**Setting up the Devboard\_PowerMonitor**

This is a work in progress.

Sections.

1)3.3 Volt switch

2)Battery Recharge and Recharge Enable Switch

3)Powering the Device.

3)Programming the CPLD/FLASH

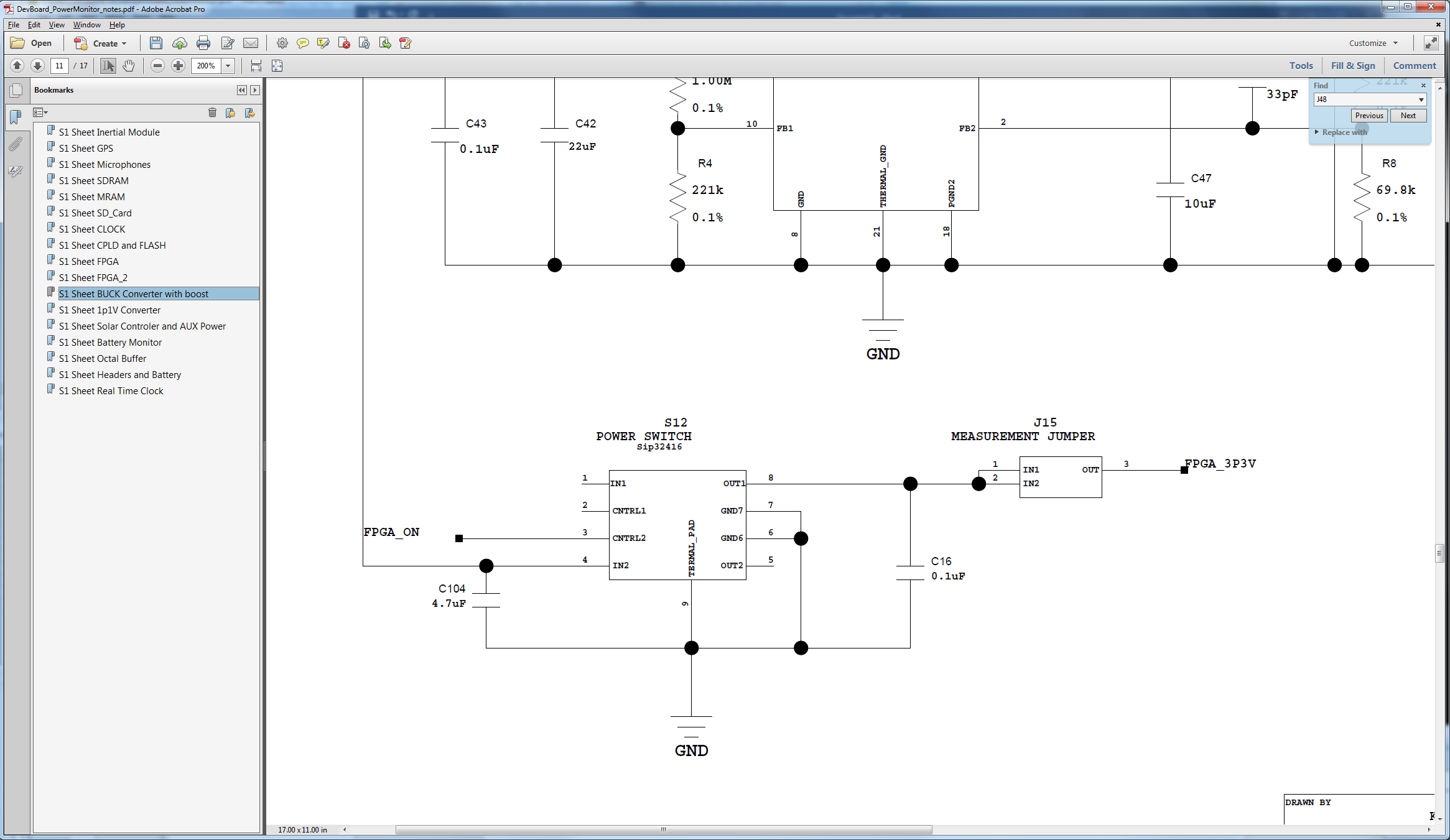
a) JTAG Jumping

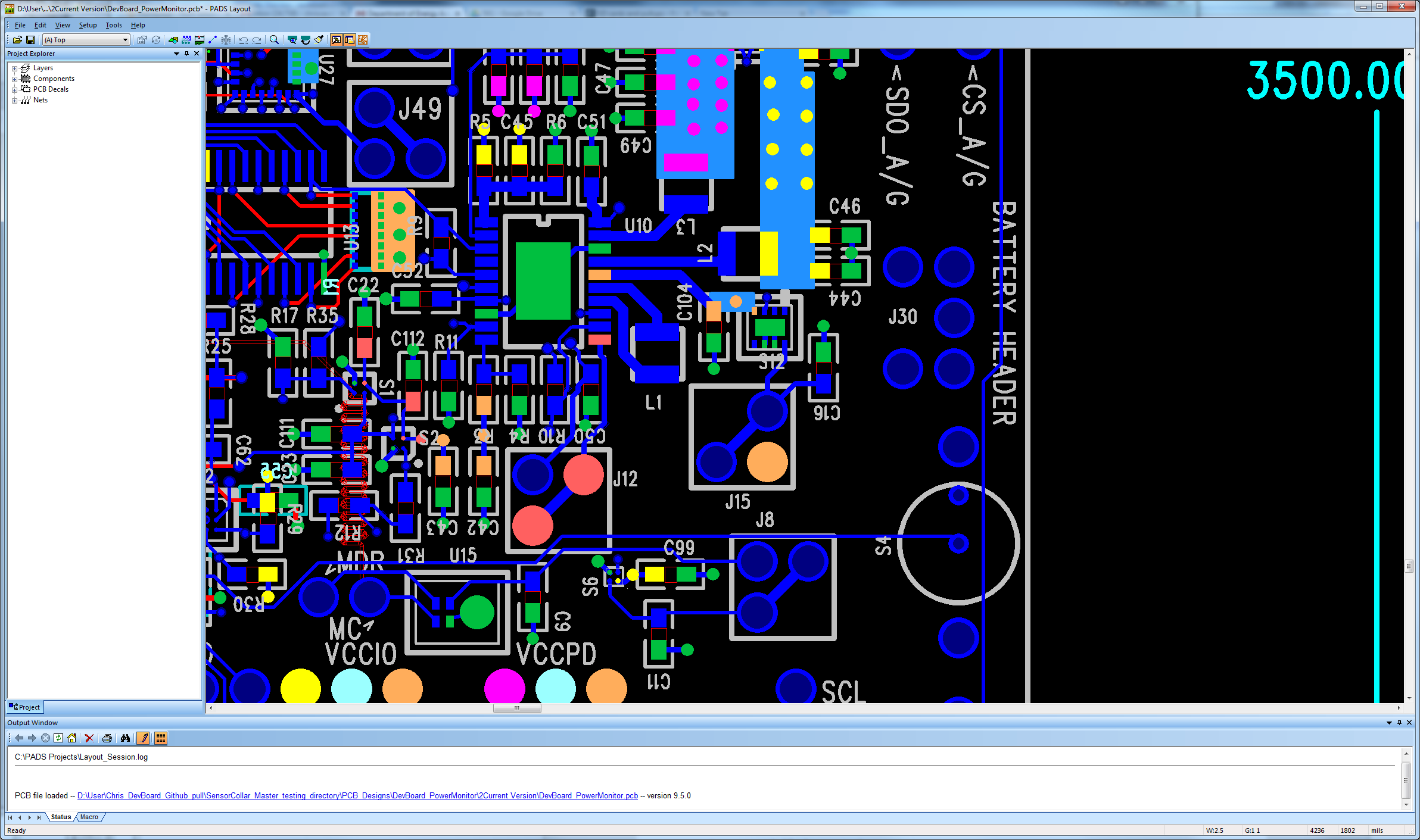
b) Programming correct images to CPLD and Flash

4)Botting and Programming the FPGA.

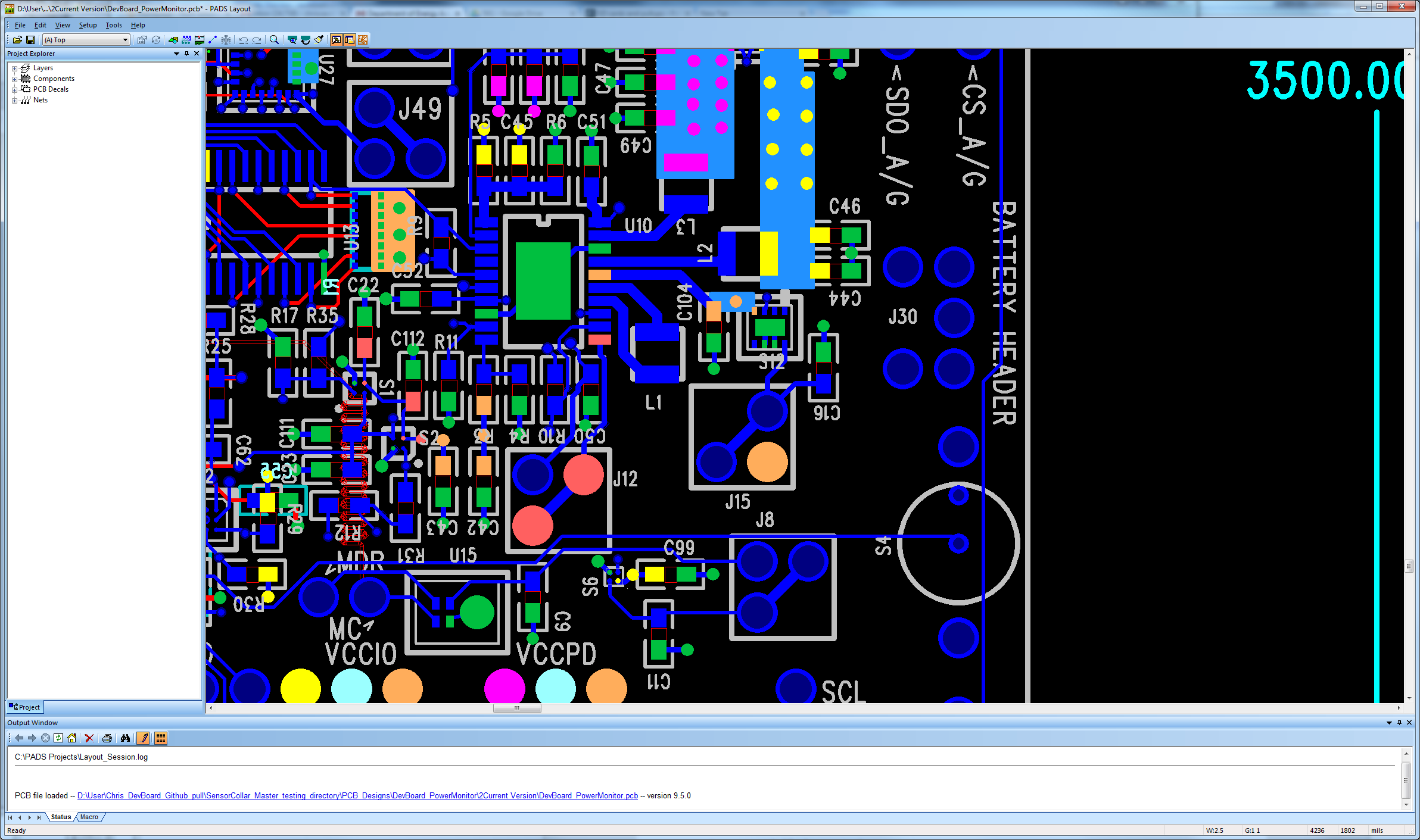
**1) 3.3 Volt Switch Fix**

The switch which controls the 3.3V rail which feeds the FPGA needs to be fixed. The switch is S12. The problem is that the control input and output do not match. Output 2 which is controlled is not tied anywhere.



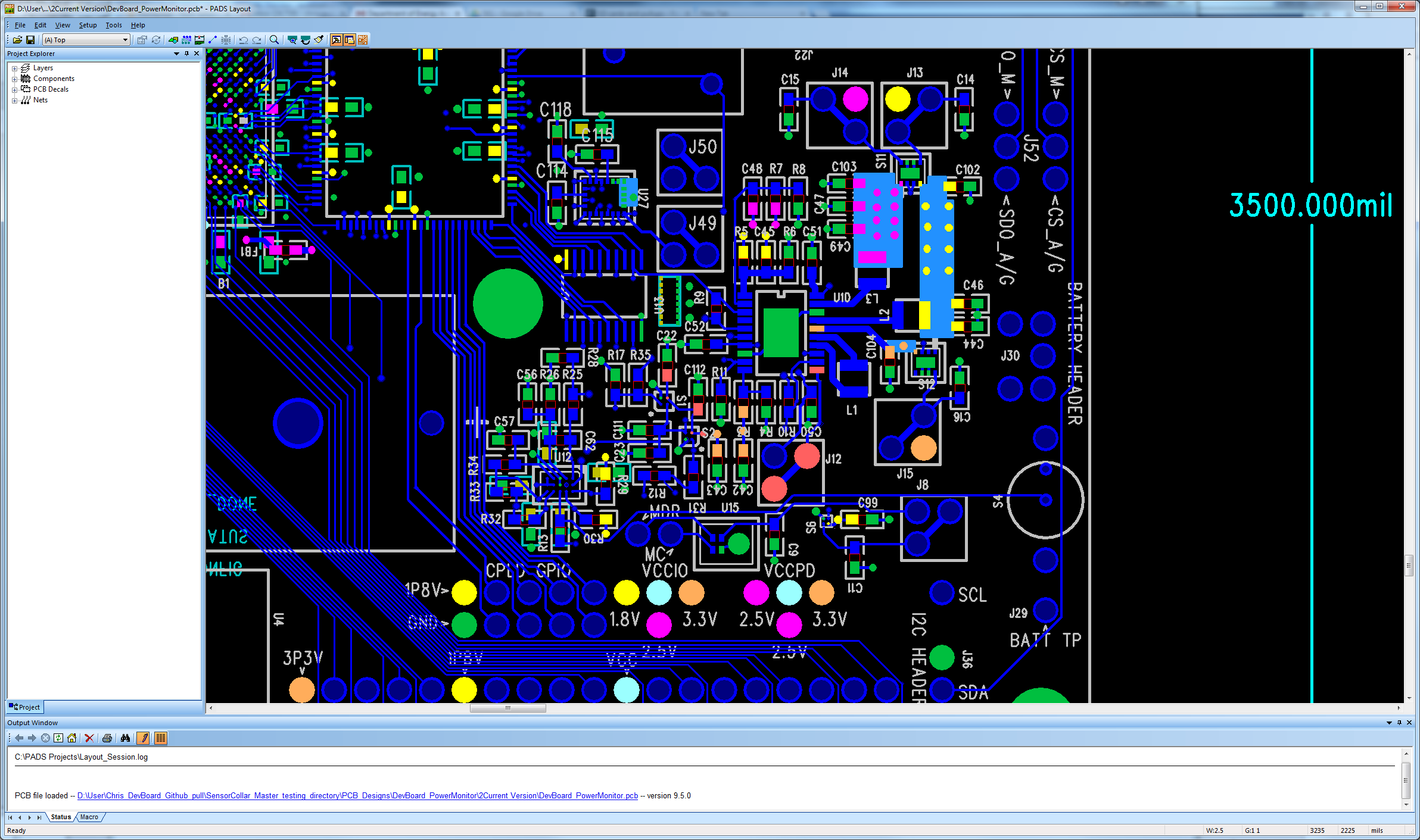


This is what needs to be physically done. Cut the trace at the dotted line. Insert a jumper wire as shown. I’m sure there are other solutions, this is what I did. You can also take out J15 completely and solder into those holes if that’s easier.



**2) S1 and S2 placed incorrectly. Pin 1 Placement Dot Incorrect. Decal Fine. Schematic Fine.**

Two switches on the board are placed incorrectly. If the switches are left in place, power is shorted to ground. You can either remove these switches or rotate them. If any battery testing is to be done on the board at a later time, these switches need to be rotated.



**Solution**

Either take the switches off with hot air or taken them off and rotate them 180 degrees. I’ve found you can do this without reapplying any paste or balls. They work after that.

**3) Powering the Device**

Jumpers needed for base operation:

J12 BuckBoost Input

J13 FPGA 1.8V

J14 FPGA 2.5V

J15 FPGA 3.3V

J21 Clock Power

J22 FPGA 1.1V

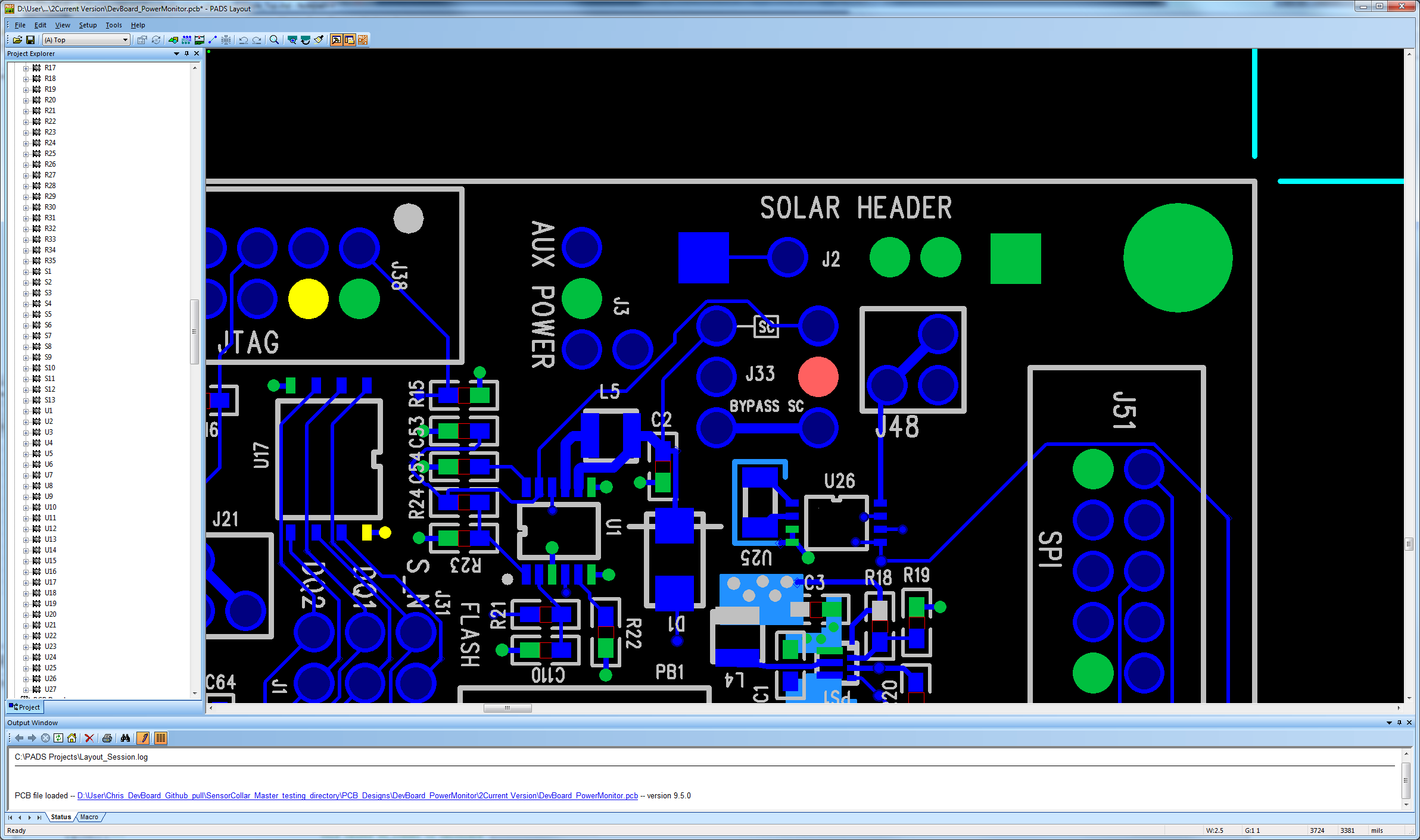
VCCIO and VCCPD Jumped 1.8V and 2.5V Respectively. Left Pin and Center Pin Jumped on both.

Bottom 2 sets of 2 at J33 Jumped. (Shown Below)

Any other devices you wish to work on need their associated power jumper.

Power should be applied as shown below at the solar header.

I strongly recommend current limiting your supply at first in case something goes wrong.

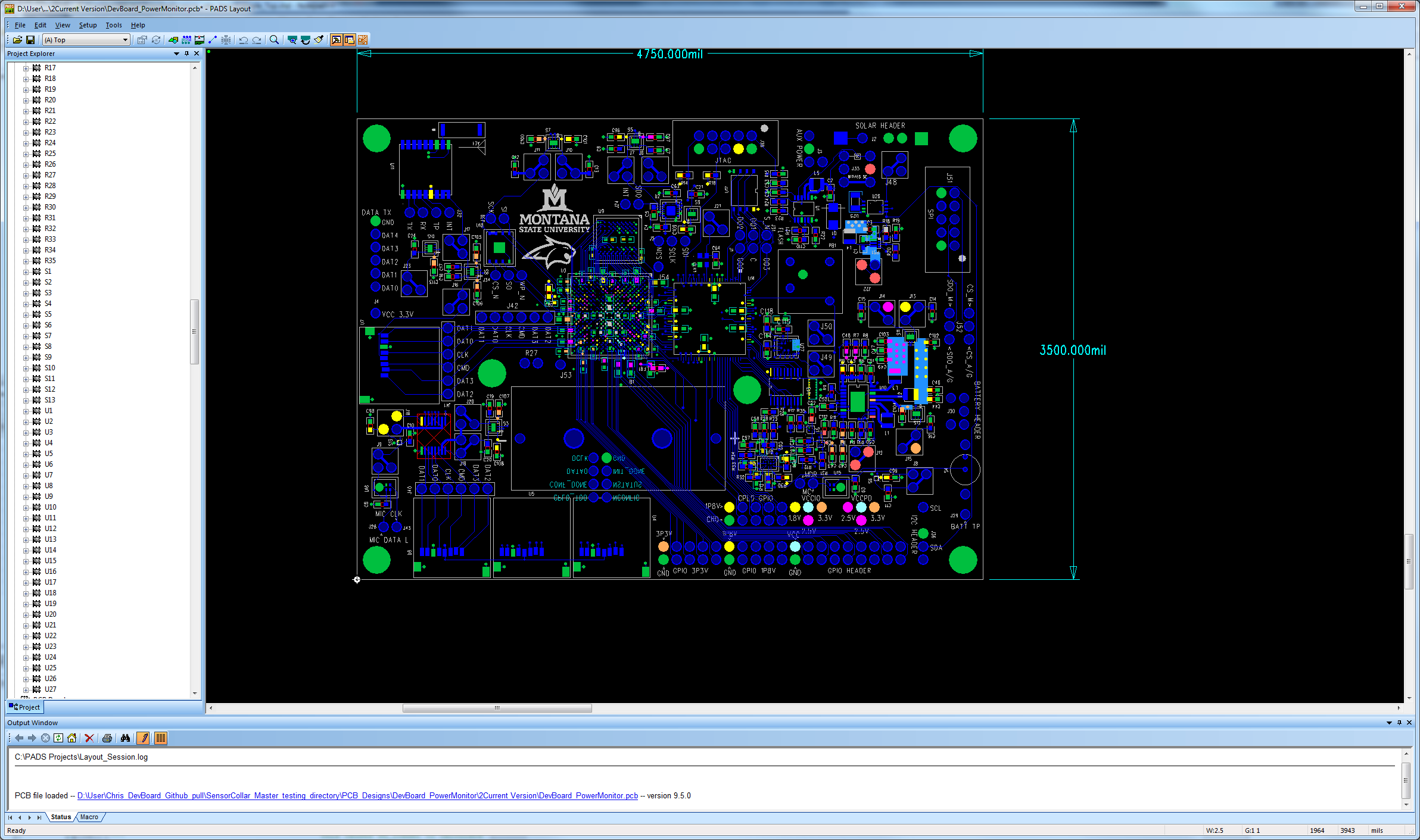


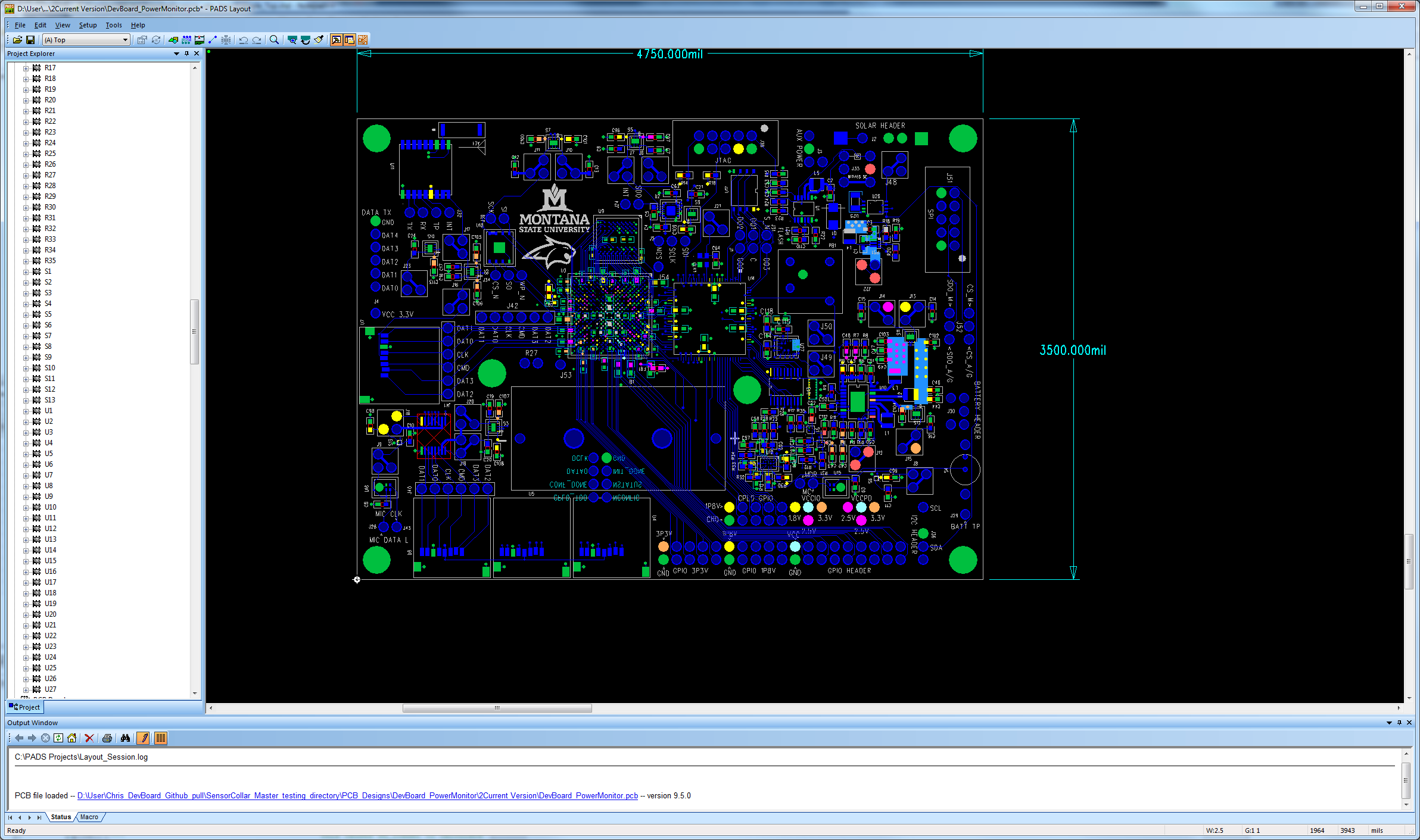
The buck boost needs >2.5V in. I recommend ~3.3V. However don’t exceed chip input max 5.5V. This configuration routes power onto power main and directly to the buck-boost regulator, bypassing the solar controller.

**3A)Programming the CPLD/FLASH**

Due to the FPGA now routing its TDI to TDO when it is shut off, the JTAG chain is broken by default. To fix this two things need to be done

1) Jump CPLD\_TDO to TDO of the JTAG header.





2) Get 1.8V Power onto the JTAG VCC line.

I recommend jumping J13 to the CPLD GPIO 1.8V (Located near the bottom of the board.)

You should now be able to see the CPLD on the JTAG chain in Quartus when a USB Blaster is connected to the JTAG header. The ribbon cable lays away from the power board on the USB Blaster. Now we will load a CPLD image which will allow us to see and load the FLASH with an FPGA image.

**3B) Programming the CPLD/FLASH**

Source\_Code\DevBoard\_PowerMonitor\CPLD\_INIT

Used for loading flash with FPGA image.

Before opening the Quartus project. Copy DevBoard\_PowerMonitor\_CPLDInit\_IO.qsf and rename to DevBoard\_PowerMonitor\_CPLDInit.qsf

A note on the QSF files. The project attempts to keep the QSF which Quartus maintains apart from the QSF which defines the board specific pins/ports/names as defined by the physical board.

For this reason, one should copy the Toplevel\_name\*\_IO.qsf to Toplevel\_name\*\_\*.qsf when starting from GIT for the first time.

Files need for this stage. These files need to be added to Quartus project.

../../MainCollar/General/Utilities\_pkg.vhd

../../MainCollar/PowerController/FlashWrite.vhd

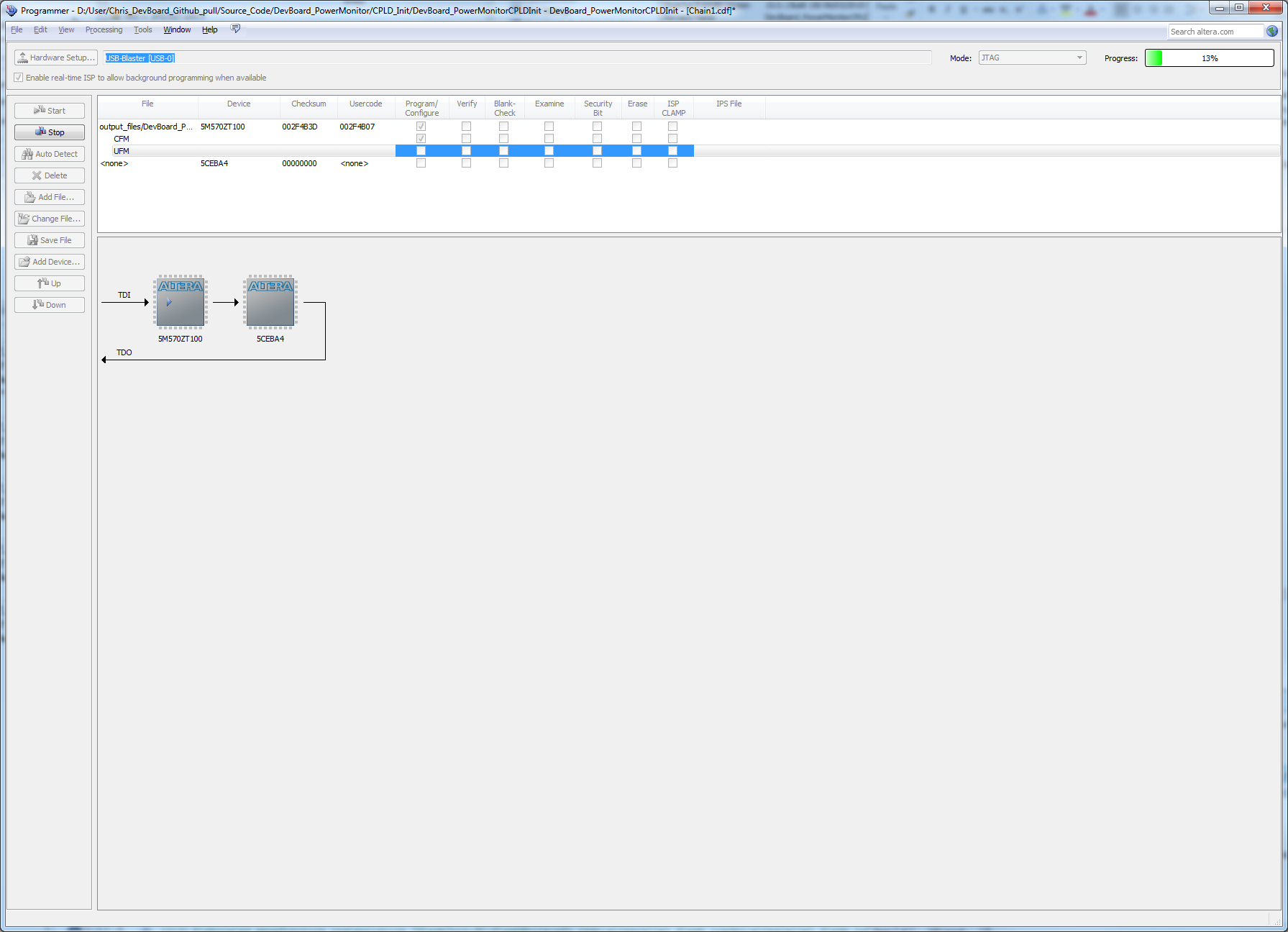
../../MainCollar/PowerController/FlashWrite.qip

../../MainCollar/PowerController/FlashInit.vhd

DevBoard\_PowerMonitorCPLDInit\_TopLevel.vhd

You should be able to compile now. A POF should now exist in the output\_files directory.

Open the programmer and program the POF to the CPLD. The CPLD is the first device in the JTAG chain. Its device identifier is 5M570ZT100. The image below shows the FPGA in the chain too. However, this won’t be there when you first initially program a new board. Notice the check boxes used to program. Enable real-time ISP is enabled. Program/Configure the CFM of the CPLD. We don’t need/want to program the UFM of the CPLD.

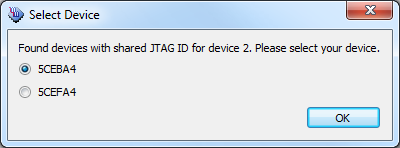


One done flashing now reboot the board. The new CPLD image only starts to run after a reboot of the device.

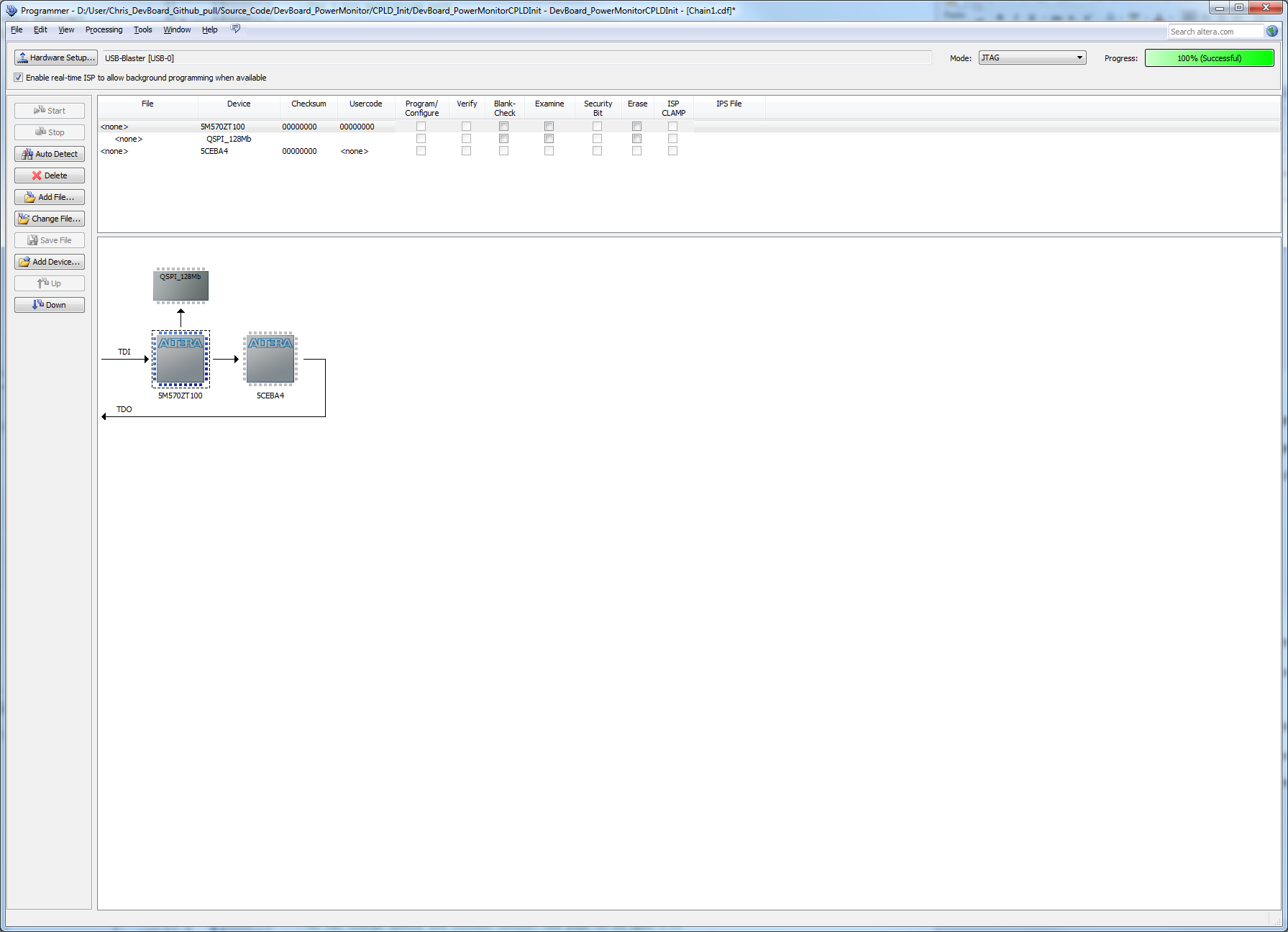
Remove the JTAG TDO jumper. It is no longer needed, as the FPGA turned on from now on. You can also remove the 1.8V jumped from the CPLD GPIO bank. The switch controlling the FPGA 1.8V rail is now turned on as well.

Now we can program the flash with an FPGA image.

The programmer needs us to select the ID for the FPGA.



If we take a look at the JTAG chain now:



Now the JTAG chain has the piece of flash and the FPGA on it. The flash is accessed through the parallel flash loader which is now loaded into the CPLD. The FPGA has now turned on.

Now a converted FPGA image must be programmed to the attached flash.

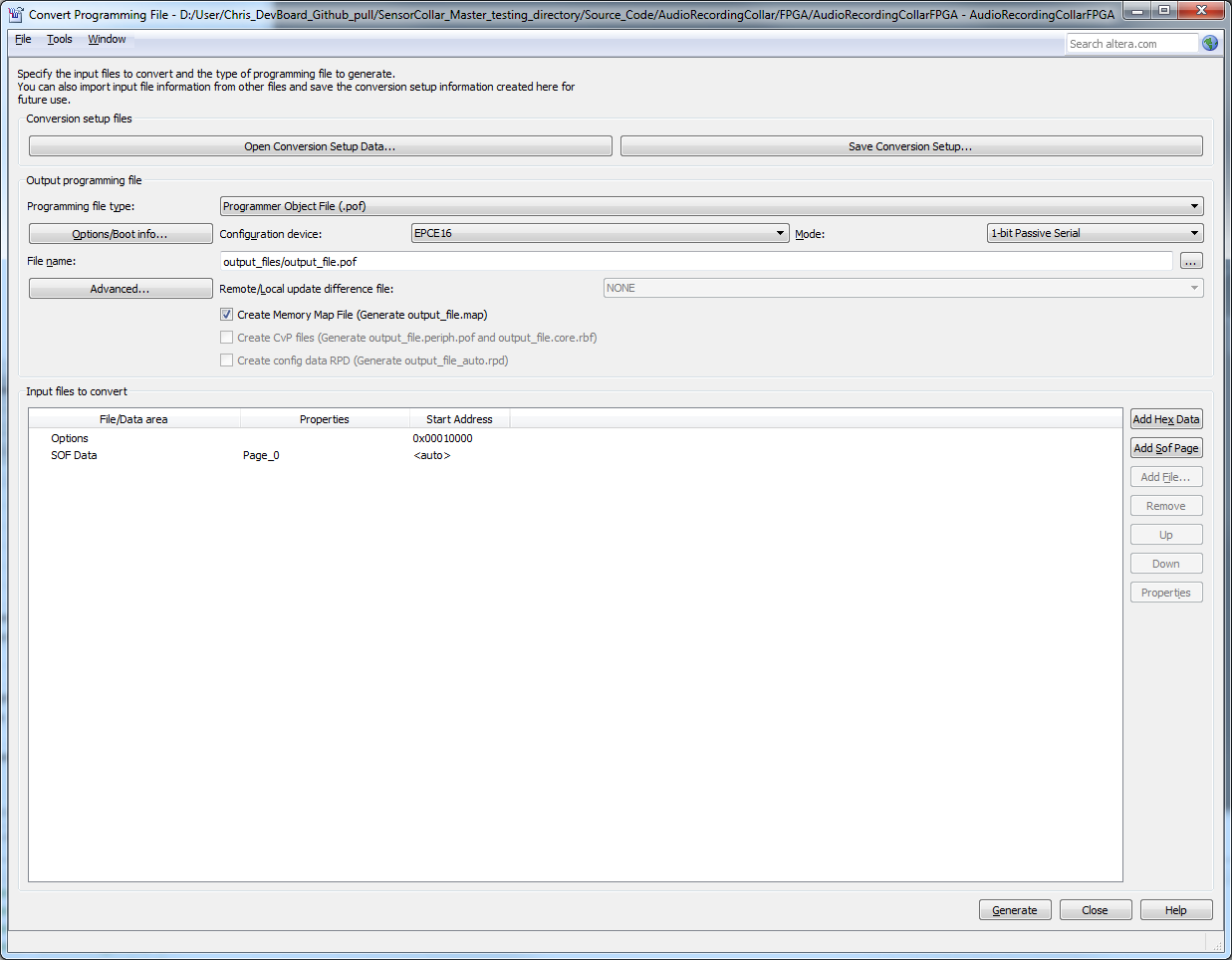
This will involve:

Source\_Code\DevBoard\_PowerMonitor\FPGA\_INIT

Bare bones FPGA project for loading to flash.

First a bare-bones bootable image of the FPGA must be compiled. A directory FPGA\_INIT exists for this purpose. It does not include any systems or the sdc timing framework. It simply displays a counter on a GPIO. This is a good way to check that your FPGA has booted and it booted with the image from flash.

Conversion of a SOF to POF is done through the Convert Programming Files section of the Quartus Application. It is located under the Files menu.



Source\_Code\DevBoard\_PowerMonitor\CPLD

Used for loading FPGA from Flash on Boot and Normal Operation

Source\_Code\DevBoard\_PowerMonitor\CPLD\_INIT

Used for loading Flash with FPGA image.

Source\_Code\DevBoard\_PowerMonitor\FPGA