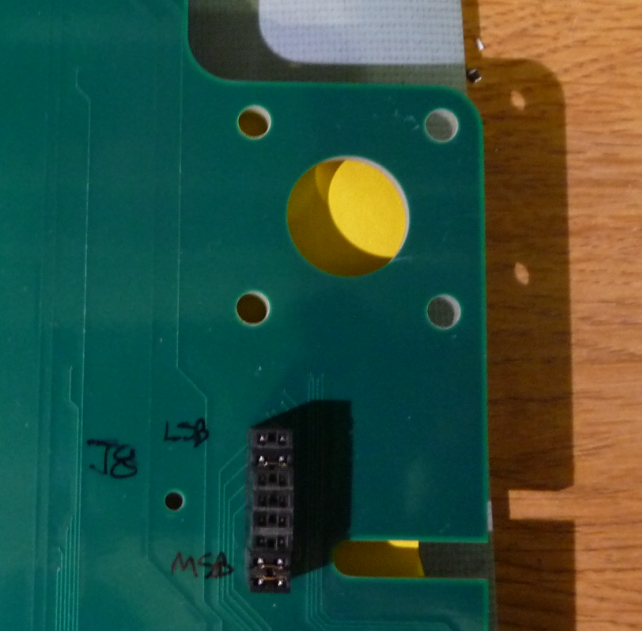
**Quabo Firmware Release 009.0**

**RR/Wei December 17, 2019**

**This firmware release includes the remote configuration ability which Wei developed, and a change in the Quabo IP address selection and the Quabo MAC address**

**IP Addressing**

Previous firmware versions allowed the selection of either IP address 192.168.1.10 or .11 by use of a single jumper on J3. IP address for this new version is set by the J8 8-bit jumper block plus two hard-wired bits on Mobo, or the J8 10-bit jumper block on Mini-Mobo. An installed jumper is a ‘0’; a removed one is a ‘1’. The silkscreen seems to have been erased from the bottom of the batch of Mobos that we had machined; here’s J8:



The LSBs are encoded as follows:

Quabo Position LSBs

J1 00

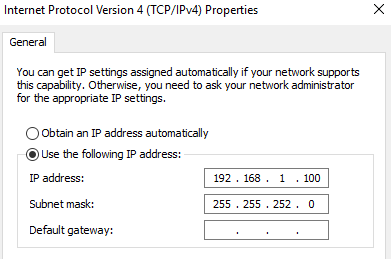
J2 10

J3 01

J4 11

(Note that these are not in the order you might expect)

In order to use 10 bits for IP addressing, we need to open up the network mask (to permit access to more than 256 different addresses). In Windows:

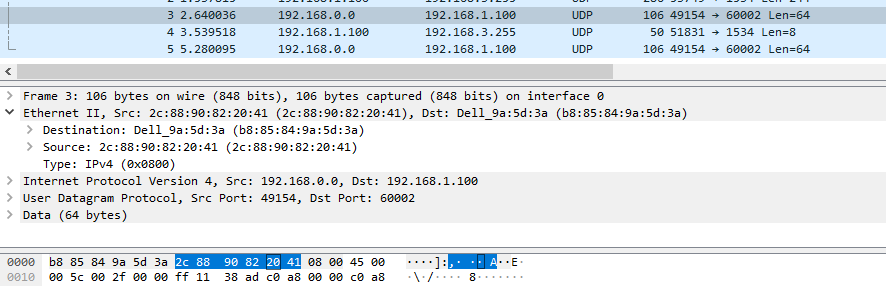


That 255.255.252.0 is ff.ff.fc.0, so unmasking 10 bits of addressing. We’ve left the host IP address at 1.100 in this version, though we’ll change it to maybe 192.168.3.254 in a future version, since 1.100 will likely be used for a Quabo in a full observatory.

If you install all 8 of the J8 jumpers and plug into J1 of a Mobo, or install all 10 of the J8 jumpers on mini-Mobo, you’ll be setting the IP address to 192.168.0.0, as I’ve done in the following.

**MAC Addressing**

V9.0 reads the unique ID from a chip on Quabo and uses this as Quabo’s MAC address:



Note that this is not the same as or derived from the WR MAC address; the UID chip used to get the general MAC address is read from the flash configuration RAM, while that used for the WR MAC is read from a separate UID chip. This is transparent to the user, but should be noted for debugging purposes.

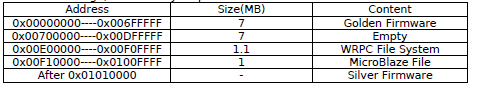
Set the IP address as described above before proceeding with the configuration of the new firmware, since this takes place over the network. Any IP address can be used, but I’ve set it to 0.0 for this example.

**Setting up the remote-configuration firmware for Quabo**

Wei has finished incorporating remote configuration capability into the Quabo firmware. Here’s how it works, and how to set it up.

The 32MB flash chip on Quabo is now divided into several regions, including an area at address 0 for the “GOLD” copy of firmware (this is the copy that Quabo will always boot to at power on) and an area at 0x0101\_0000 for the “SILVER” copy, which is where we will put the current version of firmware (the idea being that we’re certain that the GOLD copy works, so that if something goes wrong with SILVER, say we’ve made a mistake and the TFTP communication fails, we can cycle power, boot to GOLD, and uploaded a corrected SILVER copy). After booting to GOLD on power-up, Quabo must be commanded to boot to SILVER using TFTP. This will be done after every power cycle.

Here is the overall flash memory map:

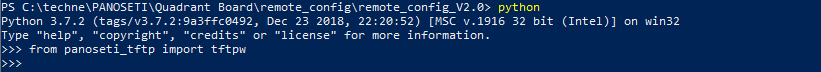


The remote\_config folder contains the GOLD copy, quabo\_v0090\_GOLD in both .bit (for directly loading to the FPGA via a JTAG cable) and .bin (to load to flash via TFTP) formats, as well as the panoseti\_tftp.py Python module which Wei wrote to simplify the user interaction, and a more extensive user manual for remote configuration. The GOLD copy is the same as the SILVER one at this point, except that its ability to output housekeeping packets has been removed. This is so that we can tell definitively that we’ve booted to the right version. GOLD will remain unchanged while SILVER will be updated as we make firmware revisions.

The communication with quabo is done using TFTP from within Python. So you first need to install the tftpy package:



After having done that, navigate to the remote\_config folder. Start Python (3.xx) and import the needed module:



**The following actions will need to be taken once only for each Quabo, and will in future be done “at the factory” (ie, by Wei or Rick or the assembly house):**

1) Connect a JTAG cable to quabo and program the FPGA with quabo\_v0090\_GOLD.bit. This gives us the ability to communicate with Quabo via TFTP.

2) Load the flash with the GOLD copy by entering the following commands, establishing the connection with the proper IP address, as set by the Mobo jumpers described above:



The upload takes about 1 minute. Note that the put\_bin\_file command explicitly specifies address 0x0; the default if no address is supplied is the SILVER address of 0x0101\_0000. In the field, the user will always use the default.

Now the GOLD copy is stored in flash, and so this quabo is set up for remote configuration. If you cycle power at this point, no housekeeping packets will be produced.

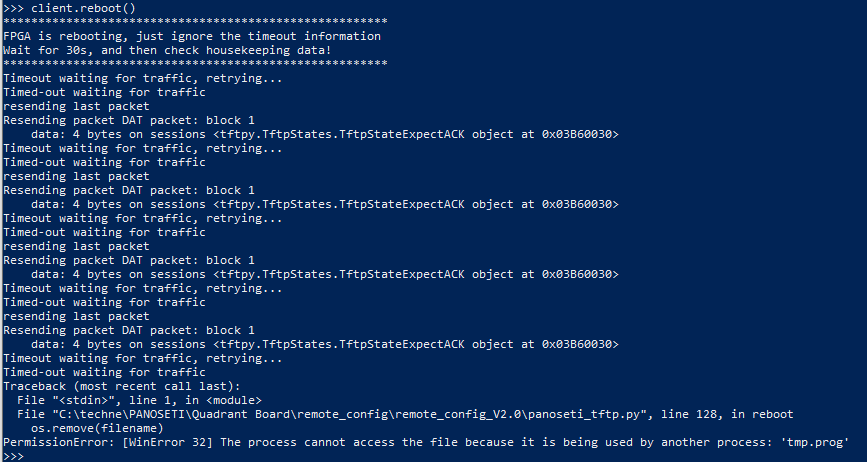
**The following will be done each time a new firmware release is to be loaded up.**

In this step the SILVER (current) firmware version is stored in the SILVER address.



This upload will also take about 1 minute.

Having loaded up the new firmware, the quabo must now be told to reboot to that new firmware:



The error messages here are expected: Quabo is in the process of rebooting and cannot respond to TFTP transactions.

**The client.reboot() command must be sent after every power cycle. Reboot takes about 15 seconds.**

Having rebooted, the housekeeping packets should now be occurring every second.