ADC basics for TDAQ (Lab 8)

tutor lab setup guide

Andrea Borga (andrea.borga@nikhef.nl, tutor)

Cairo Caplan (cairo.caplan@cern.ch,lab assistance)

Table of Contents

[Lab setup 2](#_Toc428524847)

[PC and OS requirements 3](#_Toc428524848)

[Hardware installation 3](#_Toc428524849)

[Software tools and packages 3](#_Toc428524850)

[ROOT software 4](#_Toc428524851)

[Software installation 3](#_Toc428524852)

[Operating the lab 5](#_Toc428524853)

[Known issues 7](#_Toc428524854)

# Lab setup

*Figure 1 - Equipment setup*

TRIG

ADC

IN

1

2

3

4

ACQ

Fmc-Adc-100M-14b-4cha

BNC Connectors

LEMO Connector

T-BNC

AFG

Oscilloscope

ADC card

LEMO Connector

1. Tektronix AFG3252: Arbitrary Function Generator, is the **input** of our system
2. Tektronix DPO70404C: Oscilloscope, it the **monitor** of our system
(to cross-check that the signals fed to the system are truly what intended)
3. SPEC+FMC ADC card plus host PC: this is the **core ADC DAQ**
	1. FPGA Mezzanine Card (FMC) ADC: <http://www.ohwr.org/projects/fmc-adc-100m14b4cha>
	2. Simple PCI Express Carrier (SPEC) card: <http://www.ohwr.org/projects/spec/wiki>
	3. Linux based host PC


*Figure 2 - Equipment setup*

# PC and OS requirements

Minimum system requirements:

* CPU: Dual Core AMD Athlon 64 5600+ or higher
* RAM: 2GB DDR or higher
* HDU: 15 GB (~50 GB recomended)
* PCIe: 1 PCIe Gen1x4

Tested on SLC6 Carbon 6.6 and 6.7 64 bit.

# Hardware installation

The SPEC card is powered from the PCIe fingers, the power consumption is compliant to the PCIe specs. **There is no need to power the SPEC from its external 12V supply.** The FMC mezzanine plugs onto the carrier connector and gets power from it. The mezzanine plus carrier card assembly must be placed on a free PCIE x4 or higher slot. The assembly should look like *Figure 2*.

# Software tools and packages

The software packages and tools are provided in *lab8\_source.zip* that can be found in <https://espace.cern.ch/isotdaq-sharepoint/> under /Documents/Lab8-ADC\_Basics. They are mostly the same as the ones that can be found on the FMC ADC software repository, but cherry picked and tested.

The following folders are available in the lab8\_source folder:

1. Binaries – Folder with the Gateware files
2. Doc – Manual files of the standard FMC ADC software and API, gateware and driver.
3. Kernel – The FMC ADC software driver
4. Lib – Software libraries for the ZIO bus interface and for the FMC ADC driver.
5. Libtools – Standard plus some custom example software applications for this exercise.
6. Svec-sw – Software driver for the SVEC carrier board, not used for this lab.
7. Spec-sw - Software driver for the SPEC carrier board, PCIe carrier used in this lab.
8. Tools – Some software configuration tools.
9. Others – Old or not used previous software used for this exercise
10. FMC-bus – Standard FMC bus support software.
11. Scripts: unload\_drivers.sh, load\_drivers.sh, Makefile

# Software installation

The instructions of the software installation are also provided in the file INSTRUCTIONS\_SETUP.txt.

## Operating System

1. Install Scientific Linux 6.6 [or 6.7] (Carbon) for x86\_64 systems from here :

<http://ftp.scientificlinux.org/linux/scientific/6.6/x86_64/iso/SL-66-x86_64-2014-11-09-LiveDVD.iso>

[*optional*: install it with an user called “isotdaq” and password “isotdaq”]

1. Once the installation is complete, boot the system and check if the distro version is correct:

cat /etc/issue

The first line shall be : "Scientific Linux release 6.6 [or 6.7] (Carbon)"

1. Install some extra packages and repositories, as superuser:

yum install yumex

yum install yum-conf-sl6x

1. Installation of the Scientific Linux RealTime kernel:
	1. Download the Scientific Linux "Messaging Realtime Grid" repository file

wget [http://linuxsoft.cern.ch/cern/mrg/slc6-mrg.repo -O /etc/yum.repos.d/slc6-mrg.repo](http://linuxsoft.cern.ch/cern/mrg/slc6-mrg.repo%20-O%20/etc/yum.repos.d/slc6-mrg.repo)

yum groupinstall 'MRG Realtime'

yum install kernel-rt-devel kernel-rt-headers

Restart the PC:

init 6

* 1. If the machine has booted correctly, check if the current Linux kernel used is the Real Time patched which was just installed:

uname –r

 It must contain the ‘rt’ sequence, like in “3.10.0-229.rt56.158.el6rt.x86\_64”

1. Installation of extra packages (if not yet installed)
	1. GNU plot
	yum install gnuplot

## ROOT software

The ROOT version used on the setup is the version 5.34/32, which is the one available on the Fedora EPEL 6 repository.

1. Installation of CERN ROOT software packages:
	1. ~~Enable Fedora EPEL repository:~~

wget http://download.fedoraproject.org/pub/epel/6/x86\_64/epel-release-6-8.noarch.rpm

rpm -ivh epel-release-6-8.noarch.rpm

* 1. Install ROOT packages used for this exercise:

yum install root root-fftw root-guibuilder

* 1. You can test ROOT by trying to load it:

root –l

to quit .q

## ISOTDAQ software package

1. Check the ADC board enumerated on the PCI(E) bus, as superuser run:

lspci –vvv | grep CERN

Check if the output line contains the ‘CERN’ word, as in:

**01:00.0** Non-VGA unclassified device: CERN/ECP/EDU Device 018d (rev 03)

Note down the address of the device, here as **01:00.0**, the first two fields will be used by the acquisition programs as “**0x100**” or “**0100**”.

[If the card doesn’t show up, the lab kit comes with a sharp harakiri knife...]

1. Download the package from SharePoint **on a Linux host!** If you download it with windows and copy it over to your Linux host the archive might get unpredictably corrupt!
	1. <https://espace.cern.ch/isotdaq-sharepoint/Shared%20Documents/Lab8-ADC_Basics/lab8_source.zip> (with a web browser)
	2. wget <http://www.nikhef.nl/~aborga/lab8_source.zip> (from terminal directly)
2. As normal user, unzip it, this will create an ~/adc folder:

cd ~

unzip lab8\_source.zip

1. As a superuser:
	1. Create a /lib/firmware/fmc/ directory:

mkdir /lib/firmware/fmc/

* 1. Copy the gateware files to this folder:

cp ~/adc/binaries/spec-fmc-adc-v4.0.bin /lib/firmware/fmc/

cp ~/adc/binaries/spec-init.bin-2012-12-14 /lib/firmware/fmc/

1. As normal user:
	1. Go to the ~/adc folder and try to load the FMC ADC driver stack:
	cd ~/adc/
	./load\_drivers.sh
	The script will ask for the root password.
	The script prints the end of the dmesg command, check if the driver was correctly loaded or if there is any error message.
	2. In case unloading the driver is needed:
	./unload\_drivers.sh
	3. Now you are ready to play with the ADC FMC board, the example software is located on the ~/adc/libtools.

## Recompiling the driver from sources

1. Download the package from SharePoint on a Linux host! If you download it with windows and copy it over you linux host the archive might get unpredictably corrupt!
	1. <https://espace.cern.ch/isotdaq-sharepoint/Shared%20Documents/Lab8-ADC_Basics/lab8_source.zip> (with a web browser)
	2. wget <http://www.nikhef.nl/~aborga/lab8_source.zip> (from terminal directly)
	3. unzip lab8\_source.zip
2. the source binaries were compiled for kernel 3.10.0-229.rt56.158.el6rt.x86\_64, this might be updated after this manual had been written (in fact it already did...)
	1. cd into the adc folder
	2. make clean && make
	[if this fails... remember about the harakiri knife mentioned above...]

# Operating the lab

With the software installation step completed, you should be able to test the equipment for this lab. This part can is divided in three:

* Setup the equipment: see “Lab Setup”
* Load the drivers: see “Software installation”
* Run test software: read below

**NOTE:** always check the content of programs **before** executing them!

Please also refer to the *lab8-form\_1.1.pdf* from here on.

**Basic acquisition and control programs**

* Trigger\_ext:
	+ Trigger\_ext/fald-acq.c
		- ./fald-acq -a 1000 -b 0 -n 1 -l 1 -g 1 –r 10 -e -X **0100**
		- AFG value with default scale setting
			* CH1 sine 1MHz 380 mVpp
			* CH2 pulse 5 Hz high 2.7 V low 0 V, Duty 2%
	+ /libtools/acq-program.c

One shot of such programs should give a single GNU plot with the current wave form fed to the ADC.

* Trigger\_int:
	+ Trigger\_int/fald-simple-acq.c
* Low level card control scripts:
	+ Try for example: ./set\_termination\_ch1.sh (enables 50 Ohm input termination)
	+ /adc/tools/fau-config-if (GUI to enable 50 Ohm input termination)

**Basic ROOT programs**:

Running them those as simple as typing

root -l program\_name

exiting root ctrl-c then .q

* Trigger\_ext:
	+ V\_t\_continuous.c: *trigger source to ext\_trigger*
		- query replace 0100 with your\_board\_ID
		- query replace 0x100 with 0x0\_your\_board\_ID
	+ V\_t\_continuous\_CH2.c: *trigger source to CH2*
		- query replace 0100 with your\_board\_ID
		- query replace 0x100 with 0x0\_your\_board\_ID
		- AFG value with default scale setting
			* CH1 sine 10MHz 70 mVpp
			* CH2 pulse 5 Hz high 1.4 V low 0 V, Duty 2%
* Trigger\_int:
	+ V\_t\_continuous\_INT.c: *don’t go beyond 1 MHz!*
		- query replace 0100 with your\_board\_ID
		- query replace 0x100 with 0x0\_your\_board\_ID
		- AFG value with default scale setting
			* CH1 sine 200 kHz 400 mVpp
			* CH2 OFF
* Demonstration : /libtools/adc\_gui.Cpp
* FFT root program: others/fft.C

# Suggestions to tutors

Here is a list of useful suggestions for guiding students through this lab:

1. remind the students about the -r parameter in point 5
2. make them study the impact of the input range selection vs the resolution of the ADC measurement.
3. Compute the ENOB by terminating the input of CH1
4. Compare the resolution vs. sampling rate of the scope ADC (probably 8 bit)
5. Run in different trigger modes: external "LHC like" mode, internal "astroparticle like" mode
6. Do they realize that the ADC input is not set to 50 ohm by default? Show them the effect of reflections
7. Do they realize that the ADC input bandwidth is limited to 30 MHz? Show them how this affects the shape of square waves and the attenuation of harmonics
8. Show the effect of aliasing by just cranking up the frequency
9. Remember that the precooked fft script needs a tone (sine) or square in the range of the 2 MHz

# Known issues

Here a list of issues experienced in lab8 2015:

1. Fald programs are not stable for acquisitions using a trigger faster than 1 MHz.
2. Fald programs will get stuck if their trigger source, internal or external, is not enabled on the AFG. This can even cause a system crash when using the fals-acq program, as it is supposed to run in a loop. Workaround: run the program, then enable the trigger source; stop the program, then disable the trigger.
3. Continuous acquisition using the Fald-acq results in a time sliding output/plot. This is probably related to how the external trigger is synched with the ADC clock at the FPGA input. Workaround: use V\_t\_continuous\_CH2.c.
4. Fald-simple-acq does not have a range setting parameter. Workaround: feed the ADC a signal with a range < 100 mVpp.
5. The outputs of fald-simple-acq and acq-program (currently) are shown in ADC counts, while fald-acq’s outputs are already converted into volts. Workaround: ask students to work on conversion in root.
6. The ROOT program adc\_gui.C , as the others ROOT programs in the lab, becomes unresponsive if the acquisition mode is running but the trigger condition is not met. So, before turning off the trigger source on the AFG you should press the “Stop Acquisition” button.