RADIANTv2 Initial Setup

**v0.1 - PSA - 3/2/21**

# Files To Download

* <https://github.com/RNO-G/radiant-circuitpython/releases/download/v0r1/radpy_v0r1.uf2>
* <https://github.com/RNO-G/radiant-circuitpython/archive/v0r1.zip>
* <https://github.com/RNO-G/radiantBoardManager/releases/download/v0r2p2/radiantBoardManager_v0r2p2.uf2>
* <https://github.com/RNO-G/firmware-radiant/releases/download/v0r2p3/radiant_top_v0r2p3.bin>

# Programs Needed

* Serial terminal program (minicom, TeraTerm, HyperTerminal, etc.)

The RADIANT board comes programmed with a UF2-compatible bootloader in the board manager, allowing relatively quick setup. When powered on and connected to USB, a disk should appear with the RADIANTBOOT label.

# Initial Steps

1. Turn on RADIANT power (5V, 2.5A)
2. Plug in USB to *RADIANT* USB (located on underside of board, near power connector)
3. Device should enumerate and “RADIANTBOOT” drive label will appear
4. Copy radpy\_v0r1.uf2 to RADIANTBOOT drive
5. Device will disconnect, and re-enumerate with a CIRCUITPY drive label + serial (COM) port
6. Extract/Copy all files from “v0r1.zip” (from radiant-circuitpython) to the CIRCUITPY drive.
7. Connect terminal program to serial port, baud rate doesn’t matter (it’s USB).
8. Hit any key, then Ctrl-D to reload and hit any key again. At this point you’re ready.

# Note on General Usage

# CircuitPython’s memory usage is tight, so reloading/starting over is often needed. To do that just hit Ctrl-D from the terminal.

# Basic checkup

Once the CircuitPython image and CIRCUITPY code is loaded, the board should immediately turn on the FPGA power supplies (but not LAB4/trigger). This can be checked manually, or, from the serial terminal, do:

>>> from radiant.radpower import RadPower  
>>> VCC = RadPower()  
>>> VCC.status()  
1.0 1.00386 ON GOOD  
1.8 1.79099 ON GOOD  
2.5 2.49514 ON GOOD  
A2.5 OFF BAD  
A3.0 OFF BAD

(Note that the A2.5V and A3.0V rails do not have direct analog checks, so they are not shown here)

# Turn on the main LAB supplies

>>> VCC.lab(True)  
True  
>>> VCC.trig(True)  
True

Repeating VCC.status() will now show A2.5/A3.0 as ON and GOOD. The “True” response tells you that the supplies turned on OK (the power good ‘PG’ pin went high).

# Programming the DAC I2C addresses

First step in the initial setup is to assign the DAC I2C addresses. There are 6 MCP4728 DACs on the board – initially, they all have the same I2C address. We need to assign them individual ones, which is done via a CircuitPython script. Open terminal, and do

>>> import assigndac  
>>> assigndac.setup()  
>>> assigndac.assign()

**NOTE**: if there are any errors after “assigndac.setup()”, stop and record those errors. Do *not* run assigndac.assign(). This may indicate a problem with some of the devices on the board. assigndac.setup() should also light all the GREEN LEDs grouped every 4 channels.

# Programming the FPGA SPI Flash (via Vivado)

While the FPGA *can* be programmed via the board manager itself, it’s much quicker if there is a Vivado installation + Xilinx programmer available. Later an “FPGA bootload” image will be available which will allow setup even without Vivado.

Hook up the Xilinx programmer and launch Vivado.

Go to Flow->Hardware Manager. “Open Target” -> “Auto Connect”. The Xilinx device + FPGA should show up.

Right click on xc7a200t\_0. Select “Add Configuration Memory Device…” Go to “Select Configuration Memory Part.” Type in “s25fl256s”. Select the one with the 0 on the end “s25fl256sxxxxx0-spi…”.

Right click on the new SPI device (s25fl256…). Select Program Configuration Memory Device. Click the … for the Configuration file and select “radiant\_top\_v0r2p2.bin” (from the setup package).

Click OK. This will take a while.

Right click on xc7a200t\_0. Click “Boot from Configuration Device”. The LED near the FPGA should begin rapidly blinking. **You should now close the Hardware Manager.** Important Note: if the Hardware Manager is open, **and** you power cycle the RADIANT, the FPGA *will not program*. This is apparently “a feature.” You can avoid this by ensuring you close the Hardware Manager.

Repeating for importance: *close the hardware manager.* If you are ever in doubt if the FPGA’s programmed, watch for the blinking LED.

# Initial Setup Done

At this point, the “once and done” procedures (for now!) are complete. Note that in the future three additional “once and done” procedures may be added – first, to permanently program the clock chip (which is done dynamically now at first power-on), second, to permanently program the CPLDs (which is again done dynamically now), and third, to give all of the onboard DACs a “reasonable” default value. None of these procedures are ‘necessary’ but may be done to save time in the future.

**Note**: without any programming, the LAB4D “pedestal” DAC is initialized to “mid-scale” (1.67V) which is very high for a pedestal, but still a reasonable value for an initial checkout.

# What We Know At This Point

* Board manager is OK
* Board manager SPI is OK
* Clock is OK
* GPIO extenders and DACs are OK
* 1.0/1.8/2.5V supplies are OK
* FPGA is OK
* FPGA SPI flash is OK

While the A2.5/A3.0 supplies *seem* OK at this point they’re not guaranteed: we just know that their PG pins went high. Readback occurs indirectly through the LAB monitoring.

# RADIANT Board Checkout

Once the board is set up, we can now proceed to testing out the remaining base functionality. To do this we need to program the CPLDs. Start off by reloading (Ctrl-D).

>>> from radiant.radfpga import RadFPGA  
>>> from radiant.radcpld import RadCPLD  
>>> import board  
>>> dev = RadFPGA(board.UART())  
>>> dev.write(0x14, 0x80000000)  
>>> cpl = RadCPLD(dev, 0x1C)  
>>> cpl.configure(“radiant\_aux\_impl1.bit”)  
>>> cpr = RadCPLD(dev, 0x20)  
>>> cpr.configure(“radiant\_aux\_impl1.bit”)

Next, check out that the CPLDs configured properly.  
>>> dev.write(0x14, 0)  
>>> dev.write(0x8, 0x1FF01FF)

This should light up all 4 LEDs attached on both sides to either CPLD.

# Check out the individual LAB supplies

Reload (Ctrl-D). (If you get a memory error at any point here, reload and try again).  
>>> from radiant.radfpga import RadFPGA  
>>> import board  
>>> from radiant.radgpio import RadGPIO  
>>> from radiant.radbist import RadBIST  
>>> dev = RadFPGA(board.UART())  
>>> gp = RadGPIO(board.I2C(), board.SCK, board.MOSI)  
>>> bist = RadBIST(gp, dev)

You can now cycle through each of the LABs and read their monitoring. Note that you should turn each group of LABs on first:

>>> gp.lab(0, True)   
>>> gp.trig(0, True)

This activates channels 0-3.

>>> bist.labstat(0)  
PED: 1681.23  
RAMP: 134.402  
AMON: 104.624  
3.0: 2968.1  
2.5: 2506.15

(These values are in millivolts). **Note:** while the pedestal voltage (PED) is relatively high here (1.67V), it’s still fine for the LAB4D and this is the default value for the DACs. It’s helpful to just leave it here to ensure that all of the LAB4D quads have functional pedestal circuits.

Go through all 24 channels (turning on each quad from 0 to 5 via gp.lab()/gp.trig()).

# Check out LAB4 Programming

Next do

>>> dev.writefile(“l4danadef.txt”)  
>>> bist.labstat(0)  
PED: 1681.23  
RAMP: 134.402  
AMON: 696.957  
3.0: 2968.1  
2.5: 2506.15

Note that AMON has increased to 700 mV after programming the LAB4D, indicating that it is alive.

# What We Know At This Point

* Board manager is OK
* Board manager SPI is OK
* Clock is OK
* GPIO extenders and DACs are OK
* 1.0/1.8/2.5V supplies are OK
* FPGA is OK
* FPGA SPI flash is OK
* CPLDs are OK
* Per-LAB pedestal circuits are OK
* A2.6V/A3.0V supply is OK
* Per-LAB supplies are OK
* LAB programming is OK and analog functionality is OK

At this point basic board checkout is complete: remaining tests should be done communicating with the CB.

# Load Production Board Manager Firmware

1. Put the Board Manager into bootloader mode by pushing the Reset button twice relatively quickly. (S16, near the array of board manager test points) You will know it is in Bootloader mode if the LED shows a “breathing” effect (dimming instead of blinking)
2. The board should re-enumerate again with RADIANTBOOT.
3. Copy radiantBoardManager\_v0r2p2.uf2 to RADIANTBOOT.
4. The board should re-enumerate again, but only with a serial port. (You may need to power cycle the RADIANT).

At this point the board is ready to use by the carrier board for calibration.