Electronics Production: Building and Programming the FabISP

I. General Information

The FabISP is an in-system programmer that can be used to program other boards. Each Fab Lab comes with an off-the-shelf ISP, but learning how to build ISPs is an important lesson in electronics production and programming. And by making additional ISPs for the lab, students will have their own to use for future electronics projects.

More information about the FabISP can be found here:

http://academy.cba.mit.edu/

II. Production: Milling and Soldering

Mill the FabISP

Refer to the milling files here:

Traces: http://academy.cba.mit.edu/

Outline: http://academy.cba.mit.edu/

After milling, check the board carefully for any shorts or places where traces accidentally connect.

Stuff the FabISP

“Stuffing” a board means populating it with the proper electronic components and soldering them on. To do so, refer to the component diagram here:

http://academy.cba.mit.edu/

After soldering, check for shorts or “solder bridges”. Pay particular attention to the USB connection- the traces are very small and solder bridges can easily happen.

III. Computer Setup & Necessary Software

Ubuntu: Installing avrdude, GCC Software, and Dependencies

Log on to Ubuntu. Open Terminal and type:

sudo apt-get install flex byacc bison gcc libusb-dev avrdude

Then:

sudo apt-get install gcc-avr

Then:

sudo apt-get install avr-libc

Then (although this one may already be installed, you may also get some error messages):

sudo apt-get install libc6-dev

Firmware

Download the FabISP firmware from here (firmware.zip file):

http://academy.cba.mit.edu/classes/electronics_production/index.html

Save the file to a convenient location on your computer, for example: Documents> fab_isp.

Unzip the file (note that there are two folders- choose the one named "fabISP_mac.0.8.2_firmware" and rename it "firmware", then erase everything else in the "fab_isp" folder). Note that the firmware folder contains several files including a "Makefile" which should have no file extension.

IV. Programming

Connect Power

Connect FabISP to computer with USB cable

Connect Fab ISP to computer via AVR programmer

You should see a green light indicating that the board is getting power, if not, check that the programmer is connected properly, and check the board for shorts.

Navigate to the Firmware Folder

You will need to use terminal to program your board. When programming, it is necessary for

terminal to have access to the firmwared files. You will have to direct terminal to the location of these files.

In terminal, type:

cd ~/Documents/fab_isp/firmware

Check the Makefile

The Makefile is in the firmware folder. By default, it is set up to use the the AVRISP2 programmer. You will need to edit the appropriate line in the Makefile if using another programmer.

To check the makefile, type the following into terminal:

nano Makefile

If all looks correct, use "Ctrl x" to exit the makefile

Clean the Folder

You will want to delete all the extra files in the firmware folder. There are a few “compiled” files in this folder; they are provided for reference. However, we need to erase them in order to learn the compiling process by going through it ourselves.

Type:

make clean

You should see that three of the files were erased and you now only have "main.c", "Makefile", and "usbconfig.h" files left, along with the "usbdrv" folder.

Make the Hex File

You will now need to compile your files in order to create a hex file (this is the file that is actually burned to your microcontroller chip).

Type:

make hex

Set the Fuses

You will then need to set the fuses on your microcontroller chip. These are a few bytes of "permanent" storage on the chip (as in you can re-set them if needed, but you typically only

set these once). The fuses define basic parameters of the chip, including things like its operational speed and voltage.

Type:

```
sudo make fuse
```

If successful, you should get a message similar to this one:

```
avrdude -c avrisp2 -P usb -p attiny44  -U hfuse:w:0xDF:m -U lfuse:w:0xFF:m

avrdude: safemode: Verify error - unable to read hfuse properly. Programmer may not be reliable.
avrdude: safemode: Fuses OK (H:FF, E:DF, L:FF)

avrdude done.  Thank you.
```

Make the Program

Now you will actually need to upload the program to the chip, so that this board can be used as an ISP.

Type:

```
sudo make program
```

You should get a message similar to this:

```
avrdude -c avrisp2 -P usb -p attiny44  -U flash:w:main.hex:i

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################

avrdude: Device signature = 0x1e9207
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: writing flash (2002 bytes):

Writing | ################################

avrdude: 2002 bytes of flash written
avrdude: verifying flash memory against main.hex:
avrdude: load data flash data from input file main.hex:
```

avrdude: input file main.hex contains 2002 bytes
avrdude: reading on-chip flash data:

Reading | ################################

avrdude: verifying ...
avrdude: 2002 bytes of flash verified

avrdude: safemode: Fuses OK (H:FF, E:DF, L:FF)

avrdude done. Thank you.

avrdude -c avrisp2 -P usb -p attiny44 -U hfuse:w:0xDF:m -U lfuse:w:0xFF:m

V. Verifying the FabISP is Working

If you have successfully completed all the previous steps, it is a good idea to check to see if the ISP is actually working. Keep the terminal window open and type the following:

lsusb

This will provide a list of USB devices connected to your computer. The FabISP will look something like this:
Bus 001 Device 036: ID 1781:0c9f Multiple Vendors USBtiny

If you see this line, then congratulations, you have properly programmed your ISP!

VI. Using the FabISP as a Programmer

Now that the board is programmed, we want to be able to use it to program other boards. It is, after all, an ISP, or in-system programmer. When we built the board, we needed a few extra jumpers so that we could properly program it. Now that it is programmed, we can remove those jumpers.

Remove the two jumpers on the board- the 0 ohm resistor and the solder jumper.

Now the board is ready to be used as a programmer! Remember that any time you want to use it, you will have to make sure that the right programmer is indicated in the Makefile- the FabISP is referred to as "usbtiny".