects the ATmega328p to use a 16 MHz clock. For this project the clock is set to 20 MHz. That means that things like baud rate and timing (millis(), micros(), delay(), delayMicroseconds()) will not work as expected.

The IDE accommodates other clock speeds using the parameters defined in the boards.txt file. This file specifies the clock speed for each type of board. Unfortunately, none of them have a 20 MHz clock. What we need to do is define a new board type with a 20 MHz clock. Since the hardware for this project looks a lot like an Arduino UNO we will make a copy of the UNO parameters and call it Arduino UNO-20.

In the boards.txt file find this:

```
#####  
uno.name=Arduino Uno  
uno.vid.0=0x2341  
uno.pid.0=0x0043  
uno.vid.1=0x2341  
uno.pid.1=0x0001  
uno.vid.2=0x2A03  
uno.pid.2=0x0043  
  
uno.upload.tool=avrdude  
uno.upload.protocol=arduino  
uno.upload.maximum_size=32256  
uno.upload.maximum_data_size=2048  
uno.upload.speed=115200  
  
uno.bootloader.tool=avrdude  
uno.bootloader.low_fuses=0xFF  
uno.bootloader.high_fuses=0xDE  
uno.bootloader.extended_fuses=0x05  
uno.bootloader.unlock_bits=0x3F  
uno.bootloader.lock_bits=0x0F  
uno.bootloader.file=optiboot/optiboot_atmega328.hex  
  
uno.build.mcu=atmega328p
```

```
uno.build.f_cpu=16000000L
uno.build.board=AVR_UNO
uno.build.core=arduino
uno.build.variant=standard
```

Duplicate the section, and then modify the duplicate so it looks like this:

```
#####
```

```
uno20.name=Arduino Uno-20
```

```
uno20.vid.0=0x2341
uno20.pid.0=0x0043
uno20.vid.1=0x2341
uno20.pid.1=0x0001
uno20.vid.2=0x2A03
uno20.pid.2=0x0043
```

```
uno20.upload.tool=avrdude
uno20.upload.protocol=arduino
uno20.upload.maximum_size=32256
uno20.upload.maximum_data_size=2048
uno20.upload.speed=115200
```

```
uno20.bootloader.tool=avrdude
uno20.bootloader.low_fuses=0xFF
uno20.bootloader.high_fuses=0xDE
uno20.bootloader.extended_fuses=0x05
uno20.bootloader.unlock_bits=0x3F
uno20.bootloader.lock_bits=0x0F
uno20.bootloader.file=optiboot/optiboot_atmega328.hex
```

```
uno20.build.mcu=atmega328p
uno20.build.f_cpu=20000000L
uno20.build.board=AVR_UNO
uno20.build.core=arduino
uno20.build.variant=standard
```

I have highlighted the differences between the two sections. Once you have made the changes and saved the file back to its original location, you need to restart the IDE for the change to take effect.

If all goes well, the new Arduino UNO-20 should be available in the list of boards under the Tools->Board: menu. Choose it for this project hardware. This one change actually gets us 95% of what we want. It allows the baud rate to be set correctly and the timing functions to be mostly correct. (There's more than appears going on here, but for now just accept that it works for this application with Arduino IDE 1.6.4 and presumably newer.)

Because there is no bootloader in the ATmega328p as it ships from the factory, you will need to program it the first time using an ISP. I use the AVR Pocket Programmer from SparkFun plus the pogo-pin adapter. It's pretty cheap and it works well for me. After choosing the board, Arduino UNO-20 under Tools->Board:, and the appropriate programmer, USBtinyISP under Tools->Programmer: (for the SparkFun AVR Pocket Programmer), burn the bootloader with the command Tools->Burn Bootloader then upload the application with the command File->Upload Using Programmer.

After uploading the application with the ISP, the Frequency Reference is fully functional. You can connect to it using a terminal emulator (or the Serial Monitor in the Arduino IDE) to send commands and see status messages from the device. However there is one thing that you cannot do, that is reprogram the device using the serial port. If you have no desire to modify the software or you are comfortable using the ISP every time, then you can stop here.

Going Further

The reason why you cannot upload over serial is that the bootloader specified above was built using the assumption of a 16 MHz clock. To make serial programming work, you need a new bootloader that has been built for 20 MHz. You can find a precompiled bootloader for 20 MHz here:

<http://www.grozeaion.com/electronics/arduino/155-overclocking-atmega328p>

On that page you will find a lot of instructions for building the bootloader, but about half-way down the page there are links to versions that have been precompiled for 20 and 24 MHz. Since the author does not highlight the links, they can be hard to find. Here is a direct link to the file:

<http://www.grozeaion.com/Electronics/optiboot.zip>

The author explains how to install them, but to simplify: Find the optiboot_atmega328__20MHz.hex file in the optiboot.zip file. Copy that .hex file into the bootloaders/optiboot folder (near where you found the boards.txt file). There will already be an optiboot_atmega328.hex file in the correct folder.

Now that we have the 20 MHz bootloader, we have to tell the IDE to use it for this board. The author of the bootloader has a number of suggested changes to the boards.txt file; however I have found that these two changes are enough.

In boards.txt change this:

```
uno20.upload.speed=115200
```

to this: (changes highlighted)

```
uno20.upload.speed=250000
```

and this:

```
uno20.bootloader.file=optiboot/optiboot_atmega328.hex
```

to this: (changes highlighted)

```
uno20.bootloader.file=optiboot/optiboot_atmega328_20MHz.hex
```

Make sure you change uno20 and NOT uno! Again, you will need to save the boards.txt file back to the original location and restart the IDE for the changes to take effect. Then you will need to reburn the bootloader with the Tools->Burn Bootloader menu item.

That's it. With that change and reburning (or burning it for the first time if you had not previously done so) the bootloader the Frequency Reference will be able to be programmed over serial.