Building a Working ACDC/ACC Test Setup (REV B)

Digitizing a Sine Wave

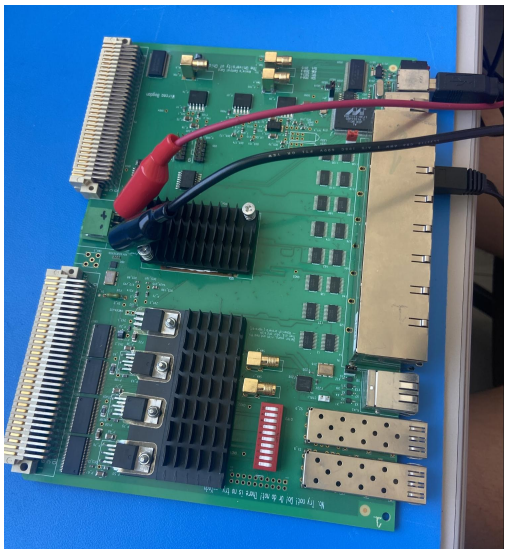
Marek Michulka
University of Chicago

Tobias Abelmann
University of Chicago

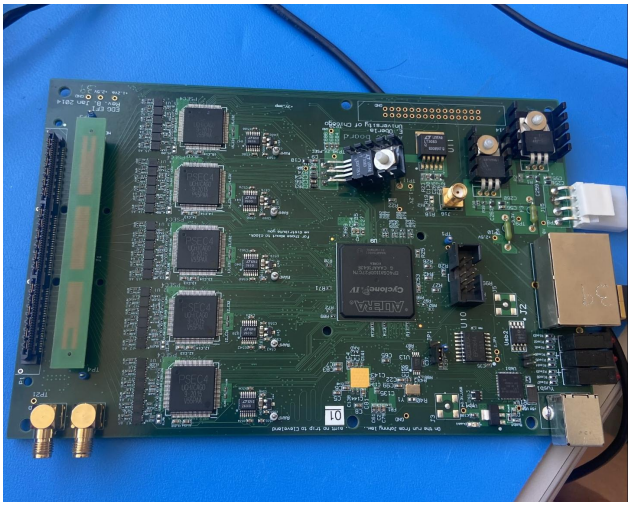
16th September 2021

1 Test Setup Basics

1. Go to the University of Chicago electronic shop in the accelerator building (or wherever the necessary components – listed below – for a test setup might be). Locate a couple ACDC Rev B boards (one will work, but having multiple in case there is anything finicky is a good idea[1]), an ACC board, and two working power supplies capable of outputting 3 amps at 5 volts each. See Figs. 1a and 1b for examples.



(a) ANNIE Central Card (ACC)



(b) ACDC Rev B

Figure 1: ACC and ACDC

2. Look for a Raspberry Pi or some other functional linux machine.[2]

[1] Boards 36 and 39 were working well at the time this was written.
[2] Ask Mary Heintz for help with this if necessary.

3. Find the necessary cable.

   (a) A gray power cable, which can be seen in Fig. 2 for the ACDC.

   (b) A set of two alligator clip cables for the ACC (unless there is a power cable that connects to the board – note that the ACC has a different pin structure than the ACDC for the power cable).

   (c) Ethernet cables to connect from an ACDC to an ACC.

   (d) And a USB 3.0 to USB Type B cable to connect your Linux Machine (the Rasberry Pi) to the ACC.

4. Connect the gray ACDC power cable to one power supply (putting black to ground) and to an ACDC.

5. Connect the alligator clip cables to the ACC as pictured, following the +/- markings on the card, to another power supply (if using a different ACC than the one pictured, reference "Testing of the ANNIE Central Card for a 5-15GSa/s PSEC4 Readout System" by John Podczerwinski and Horatio Li, 316 in the LAPPD library for the +/- pin layout).

6. Turn on the power supplies and verify that in the first ∼30 seconds the current draw rises by about half an amp; this occurs when the boards' respective FPGAs turn on. If this does not happen, check that the boards are connected properly, try power cycling, and if that fails, try another board, try another power supply, and lastly ask for help from Mary Heintz, Paul Rubenstein, Eric Oberla, or Evan Angelico.

7. For the first time working with a particular board, smell for smoke when switching on the power supply.

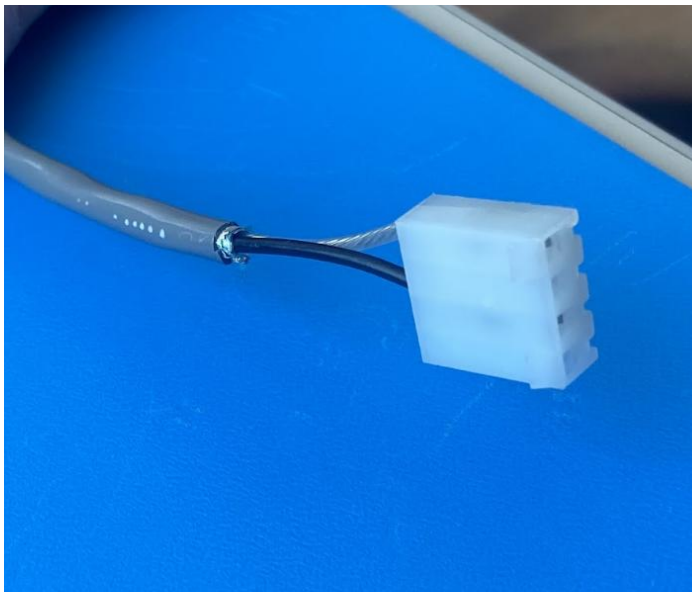


Figure 2: ACDC Power Cable

2 Cloning the Git Repository to Your Linux Machine

1. Connect to your Linux machine.
2. Install Git (if it isn't already installed) using the following command:

   sudo apt-get install git

3. Clone the acdc-daq-revc repository (within https://github.com/lappd-daq):

   git clone https://github.com/lappd-daq/acdc-daq-revc.git

4. After cloning the repository, enter it:

   cd acdc-daq-revc

5. Switch to the UChicago-Test branch:

   git checkout uchicago-test

6. And now try pulling from the branch:

   git pull

3 Installing ACC Firmware

1. Power on the ACC.
2. Find a computer with Quartus Prime installed or install Quartus Prime on the desired computer (we used Quartus Prime 20.1)[3].
3. On that computer, download the firmware from this link by clicking the green "code" button and then "download zip." This will download all the firmware versions for both the ACC and the ACDC.
4. In Quartus, click "open project" and navigate to and select the .qpf file for the version of the firmware that you are using (we used ACC version 3.01). On the left side of the screen (or the top) find the "compile design" button (it looks like a play button) and compile the firmware. This should take a few minutes.
5. In the meantime, locate a USB blaster (if one isn't nearby, ask Mary Heintz for help) and connect the USB blaster to the computer running Quartus via USB. An example can be found in Fig. 3.
6. Next convert the .sof file generated into a *.jic* file.
   (a) Go to "file" and "convert programming files."

[3]Most computers in the electronics shop would work for this, but ideally find one close to your test setup.

(b) Select the dropdown menu under "programming file type" and change from *.pof* to *.jic*.

(c) Click the three dots (...) next to "configuration device." This should open a new menu.

(d) At the top of the screen, change the device family from "Arria 10" to "Arria V."

(e) In the configuration device select "EPCQ256" and click OK.

(f) Name the output file (sisyphus.jic for example).

(g) Click "Flash Loader" at the bottom of the page and click "Add Device."

(h) Click "Arria V" for device family and "5AGXFB5H4" for device name.

(i) Below "Flash Loader", click on "SOF Data" and click "Add File." Then navigate to the .sof file that was compiled.

(j) Click "Generate." This can take a few seconds.

7. After converting the .sof file into a *.jic* file, program the FPGA of the ACC. This is called "flashing" the firmware. To flash the ACC firmware:

   (a) Click "Program Device" in the menu under "Compile Design." This should open a new window.

   (b) Make sure the list is empty (if there are any factory added devices, remove them).

   (c) Click "Add File."

   (d) Select the *.jic* output file generated in the last step. Make sure mode is "JTAG."

   (e) Click "Hardware Setup," double click "USB-Blaster," and close.

   (f) Plug the USB blaster into the port of the ACC boxed in yellow in Fig. 4 while wearing a static strap.

   (g) Check the box under "Program/Configure" and click start.

8. After programming the ACC, power cycle. If the programming worked properly, the ACC should appear when running

   ./bin/connectedBoards

   from your linux console (provided that it is connected to the ACC via USB).

Figure 3: USB Blaster

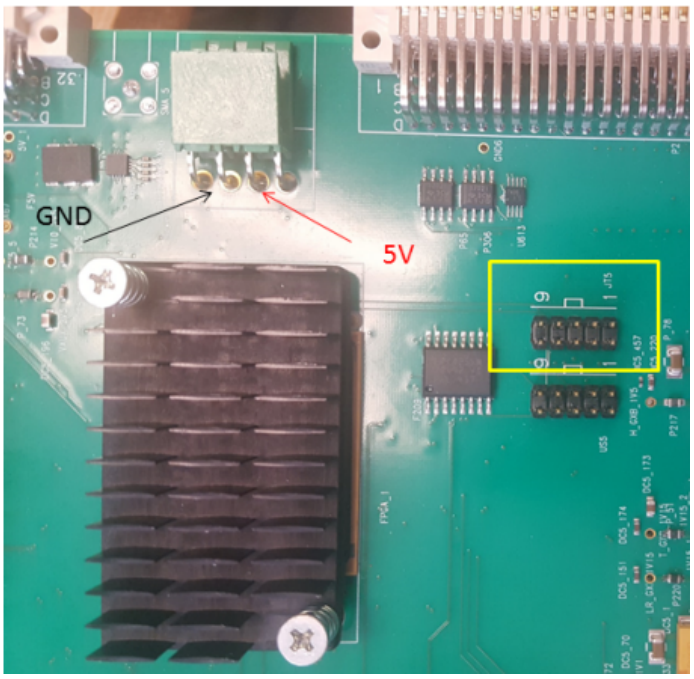


Figure 4: JTAG port used for flashing firmware onto the ACC. If flashing a .sof or .pof file, refer to document 316 in the LAPPD library.

4 Installing ACDC Firmware

1. If you have already downloaded ACC firmware, you should have ACDC firmware as well, if not, follow the above instructions for how to download the firmware from Github.

2. In Quartus, find the correct version number for the ACDC firmware (we used v3.04) and inside that folder, open the *.qpf* file. As done for the ACC, compile the ACDC firmware to get a *.sof* file.

3. Convert the *.sof* file to a *.jic* file.

   (a) Go to "file" and "convert programming files".

   (b) Select the dropdown menu under "programming file type" and change from *.pof* to *.jic*.

   (c) Click the three dots (...) next to "configuration device". This should open a new menu.

   (d) At the top of the screen, change the device family from "Arria 10" to "Cyclone IV GX".

   (e) Look at the smaller Altera chip on the ACDC. This can read either "EPCQ64A", "EPCS64", or "EPCQ64". An example can be seen in Fig. 5. Select the one that your chip reads under "Configuration Device" and click *OK*.

   (f) Name the output file (e.g., tantalus.jic)

   (g) Click "Flash Loader" at the bottom of the page and click "Add Device".

   (h) Click "Cyclone IV GX" under device family and "EP4CGX110" under device name.

(i) Below “Flash Loader”, click on “SOF Data” and click “Add File”. Then navigate to the *.sof* file that was compiled.

(j) Click “Generate”. This can take a few seconds.

4. Flash the *.jic* file onto the ACDC (see the instructions for flashing the ACC – the steps are identical).

5. When this is completed, power cycle the ACDC and run the following command:

   ./bin/connectedBoards

   to check that the ACDC is properly connected.

Figure 5: Altera Chip on ACDC

5 Checking Connection and Calibrating Pedestals

Once firmware is installed on both the ACC and ACDC, begin running commands on the linux machine to test if the boards are fully functional.

1. Turn on the connected power supplies.

2. Run the command:

   ./bin/connectedBoards

   to see if the boards are communicating, with both boards on the noted firmware versions and the proper software on the Linux machine, this should, theoretically, work without issue.

3. Run the command:

   ./bin/calibratePed

   This will record 1000 events through the ACDC boards. Some error messages may appear, but if the program successfully runs through the 1000 events and calls "acc destructor" at the end, it should be working as expected. This function takes about 5-10 minutes to run (including the Python analysis of the data at the end).

4. If *calibratePed* works, then in theory the test setup is working with full functionality.

6 Digitizing a Sine Wave[4]

1. It is necessary to have some code that can plot the ACDC outputs, taking into account the calibrations. Our code is in Plotting/environment in the uchicago-test branch of the LAPPD Github.

2. Locate an oscilloscope (e.g., DPO 7354) and an arbitrary waveform generator capable of generating frequencies of 40 Mhz or more (e.g., AWG 5012) and move them to your workspace. See Fig. 6 for an example.

Figure 6: The Oscilloscope (Top) and the Arbitrary Waveform Generator (Bottom)

[4]Pray to our sweet lord Jesus Christ it works.

3. Find the necessary connectors:

   (a) A BNC-SMA cable (depending on the function generator) to connect an ACDC to the function generator.

   (b) A SAMTEC-SMA card.[5] See Fig. 7 for an example.

   (c) A cable to connect the function generator to an oscilloscope (depends on your function generator/oscilloscope, we used BNC-BNC).

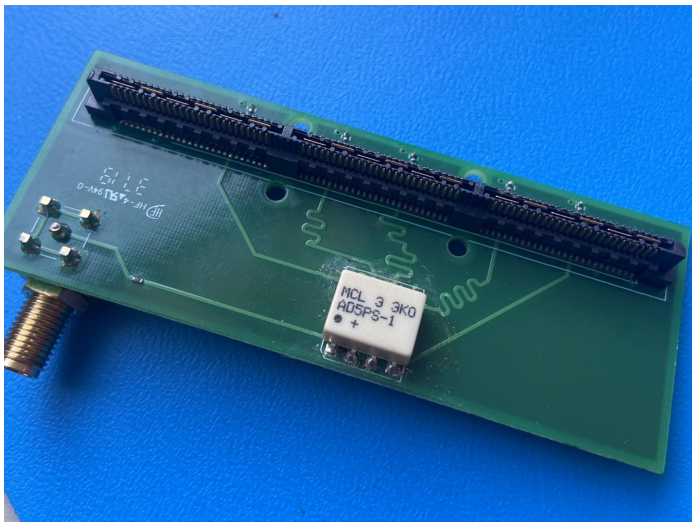


Figure 7: SAMTEC-SMA Card

4. Power on the ACDC and ACC.

5. Run the following command in your Linux machine:

   ./bin/setPed 2048

6. Run the command:

   ./bin/calibratePed

7. Generate a sine wave of at least 40MHz, 2Vpp on the function generator.

8. Verify that your function generator isn't busted with the oscilloscope

9. Run the command:

   ./bin/listenForData

   (a) Reset ACDCs and ACCs.

[5] Designed by Evan Angelico; the ones left in the UChicago electronics shop are all somewhat finicky.

(b) Use software triggers.

(c) Have a PPS ratio of 1.

(d) Use boards 0xFF.

(e) Do not use the calibration mode unless you are plugged into the calibration input.

(f) Do not use raw mode unless you have a good reason for it.

(g) Choose as many events as your heart desires (maybe 10).

10. Copy the pedestal file (Autogenerated Calibrations) and the data file (Results) from the Raspberry Pi to wherever you will run the plotting code.

11. If you are using our code, look at the README for further instructions on plotting.