PETsys Electronics

TOFPET2 ASIC Evaluation Kit

Software user guide v2017.04

Contents

Release notes

Version 2017.06

First version of Software package and Software User Guide for TOFPET2 ASIC.

Chapter 1

System Overview

For future updates on software and firmware, as well as drivers, documentation and table maps, please check the Client Package area under the URL link:
https://drive.google.com/folderview?id=0Byh2q2A3bm1hQk5HZGROYWFURUk
The following sections provide a brief overview of the hardware and software specifications.

1.1 TOFPET2 ASIC Evaluation Kit (before February 2017)

If the user is upgrading from a TOFPETv1 evaluation kit bought before February 2017, the system is composed of (as illustrated by figure 1.1):

- A Power Adapter Board (PAB), which can hold up to two TOFPET2 ASIC test boards.
- Two TOFPET2 ASIC test boards and respective adaptors (2 adapters needed per board)
- A Xilinx ML605 Virtex6 evaluation kit.
- A computer, running 64-bit Linux.

The two TOFPET2 ASIC test boards contain one ASIC each, SiPM connectors and some power supply and filtering.

The PAB provides the test boards with power, bias voltage for the SiPM, a filtered low jitter clock (200 MHz) and a FPGA interface.

The ML605 FPGA implements the communication logic necessary to configure the ASICs, the clock filter and the HV DAC which produces the bias voltages, as well as reading out the data produced by the ASIC. It can also generate a test pulse, with an adjustable delay in relation to the ASIC reference clock, which is delivered to the ASICs through the PAB interface but is also available, as a 2.5 V LVCMOS signal, on the GPIO_USER_P connector on the ML605 board. This can be used for various measurements, from triggering the TDCs inside the ASIC with known test pulse to triggering a laser driver.

An external test pulse can also be used, by providing a 2.5 V LVCMOS signal on the GPIO_USER_N connector on the ML605 board.

Communication between the FPGA and the computer is handled via Gigabit Ethernet.

1.1.1 Programming the ML605 FPGA board

Instructions to program the ML605 FPGA board can be found in http://e-lab.github.io/html/wiki-program-ml605-poweron-flash.html or http://www.xilinx.com/support/documentation/application_notes/xapp518-isp-bpi-prom-virtex-6-pcie.pdf. The firmware (.bit) files to be uploaded are provided by PETsys Electronics in the Client Package folder.

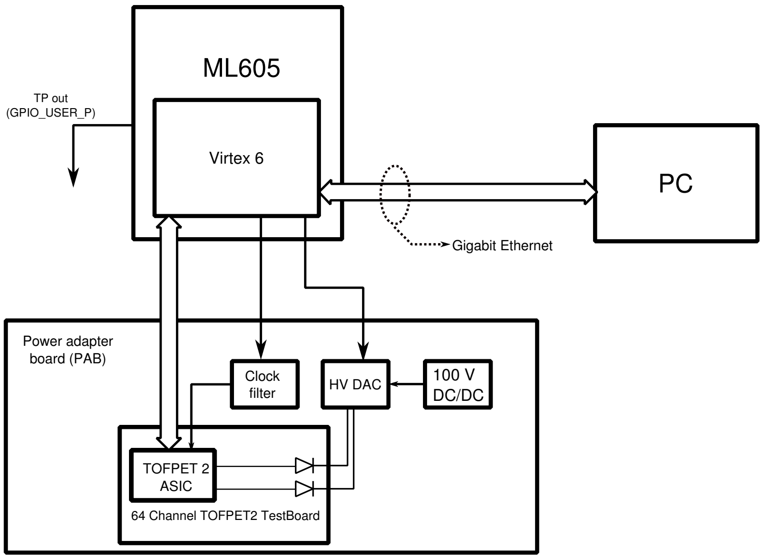

Figure 1.1: System Diagram - TOF ASIC Evaluation Kit

1.1.2 Hardware switch and FMC cable connection

The user must ensure that the ML605 SW1 switch bank has a proper configuration.

Position 1 and 2 configure clock enables and, for testing purposes, they allow the clock to the ASIC to be gated off. For normal operation, the configuration must be:

- Position 1 set to OFF
- Position 2 set to ON

Position 3 enables the external test pulse source. The GPIO_USER_N will see a logic HIGH when left floating. Unless a source of external test pulses is active and being used, position 3 of SW should be set to OFF.

Also, make sure that the FMC cable connecting to the PAB is placed on the LPC connector (J63) of the ML605 board.

1.2 TOFPET2 ASIC Evaluation Kit (after February 2017)

If the user is upgrading from a TOFPETv1 evaluation kit bought after February 2017 or has just purshased a new TOFPET2 Evaluation Kit, the system is composed of (as illustrated by figure 1.2):

- Front end boards - type D (FEB/D), each being able to hold up to 8 TOFPET2 ASIC test boards. The FEB/D have a HVDAC module and a Gigabit Ethernet mezzanine, allowing to connect to the computer with GBE.
- Two TOFPET2 ASIC test boards and respective adaptors and cables (1 adapter and 1 cable per board)
- A computer, running 64-bit Linux.

The TOFPET2 ASIC test boards contain one ASIC each, SiPM connectors and some power supply and filtering.

The FEB/D and modular mezzanines provide the TOFPET2 ASIC test boards with power, bias voltage for the SiPM, a filtered low jitter clock (200 MHz) and a FPGA interface, which implements the communication logic necessary to configure the ASICs, the clock filter and the HV DAC which produces the bias voltages, as well as reading out the data produced by the ASIC. It can also generate a test pulse, with an adjustable delay in relation to the ASIC reference clock. This can be used for various measurements, from triggering the TDCs inside the ASIC with known test pulse to triggering a laser driver.

Communication between the FEB/D FPGA and the computer is handled via Gigabit Ethernet.

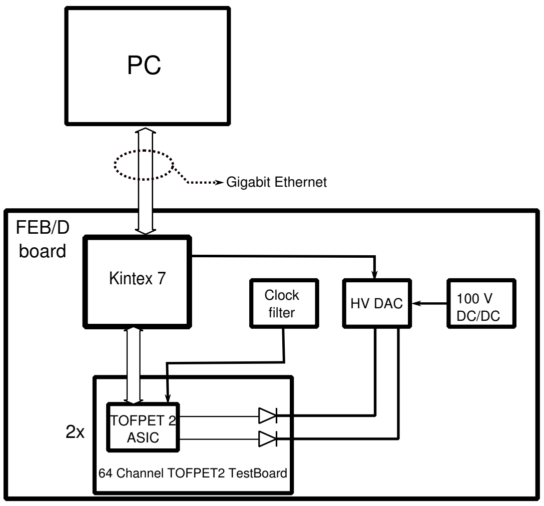

Figure 1.2: System Diagram - SiPM Readout System

1.2.1 Programming the FEB/D FPGA

Instructions to program the FEB/D FPGA can be found in the manual “TOF FEB-D_v2 - Hardware User Guide (v1.0)”. The firmware (.bit) files to be uploaded are provided by PETsys Electronics in the Client Package folder.

1.3 Computer requirements

The data acquisition computer must be capable of running 64-bit Linux and must have one available Gigabit Ethernet port. The Gigabit Ethernet interface only works at 1 Gbit/s, it is not capable of communicating with slower 10 or 100 Mbit/s ports.

This Ethernet port needs to be configured with the following parameters.

- IP address: 192.168.1.1
- Netmask: 255.255.255.0

The operative system must be 64-bit Linux, preferably CentOS 7 or higher versions (http://www.centos.org/). If using CentOS 7, a script is provided in the Software package, which installs all software dependencies and requirements. After downloading it, just navigate to the corresponding folder, and type:

```
su
sh INSTALL_SW_REQUIREMENTS_CENTOS.sh
```

For reference, the full list of requirements to compile PETsys Electronics software is described below:

- C++ development tools (GCC, Make, libstdc++, etc). If using CentOS 6.x or similar, choosing the “Software Development Workstation” profile should install most of them.
- cmake
- ROOT (http://root.cern.ch/) – tested with v5.34/09. It can also be installed from the EPEL repository (https://fedoraproject.org/wiki/EPEL).
- Python 2.6. If using CentOS 6.x or similar, it should come as part of the base installation.
- Python bitarray module (https://pypi.python.org/pypi/bitarray/ – tested with v0.8.1.

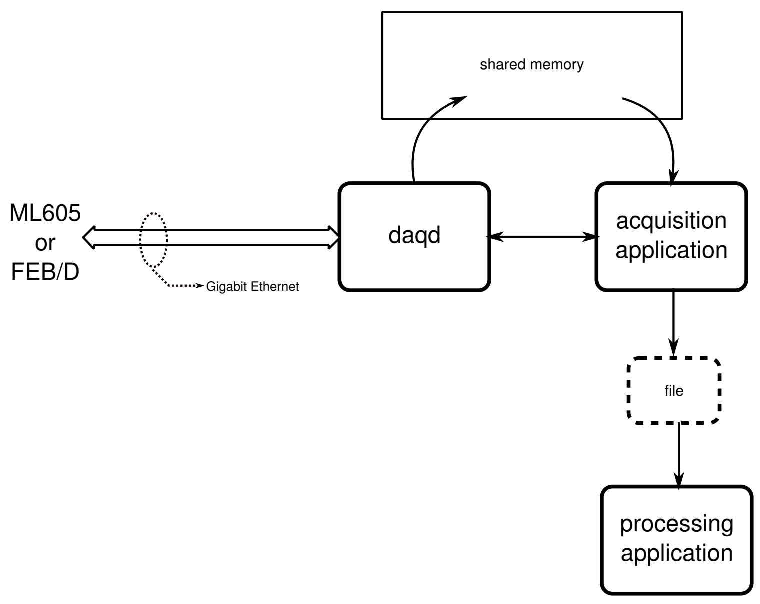

Figure 1.3: Software diagram

- Python CRC module (https://pypi.python.org/pypi/crcmod) – tested with v1.7.
- Python serial module (http://pyserial.sourceforge.net/pyserial.html) – tested with v2.7.
- Python POSIX IPC module (https://pypi.python.org/pypi/posix_ipc) – tested with 0.9.4
- Python numpy module (https://pypi.python.org/pypi/numpy) – tested with 1.4.1
- Python argparse module (https://pypi.python.org/pypi/argparse) – tested with 1.3.0

1.3.1 ROOT

If installing ROOT from packages, note that the following packages need to be installed.

- root-gui-fitpanel
- root-spectrum
- root-spectrum-painter
- root-minuit2
- root-python

1.4 Software

The computer is used to run a suite of software in order to configure the system and process the data produced by the ASICs.

As shown in figure 1.3, the acquisition software consists of three main parts.

"daqd" is a communication server which must always be running in order to communicate with the hardware. Its main purposes are to abstract away the communication method and to act as a buffer to the data provided by the hardware.

The acquisition applications, communicating with the hardware through "daqd", configure the hardware and acquire the data into files as needed.

Finally, processing applications are used to process and analyze the acquired data.

Communication between “daqd” and the acquisition applications is done by a UNIX socket (“/tmp/d.sock”) and a POSIX shared memory block (“/dev/shm/daqd_shm”).

IMPORTANT NOTE: If the user is already acquainted with the PETSYS Electronics Software aplications for TOF ASIC (TOFPET1), please note that the software complilation and usage for data acquisition and processing has significantly changed for TOFPET2. This will require some re-adaptation from the user, hence the importance of reading the following of the present guide.

1.5 Before first use

After downloading the software archive from the Client Package folder, extract it by typing in a command-line shell:

```
tar -jxvf TOFPET2_SOFTWARE_v2017.06.tar.bz2
```

The next step is to compile the C++ applications. Navigate to the newly created base folder (“sw_daq_tofpet2”) of the decompressed software. Then execute the following commands:

```
cmake .
make
```

The user should make sure that all the hardware is properly connected and powered and the correct firmware is loaded in to the ML605 FPGA (older eKit version) or the FEB/D FPGA (recent eKit version). The ethernet cable should be plugged and the PC Ethernet card configured as described as above.

Chapter 2

Quickstart

This chapter is meant to guide the user to quickly acquire data with your recently acquired hardware, process this data and search for coincidence events, drawing charge/energy spectra and coincidence time resolution (CTR). For a more detailed description on each step please address to the next chapters. It is expected that the user has a point source isotope emmiting 2 Gamma photons of 511 keV, typically a 22Na point source with an activity higher than 10 μCi (370 kBq). The Gamma photons are detected by either a single or an array of SiPMs coupled to scintillation crystal(s) and connected to PETsys Electronics TOFPET2 ASIC Testboards via the SAMTEC connector(s). In order to measure events in coincidence, the 22Na source should be placed in a virtual line between a pair (or several pairs in the case of two arrays) of crystal+SIPM pixels. If using a recent eKit version with a FEB/D, the TOFPET ASIC test boards should be connected in ports F1 and F3.

The setup should be placed inside a dark box to avoid any light contamination and preferably with some sort of active cooling (a cooling system is provided for eKits bought after February 2017). If no active cooling is possible, the user should at least place a small fan inside the dark box to avoid heat concentration around the ASIC and SiPMs.

IMPORTANT NOTE: In order to avoid any damage to the hardware and ASICs, always power off the system when plugging or unplugging any hardware and never perform data acquisitions with SIPMs with ambient lighting.

2.0.1 Launching “daqd”

Before any other communication with the hardware is attempted, “daqd” must be running. Information about the type of connection used for communication is requested by this application. Navigate to the software base folder and execute the following command:

```
./daqd --socket-name=/tmp/d.sock --daq-type=GBE
```

The expected behavior is that “daqd” remains running instead of returning to the command line. If a successful communication is established with the FPGA, the initial output of “daqd” should read (the size of frame may be different than this example):

```
Size of frame is 16400
Got FPGA reply
FrameServer::startWorker called...
FrameServer::startWorker exiting...
UDPFrameServer::runWorker starting...
```

Otherwise, the user should check that all cables are properly connected and that the firmware is properly installed. If desired, “daqd” can be terminated by pressing Ctrl-C.

If “/tmp/d.sock” or “/dev/shm/daqd_shm” already exist in the computer, “daqd” will complain and exit. If such is the case, the user should manually remove those files. Since “daqd” is the communication server, the terminal/shell in which it is running should not be closed while trying to acquire or communicate with the hardware.

2.0.2 Configuring the clock filter (only for PAB version eKits)

After power up and launching “daqd”, the first task is to configure the clock filter chip, so that the ASICs will receive clock signals. In a command shell, navigate to the software base folder and execute the following command:

```
python setSI53xx.py SI5326_config.txt
```

The expected behavior is that it will print a series of messages like the following and then return to the command shell.

```
Register 8f set to 06
Register 88 set to 40
Done
```

2.1 Configuration and calibration

PETSYS Electronics provides a file with measured voltages taken with the ad5535 HV DAC, allowing to calibrate and ensure that any voltages set by the software are correctly set by the hardware. The user must start by editing the file “config_example.ini” and set in line number 2 (in the field “ad5535_calibration_table”) the correct path to the file that was provided. Also, if using a recent version of the evaluation kit with FEB/D, the user should uncomment lines 14 and 15 of “config_example.ini” (delete the “#” character). The resulting file should be saved as “config.ini” in the software base folder.

It is also advisable to create a folder where to store data. For example, type (from the software base folder):

```
mkdir data
```

Before acquiring any data, the user should check what is the recommended overvoltage to bias the SIPMs that are connected to the TOFPET2 ASIC Testboards. This information needs to be put into a table to serve as reference for the software. PETSYS Electronics has provided two types of individual 3x3mm SIPMs to costumers. If using SIPMs KETEK PM3350TP-SB0 just type:

```
./make_simple_sipm_bias_table --config config.ini --prebd 20 --bd 25 --over 4
> data/sipm_bias.tsv
```

If using KETEK PM3325-WB, type:

```
./make_simple_sipm_bias_table --config config.ini --prebd 20 --bd 27 --over 4
> data/sipm_bias.tsv
```

If using other SIPMS, type:

```
./make_simple_sipm_bias_table --config config.ini --prebd 20 --bd BD --over OV
> data/sipm_bias.tsv
```

where “BD” is the breakdown voltage (in Volts) of the SIPM connected and “OV” is the overvoltage to be aplied to bias the SIPM (in Volts).

Some of the internal TOFPET2 ASIC circutry, like the TDCs and QDCs, require calibration. The procedure is simple and fast, and the user can perform it by running the bash script that is provided by PETSYS Electronics:

```
sh configuration.template.sh
```

This will run a suite of calibration scripts, and should take 20 to 30 minutes to finish.

2.2 Data Acquisition with a radioactive Na22 point source

The user is now ready to acquire data with the evaluation kit. To do so, type:

```
./acquire_sipm_data  --config config.ini -o data/my_data --time 150   --mode
qdc
```

The first argument above is the configuration reference file, the second argument is the prefix for the output data. The third argument is the acquisition time. Depending on the activity of the 22Na source, this may need to be larger than the 150 seconds of this example. The last argument sets the ASIC operating mode for pulse amplitude measurement, in this case using the internal charge integrator. At the end of the aquisition, the script will print to screen a small report with the number of registered events. If this number is 0, the user must make sure that the correct bias voltage is being set.

2.2.1 Coincidence events and Coincidence time resolution

If the Na22 source was placed in between two SIPM detectors, one can filter the data for coincidence events (events that correspond to the detection of two gamma rays within a very short time window and different trigger regions) using the data processing script "convert_raw_to_coincidence". To do so, type:

```
 ./convert_raw_to_coincidence --config config.ini -i data/my_data
-o data/my_data_coincidences --writeBinary
```

The file "data/my_data_coincidences" contains a list (here, in binary format) with channel ID, time of arrival and energy (QDC measurement) for each of the two events in coincidence. The user can then run the script "psDrawCTR.C" to plot the energy spectra on both channels and coincidence time difference distributions. This can be done by typing:

```
root -l psDrawCTR.C+'("data/my_data_coincidences")'
```

An example of the output is presented in fig. 2.1. The script plots three histograms: two energy distributions corresponding to the pair of channels with more coincidence events. In each energy distribution a gaussian fit is performed to determine the position and width of the photopeak corresponding to the 511 keV gamma photons. In the middle panel, it is plotted the histogram for the time of arrival difference between both coincidence events. For this distribution only events within 1 sigma of both energy photopeaks are selected. The fit of a gaussian model to this distribution determines the time of flight and also the Coincidence Time Resolution (CTR) for that channel pair. In this example the CTR is 100.5 ps sigma, which corresponds to 236 ps FWHM.

The present chapter hopefully allowed the user to quickly obtain data and CTR results from your new hardware. The following chapters will describe in more detail the routines and procedures required for both data acquisition and data processing from TOFPET2 ASIC.

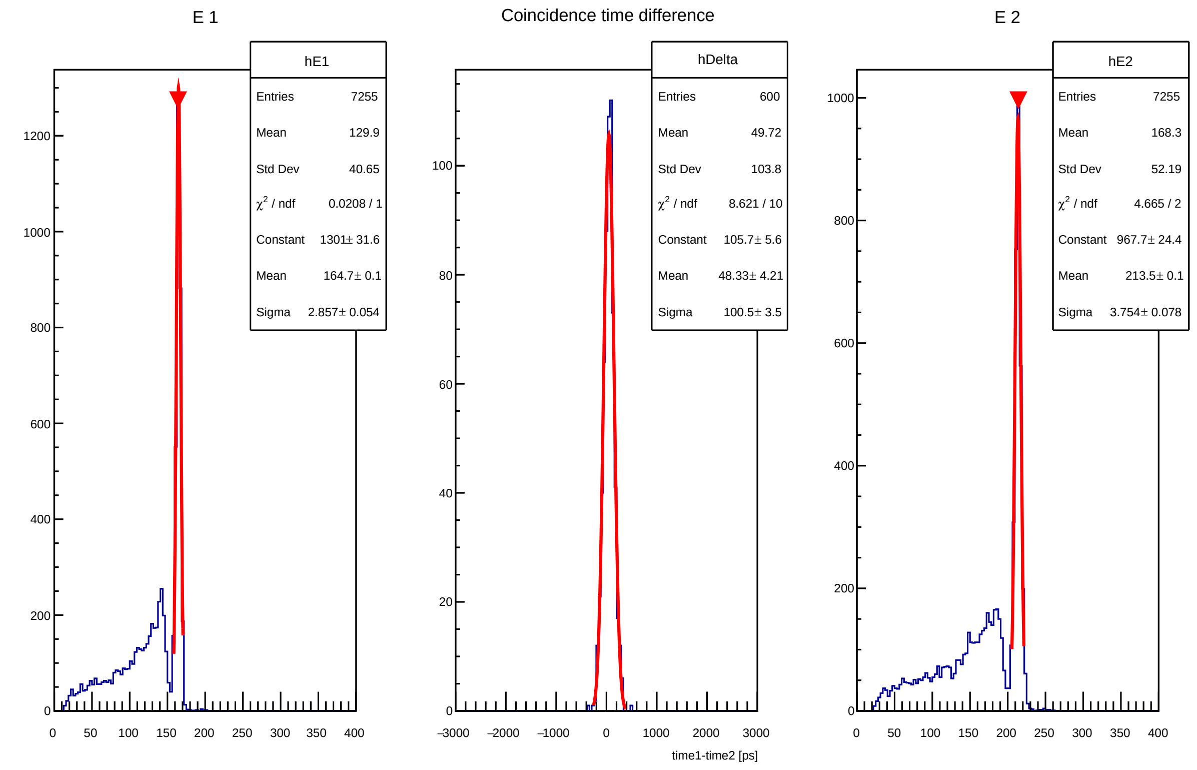

Figure 2.1: Example for the output of "aDAQ/psDrawCTR.C", showing ToT spectra and coincidence time resolution (CTR) for on channel pair and two KETEK PM3325-WB coupled to 3x3x5mm LYSO crystals.

Chapter 3

Reference Configuration File and Calibration procedure

In order to operate properly, TOFPET2 ASICs and external HV DACs requires some level of configuration and calibration. Configuration files are required to acquire data and calibration files are used to obtain "correct" information from the raw data provided by the ASICs. In the present software architecture, all the files and parameters required for configuration and calibration are referenced in a single file: the Reference Configuration file. It is a required argument for almost all software applications, either for data acquisition or data processing. If the user is familiar with the software package used to operate the TOF ASIC (TOFPET1), one could say that the new reference file replaces the "update_config.py" and "loadLocalConfig.py" scripts, as well as the "M1M2.cal" file.

An example for such file is provided in the software package and is show in listing 3.1.

Listing 3.1: config_example.ini

```
[main]
ad5535_calibration_table = pabXX_ad5535_cal.tsv

tdc_calibration_table = data/tdc_calibration.tsv
qdc_calibration_table = data/qdc_calibration.tsv
disc_calibration_table = data/disc_calibration.tsv

sipm_bias_table = data/sipm_bias.tsv
disc_settings_table = data/disc_settings.tsv

channel_map = petsys_tables/pab_tofpet2tb_fake_channel_map.tsv
trigger_map = petsys_tables/pab_trigger_map.tsv

#channel_map = petsys_tables/febd_tofpet2tb_fake_channel_map.tsv
#trigger_map = petsys_tables/febd_gbe_trigger_map.tsv


[hw_trigger]
type = builtin
threshold = 1
pre_window = 4
post_window = 20
coincidence_window = 3
single_fraction = 0

[asic_parameters]
global.disc_lsb_T1 = 57
```

The file is divided in two sections. The “[main]” section (lines 2-20) sets the configuration and calibration files, as well as map tables. These are:

- “ad5535_calibration_table” - Calibration table for the high voltage DAC present on the PAB or FEB/D. Allows to provide correct bias voltage to SIPMs (up to 10 mV). It is provided by PETSYS Electronics.
- “tdc_calibration_table” - Calibration table for the time-to-digital converters (TDC) present in each channel. Allows to correct the very low level of non-linearity in the correspondence between TDC codes and reference clock phase. Created by a calibration application provided by PETSYS Electronics.
- “qdc_calibration_table” - Calibration table for the charge integrator (QDC) present in each channel. Created by a calibration application provided by PETSYS Electronics.
- “disc_calibration_table” - Calibration table for the 3 discriminators present in each TOFPET2 channel: t1, t2 and E. It contains the information for the baseline and noise level for each of the 3 discriminators, as well as dark count rate at different thresholds. Table obtained by running a calibration script provided with the software package.
- “sipm_bias_table” - Configuration table containing 3 voltages to be defined for each HV DAC channel and applied to the SIPM that are connected to the TOFPET2 ASIC Testboards. The 3 voltages are: “Voff”, a voltage below the breakdown voltage of the SIPM used by some calibration applications to acquire data with test pulses while having the SIPM capacitance as close to normal operation as possible. “Vbd” is the breakdown voltage of the SIPM which serves as reference to set the overvoltage “Vover” to be applied in order to bias the SIPMs when acquiring data. It is defined by the user, but a script is also provided to create this table in the case where all channels have the same value on a given discriminator (see below).
- “disc_settings_table” - A configuration table that defines the threshold levels for each of the three discriminators (the typical trigger mode only uses two of the discriminators). It is defined by the user, but a script is also provided to create this table in the case where all channels have the same value on a given discriminator.
- “channel_map” - A map table used by the post-processing applications to obtain information relative to a given channel ID. It can be defined by the user, since some of its elements depend on the system geometry, but a version with fake values regarding SIPM physical geometry and pixel position is provided by PETSYS Electronics.
- “trigger_map” - A map table used by the post-processing applications to identify the possible channel ID regions where to search for grouped events or coincidence events. Also provided by PETSYS Electronics.

All the files described above consist of TAB separated value collumns allowing the user to easily read or modify them.

The second section of the reference configuration file (“[asic_parameters]”) is reserved to change the default values for the TOFPET2 ASIC parameters. PETSYS Electronics only recomends to change 5 parameters, related to two functionalities: trigger mode (4 “channel.trigger_mode_2_*” parameters) and the t1 discriminator LSB value (“global.disc_lsb_T1”). Although several trigger modes are available, we advice to operate the ASIC in the default mode which uses the t1 discriminator for time stamp, t2 for charge interation start, and E discriminator for event filtering based on their amplitude. As for the t1 discriminator LSB, it should be set accordingly to the gain of the SIPM and the pulse shape. According to the tests done by PETSYS Electronics, a proper working value is 57 (which corresponds to 5mV per DAC step) for most SIPMs except for KETEK PM3325-WB, in which case it should set to 60 (corresponding to 2.5mV per DAC step). This choice implies that the signal for each photoelectron corresponds to 5 ADC steps of the threshold DACs.

3.1 Configuration and calibration

Each row on the above mentioned table files usually correspond to a different channel or TAC (either ASIC channel or HV DAC channel, depending on the file), and the ID of such channels depends on which port of the FEB/D the TOFPET2 ASICs are connected. This means that the configuration tables are valid if and only if they were obtained for a given configuration (connection of TOFPET ASIC Test boards to FEB/D ports) and this configuration is not changed. Is it IMPORTANT to create new configuration files whenever this is not verified, as well as when using the hardware for the first time (as in the previous chapter of this guide). The time to perform the full configuration/calibration proceadure for TOFPET2 ASIC has been much reduced when compared to TOPFPET ASIC, ie, the user can obtain new calibration files in about 30 min. To do so, use the template script provided with the software and shown in listing 3.2.

Listing 3.2: configuration.template.sh

```
1  #!/usr/bin/sh
2  CONFIG_FILE=config.ini
3  DATA_DIR=data
4
5  ./acquire_threshold_calibration --config ${CONFIG_FILE} -o ${DATA_DIR}/
       disc_calibration
6  ./process_threshold_calibration --config ${CONFIG_FILE} -i  ${DATA_DIR}/
       disc_calibration -o ${DATA_DIR}/disc_calibration.tsv
7
8  ./make_simple_disc_table --config ${CONFIG_FILE} --vth_t1 20 --vth_t2 20 --
       vth_e 15 > ${DATA_DIR}/disc_settings.tsv
9
10 ./acquire_tdc_calibration --config ${CONFIG_FILE} -o ${DATA_DIR}/
       tdc_calibration
11 ./acquire_qdc_calibration --config ${CONFIG_FILE} -o ${DATA_DIR}/
       qdc_calibration
12
13 ./process_tdc_calibration --config ${CONFIG_FILE} -i ${DATA_DIR}/
       tdc_calibration -o ${DATA_DIR}/tdc_calibration
14 ./process_qdc_calibration --config ${CONFIG_FILE} -i ${DATA_DIR}/
       qdc_calibration -o ${DATA_DIR}/qdc_calibration
```

This template runs a series of applications that allow to create the three calibration tables described above ("disc_calibration_table", "tdc_calibration_table" and "qdc_calibration_table"). The input ("-i") and output ("-o") file prefixes and paths in this template are examples and can be changed by the user as/if desired.

Before running the calibration script, the user should create the SIPM bias voltage table ("sipm_bias_table") for the SIPMs that is plugged in the test board. For example, if using KETEK SIPM (delivered with recent versions of the PETSYS Electronics Evaluation Kit), the user can create a new table by typing:

```
./make_simple_sipm_bias_table --config config.ini --prebd 20 --bd 27 --over 4
> sipm_bias.tsv
```

In this example, the pre-breakdown voltage is being set to 20V, the breakdown voltage to 27V and the overvoltage to 4V (operational voltage will be 27V + 4V = 31V). These values will be set by the High Voltage DAC in all the channels whenever required by the software. If using different SIPMs (or wanting to operate the SIPM at a different overvoltage), the user should create a different file accordingly.

The user can now run the configuration script by typing (assuming the default name for the script has not been changed):

```
sh configuration.template.sh
```

3.1.1 Discriminator calibration

Lines 5 and 6 in listing 3.2 correspond to the data acquisition and processing for discriminator calibration. The data acquisition consists in a scan of the discriminator threshold in order to find the baseline and noise estimation (with the SIPM bias voltage set to a value below breakdown voltage). The script also performs a dark count scan, by performing the same threshold scan but this time with the SIPM biased at operational voltage. The data is processed in the last line which also allows to obtain the calibration table "data/disc_calibration.tsv". The

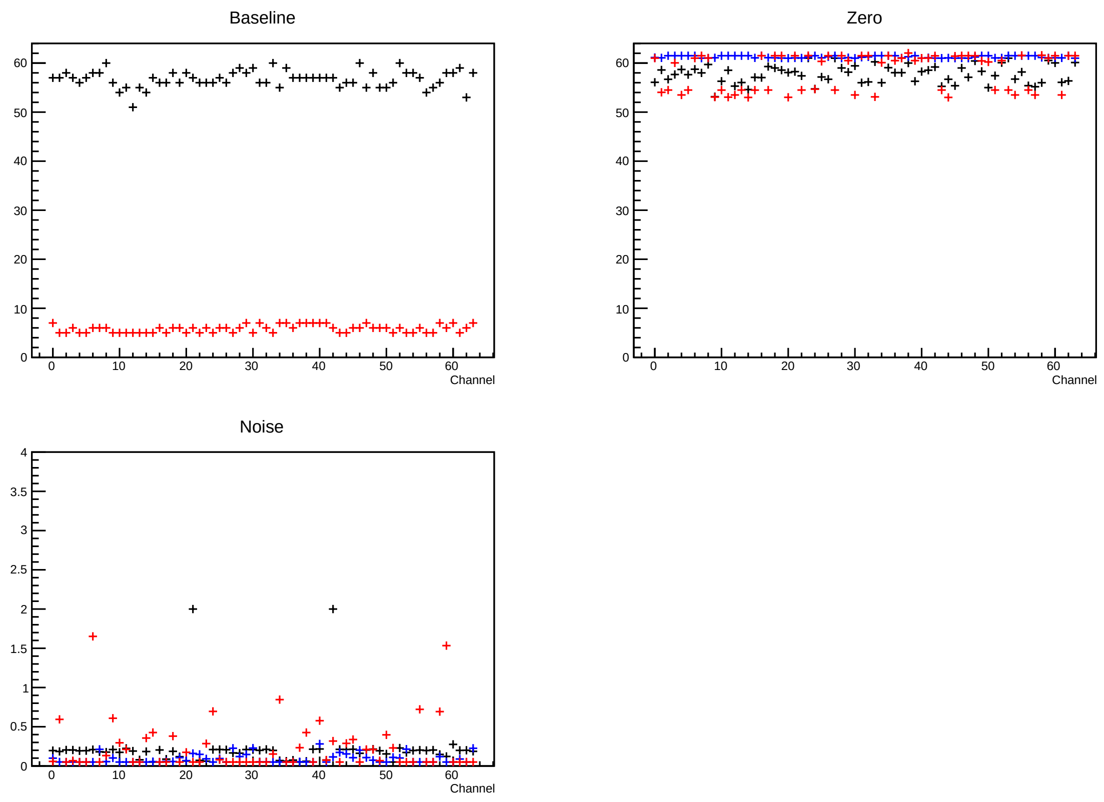

Figure 3.1: Discriminator calibration summary plots for one ASIC. On the top left plot, the value of the amplifiers' adjustment DAC after convergence, per channel. On the top right plot, the value of the threshold level which corresponds to the amplifiers' baseline, in DAC units, per channel. The bottom left shows the noise level measured per channel, again in DAC units. Different colors represent data from the 3 discriminators: t1 (black), t2 (blue) and E (red).

summary plot containing the values for the amplifiers' adjustment DAC, baseline and noise per channel are created and should have data on all channels without a significant dispersion.

Line 8 defines the threshold values for the different ASIC discriminators to be used in the remainder odf the calibration procedure (TDC and QDC).

3.1.2 TDC calibration

The tenth line in listing 3.2 launches a data acquisition from the ASICs with test pulses triggering the TDC at different known phases relative to the reference clock. Line number 13 processes and fits the obtained data in order to determine the calibration function mapping measured TDC codes and time. It writes the parameters for those functions in the TDC calibration table, in this example the resulting file is "data/tdc_calibration.tsv". The application will also create a series of ROOT files (containing all the histograms and data fits) and PDF files (sumarizing the time resolution obtained after calibration with the test pulse triggered data), one for each ASIC connected. The summary plot for each ASIC should resamble the one on Fig. 3.2, with data on all 256 TACs (4 TAC per channel) for each discriminator and the time resolution after calibration on all of them below 0.01 clock period (RMS). If the user obtains a significantly different plot, the user should contact the PETSYS Electronics support team.

3.1.3 QDC calibration

Line number 11 of listing 3.2 acquires data for calibration of the charge integrator (QDC), by varying the integration window of test pulse driven events. The data is then processed in line number 14, in order to dermine the offset current to be removed when integrating the charge of an event. As with the TDC calibration, the output

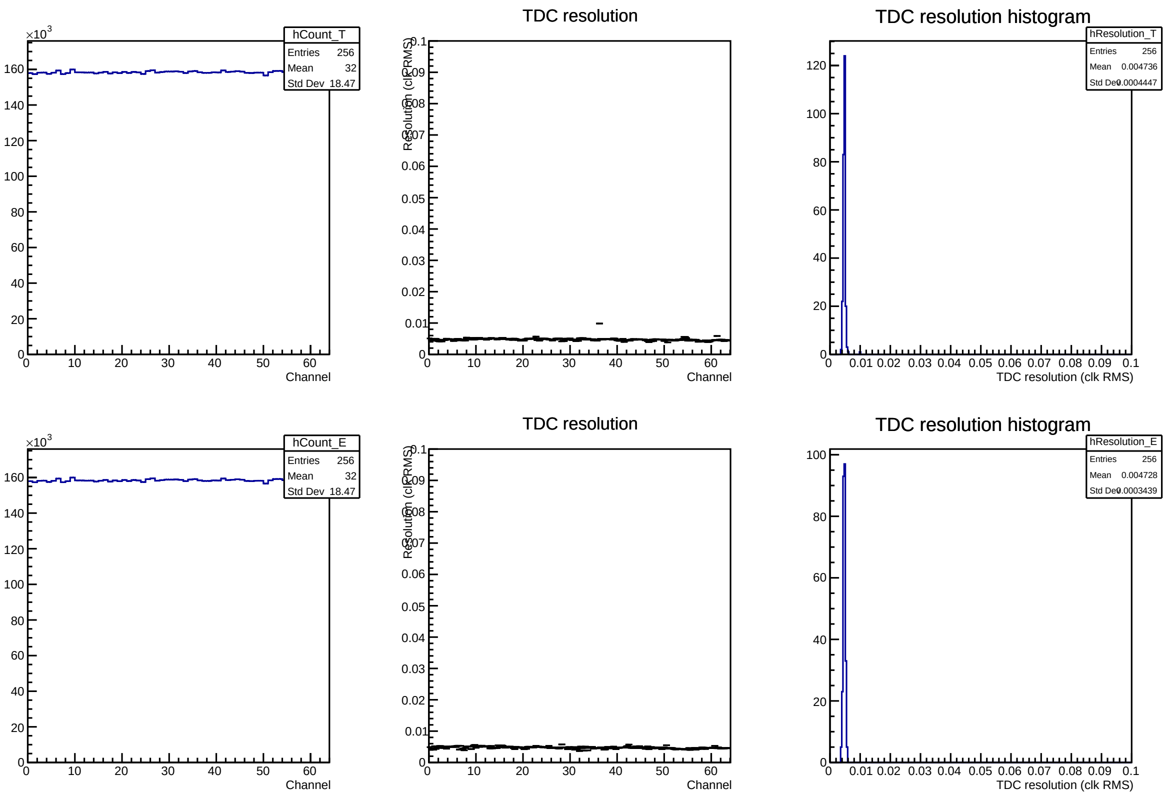

Figure 3.2: TDC calibration summary plots for one ASIC. The upper plots show the number of test pulse events used for calibration (left), the distribution per channel of the obtained TAC resolution after calibration (middle) and the overall distribution of resolutions obtained after calibration (right), for the first discriminator (t1 in default trigger mode). The plots on the bottom are the same but for the second discriminator (E in default trigger mode).

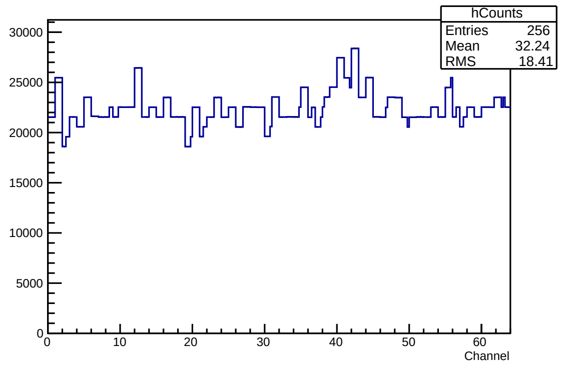

Figure 3.3: QDC calibration summary plot for one ASIC. The number of events used to calibrate each channel are plotted.

table file to be included in the reference configuration file is, in this example “data/qdc_calibration.tsv”. A plot with the number of test pulse events will be created, to ensure that all channels had sufficient data to calibrate correctly.

3.2 Map tables

The channel map table defines the trigger region each channel belongs to, as well as providing geometric information related to each channel. The trigger map file selects which trigger regions combinations are allowed for multi-event grouping and coincidence event filtering.

3.2.1 Channel map table contents

The channel map table is a tab separated table text file, with the following columns:

- portID: The DAQ port ID. For an evaluation kit with Ethernet connection (PAB or FEB/D) this is always 0.
- slaveID: The slave ID when connecting multiple FEB/Ds in daisy chain. For an evaluation kit with Ethernet connection (PAB or FEB/D) this is always 0.
- chipID: The ASIC ID in respect to the position where it is connected in the FEB/D (or PAB). Its value is 0 if the Test board is connected in port F1 of the FEB/D (or connector J15 of the PAB), 2 if the Test board is connected in port F2 of the FEB/D (or connector J16 of the PAB). If the test board is connected in ports F3, F4, F5, F6, F7 or F8 of the FEB/D, the value for chipID is 4, 6, 8, 10, 12 or 14, respectively.
- channelID: The channel ID inside the ASIC, from 0 to 63.
- regionID: Trigger region number (eg, 0, 1, 2...).

- xi: Pixel index number in the X direction (0, 1, 2, ...).
- yi: Pixel index number in the Y direction (0, 1, 2, ...).
- x: Pixel position in the X direction (in mm).
- y: Pixel position in the Y direction (in mm).
- z: Pixel position in the Z direction (in mm).

The trigger region ID number define the different possible channel regions for selection of grouped and coincidence events: the software will only consider as coincidence events those belonging to different trigger regions and selected by the trigger map (as described below). PETSYS Electronics provides two maps ("tables/febd_tofpet2tb_fake_channel_map.tsv" and "tables/pab_tofpet2tb_fake_channel_map.tsv") which should be used according to the hardware where the TOFPET2 ASIC Test board are connected (FEB/D or PAB). For the case of the PAB, only two trigger regions are defined, one for each test board in the two available connectors (J15 and J16). For the case of the FEB/D, test boards connected in two contiguous FEB/D ports (F1 and F2, F3 and F4, etc) will belong to the same trigger regions. Is is thus IMPORTANT to connect both test boards in F1 and F3, for example (F2 and F4, F5 and F7 and F6 and F8 are also possible), in order to obtain coincidence events with the software.

The pixel index numbers in the X and Y direction are simply tagged to the event data when writing some output files for analysis. Relevant if the user wants to construct a flood map with multi-channel (multi-pixel) data. The pixel position values are used by the software when considering events which trigger multiple pixels and can also be written to ROOT for analysis (PET image reconstruction, for example).

3.2.2 Triger map table contents

The trigger map table is a table with three columns, where the first 2 collumns define a trigger region pair and the purpose of this pauir for software postprocessing: either coincidence (C) or Multi-event Group (M). Again, two tables are provided by PETSYS Electronics, one for each version of ekit (PAB or FEB/D).

Chapter 4

Data acquisition

The acquisition applications consist, mostly, of a series of Python scripts and libraries which implement 3 main functions.

- Configure the hardware.
- Acquire raw data.
- Store raw data into files.

4.1 Data acquisition template

The data acquisition is performed by a python, which is shown in listing 4.1. This Python program performs the following basic tasks (after parsing the arguments, which ends at line 14):

- Read and store configuration and calibration data from the files defined in the reference configuration file
- Connect to “daqd” and initialize the hardware.
- Load the hardware configurations and define operational bias voltage (from config file)
- Modify the ASIC configuration as needed (in this case setting the two configuration options related to TOT or QDC).
- Upload the ASIC configuration into the hardware.
- Open the output raw data file for writing.
- Acquire data during a period of time.
- Set the HV bias voltage to 0 on all channels and exit.

Listing 4.1: acquire_sipm_data.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  from petsys import daqd, config
5  from copy import deepcopy
6  import argparse
7
8  parser = argparse.ArgumentParser(description='Acquire data for TDC calibration')
9  parser.add_argument("--config", type=str, required=True, help="Configuration file")
10 parser.add_argument("-o", type=str, dest="fileNamePrefix", required=True, help="Data filename 
       (prefix)")
11 parser.add_argument("--time", type=float, required=True, help="Acquisition time (in seconds)")
12 parser.add_argument("--mode", type=str, required=True, choices=["tot", "qdc"], help="
       Acquisition mode (ToT or QDC)")
13 parser.add_argument("--enable-hw-trigger", dest="hwTrigger", action="store_true", help="Enable
        the hardware coincidence filter")
```

```
args = parser.parse_args()

systemConfig = config.ConfigFromFile(args.config)

daqd = daqd.Connection()
daqd.initializeSystem()
systemConfig.loadToHardware(daqd, bias_enable=config.APPLY_BIAS_ON, hw_trigger_enable=args.
    hwTrigger)

qdcMode = args.mode == "qdc"

asicsConfig = daqd.getAsicsConfig()
for ac in asicsConfig.values():
        gc = ac.globalConfig
        for cc in ac.channelConfig:
                cc.setValue("trigger_mode_1", 0b00)

                if not qdcMode:
                        cc.setValue("qdc_mode", 0)
                        cc.setValue("intg_en", 0)
                        cc.setValue("intg_signal_en", 0)
                else:
                        cc.setValue("qdc_mode", 1)
                        cc.setValue("intg_en", 1)
                        cc.setValue("intg_signal_en", 1)


daqd.setAsicsConfig(asicsConfig)

daqd.openRawAcquisition(args.fileNamePrefix, qdcMode=qdcMode)
daqd.acquire(args.time, 0, 0);

systemConfig.loadToHardware(daqd, bias_enable=config.APPLY_BIAS_OFF)
```

Before acquiring data, the user needs to define the threshold values for each of ASIC's discriminators. These values are defined in a table which is referenced in the "disc_settings_table" field of the reference configuration file. For most SIPMs tested by PETSYS Electronics in the default triggering mode, we recommend a threshold of 20 ADCs above the baseline value for t1 discriminator, the same for t2, and a value of 15 ADCs above the baseline for the E discriminator. The user can create a table that sets the same threshold value for all the connected channels by running the "make_simple_disc_table" application. For the example of the recommended settings, this can be done by typing:

```
./make_simple_disc_table --config config_validation_2.ini --vth_t1 20 --vth_t2
20 --vth_e 15 > disc_settings.tsv
```

The above command will write the output into the table file named "disc_settings.tsv" to be referenced in the configuration file. The user is free to use other values for the thresholds, taking into account that the values in the table correspond to ADCs above the baseline and the DACs are 6-bit, ranging from 0 to 64.

Once the threshold table, SIPM bias voltage table and calibration files are defined and referenced in the configuration file, the user is ready to acquire data from the SIPMs and ASICs. This acquisition script is run as follows:

```
./acquire_sipm_data --config config.ini --mode qdc --time 60          -o
data_folder/data_prefix
```

Each argument required (or optional) is preceded by a corresponding flag. The user may see all flags and argument descriptions by typing "–help" anywhere after "./acquire_sipm_data".

In the above data acquisition example, the first flag is "–config" which is used to specify the reference configuration file described in chapter 3. Ohter required flag is "–mode", which can only take two values: "qdc" or "tot". In "qcd" mode, the output energy measurement for each pulse is obtained by linear charge integration. PETSYS Electronics recommends to use this mode (better resolution), except for very large signals that could saturate the integrator. If using "tot" mode, the energy measurement is obtained from Time-over-threshold, using the E and t1 discriminators time stamps (non-linear measurement).

The "–time" flag sets the amount of time to acquire data, 60 seconds in the above example.

The "-o" flag sets the output data file prefix. In this example, the script will produce 2 files named "data_prefix.rawf" and "data_prefix.idxf" and placed inside the folder "data_folder". The file "data_prefix.rawf" contains the raw data in compact binary format. Its exact structure is not documented in this guide since the time information obtained directly from the ASICs, as well as the energy measurement from the integrator require correction using calibration on the post processing level, as previously described. The "data_prefix.idxf" file is an auxiliary text file containing information on the range of events for each of the variables "step1" and "step2", which is only relevant if doing parameter scans as described below.

4.1.1 Hardware Coincidence Trigger

Although not used in the above example, the acquisition script has the optional flag "–enable-hw-trigger"[1]. If this flag is used (no other argument required), the hardware (FPGA) will filter events based on their time of arrival and channel ID (using the same trigger regions defined in the template trigger maps described above). The parameters defining the trigger can be set in the "[hw_trigger]" section of the Reference Configuration File and are:

- "type" - The type of hardware filter to be implemented. At the present moment, only the option "builtin" is available.
- "threshold" - A raw energy threshold, which only works if the acquisition is in QDC mode (in ADC units, before calibration).
- "pre_window" - If a coincidence event is found, this variable sets the pre-event time window of accepted events, in clock period units (from 0 to 15).
- "post_window" - If a coincidence event is found, this variable sets the post-event time window of accepted events, in clock period units (from 0 to 127).
- "coincidence_window" - This variable sets the coincidence time window, in clock period units (from 0 to 3). If two events in different trigger regions have a time stamp difference less than
- "single_fraction" - Single event sampling switch, adding up to the coincidence filtered data. If set to 0, the single event sampling is disabled. If set to 1, data from 10 clocks period in each 1025 clock period is accepted. If set to 2, data from 20 clocks period in each 1025 clock period is accepted. If set to 3, data from 50 clocks period in each 1025 clock period is accepted. If set to 4, data from 100 clocks period in each 1025 clock period is accepted.

4.2 Modifying the acquistion script to scan parameters

With PETSYS Electronics software, it is possible to perform parameter scans whitout having to acquiring data in multiple times and into mukltipe files. This can be done by modifying the base acquisition script (lst 4.1) and can be useful for data analysis and testing. The example in listing 4.2 shows how to scan 4 values for HV bias voltage and 8 values for the time discriminator threshold, in one acquision take. This block of code should replace line 43 of listing 4.1 and the user can save the resulting file with a different name.

In this example, there are two loops for two variables ("step1" and "step2"). In the first loop, the overvoltage for each HV DAC channel is assigned to the "step1" variable, which takes the values 3.5V, 4V and 4.5V.

For each value of HV bias voltage, there will be a second loop over variable "step2". This will be assigned so that the time discriminator threshold will be successively set to 0, 3, 6, 9, ..., 18 ADC **above**[2] above the amplifier output baseline.

For each pair of bias voltage and time threshold, the configuration is uploaded and the acquisition is performed for the duration of acquisition time selected.

[1]At the moment, this feature is not implemented for evaluation kits using a PAB. It will be implemented in future firmware releases

[2]The 6-bit DACs in this ASIC are all in reverse scale, 0 is maximum and 63 is minimum, hence the "c.zero_t1-step2"

Listing 4.2: 2 parameter scan.

```
for step1 in [3.5, 4, 4.5]:
        for e in systemConfig.sipmBiasTable.values():
                e.Vover = step1
        for step2 in range(0,19,3):
                cfg = deepcopy(systemConfig)
                for channelKey in cfg.discCalibrationTable.keys():

                        p,s,a,c = channelKey
                        #if a != 4: continue # Change threshold only for
                            ASIC 4

                        c = cfg.discCalibrationTable[channelKey]
                        s = cfg.discConfigTable[channelKey]
                        s.vth_t1 = c.zero_t1 - step2


                cfg.loadToHardware(daqd, config.APPLY_BIAS_ON)
                daqd.acquire(args.time, step1, step2);
```

Chapter 5

Data processing

It was mentioned above that the raw data written to disk with the data acquisition scripts does not have a correct detection time in seconds. Such data only contains the measured values from the ASIC TDCs in ADC units. The conversion from these values into time does not follow a universal function (although it is very close to a linear relation) and needs to be calibrated for each ASIC (see sec. 3.1.2). Also, if the acquisition was performed in “QDC” mode, a postprocessing correction of the raw data delivered by the ASIC is also needed (see sec. 3.1.3).

In order to use the calibration files and convert the raw data into data with correct time information, one needs to use the processing applications provided by PETsys Electronics.

5.1 Singles application

The main processing application is “convert_raw_to_singles”. By default, this program provides a list of all registered events using minimal filtering (only excludes events where the measured time in ADCs is outside of the operational nominal range), writing as output a text file where each line corresponds to one event and each column to the following variables:

- Time of detection in picoseconds
- Energy of the pulse (in QDC mode, in ADC units; in TOT mode, in nanoseconds).
- Absolute channel ID (see appendix A)

In order to process the data (follwing the example from chapter 4), just type:

```
./convert_raw_to_singles --config config.ini -i data_folder/data_prefix
-o data_folder/data_singles.txt
```

In the above example, “data_folder/data_prefix” is the input (flag “-i”) raw data file prefix and “data_folder/data_singles.txt” is the output (flag “-o”) file.

As an example of the output of this application, “data_singles.txt” in this example:

```
...
572439987088 19.973000 11
572440187287 -3.184000 24
572440440209 -3.231000 22
572442091600 -3.200000 108
572442474458 70.130997 168
572442472480 166.860001 172
572442475789 117.203003 99
572442472716 197.824005 115
572442476890 68.641998 109
572442477724 34.335999 103
```

```
572442475965 101.133003 107
572442474599 102.626999 127
572442475982 110.903999 119
572442474514 133.449997 113
572442483501 90.010002 11
572442493718 13.785000 7
...
```

When the amount of data is very large, the text output file requires a large disk storage space. The application has the option to write into a more compact binary format using the "–writeBinary" flag. If this option is set, the same three variables per event will be written to the output file in a format of 64 bit integer (time in picoseconds), 32 bit floating point (energy or TOT, depending on the acquision mode) and 32 bit integer (channel ID). The C code structure for each event is:

```c
struct Event {
        long long time;
        float e;
        int id;
};
```

If using the binary output option, the output file defined after the "-o" flag is indeed a prefix, since two files will be created. In this way, the user should not define an extension for the file , and can run the application as:

```
./convert_raw_to_singles --config config.ini -i data_folder/data_prefix
 -o data_folder/data_singles --writeBinary
```

There will be two files created in "data_folder": a "data_singles.ldat" containing the binary data with the structure presented above and a "data_singles.lidx" which contains information of the event range corresponding to diferent parameters acquired during a parameter scan as described in section 4.2. If the data acquisition did not involve a parameter scan, then this file will not be relevant but will still be created (just one line).

A third output data type is possible with the application "convert_raw_to_singles", consisting of a ROOT file with the data in the form of a ROOT TTree named "data" with one event per entry. This TTree will contain some more information useful besides the three entries described above. The full TTree stucture is:

- "step1" and "step2" are the values of step1 and step2 from the acquisition with multiple paraemters. If no scan is performed (default data acquisition script), both this variables are set to 0.
- "time" is the time of the event, in picoseconds.
- "channelID" is the absolute channel ID of the event (see appendix A).
- "tac" is the TAC number for this event, ranging from 0 to 3 [1].
- "tot" is the Time-over-Threshold for the event, in nanoseconds. In QDC acquisition mode, this time corresponds to the time of integration, while in TOT mode it is the difference between the time stamps at the E and t1 discriminators.
- "energy" is the energy of the event (in QDC mode, in ADC units; in TOT mode, in nanoseconds).
- "channelIdleTime" is the time between the event and the previous event in the same channel.
- "tacIdleTime" is the time between the event and the previous event in the same TAC.
- "xi" and "yi" are the index position in the X and Y directions of the pixel connected to channelID (obtained from the channel map file described in 3.2.1).
- "x", "y" and "z" are the physical coordinates of the pixel connected to channelID (obtained from the channel map file described in 3.2.1).

[1] described in page 4 of the TOFPET ASIC v1 - Short Data Sheet

- "tqT" is the fine time (time for crossing the time threshold) measured directly by the TDC in units of clock periods.
- "tqE" is the fine time (time for crossing the energy threshold) measured directly by the TDC in units of clock periods.

For the ROOT output, use the option "–writeRoot", so the applicatation should be run as:

```
./convert_raw_to_singles --config config.ini -i data_folder/data_prefix
 -o data_folder/data_singles.root --writeRoot
```

Option "–help" prints to screen all the arguments and flags for "convert_raw_to_singles".

5.2 Coincidence application

With the single event data, the user can create data analysis routines and extract any useful information. Nevertheless, PETsys Electronics provides an application ("convert_raw_to_coincidence") designed to filter and group the events based on geometrical and timing criteria, identifying and writting only coincidence event data, which can be useful for PET related applications. The application will group single events that occur in a given geometrical region (only relevent if using SIPM arrays), select the highest energy event in each group and identify two of these events as being in coincidence if they are detected with a time difference less than 20 ns (by default) and occur in different trigger regions. The trigger regions are defined in the channel map described above and selected through a trigger table map.
In order to use the coincidence application, just type:

```
./convert_raw_to_coincidence --config config.ini -i data_folder/data_prefix
    -o data_folder/data_coincidences.txt
```

The output file will be again by default a text file. Each line corresponds to a coincidence event. As described above, one coincidence event consists of two single events, occuring in different trigger regions. Besides the information delivered by the singles application, two more integer numbers are provided for each of the two events in coincidence. The ten columns correspond to the following variables:

- "mh_n1", the number of events that belong to the same group of this particular event (for the event in the trigger region of higher ID number). If "mh_n" is 1, then no other events arrived in a region of 20mm within a 100 ns time window.
- "mh_j1", the number ID (from 0 to "mh_n"-1) of this event in the group it belongs to (ordered from higher to lower energy, for the event in the trigger region of higher ID number). For example, one event with "mh_j" set to 1 means this event has the second highest energy in the group it belongs to. By default this number is allways 0, unless the option "–writeMultipleHits" is selected.
- Time of detection in picoseconds (for the event in the trigger region of higher ID number).
- Energy of the pulse, for the event in the trigger region of higher ID number (in QDC mode, in ADC units; in TOT mode, in nanoseconds).
- Absolute channel ID, for the event in the trigger region of higher ID number (see appendix A).
- "mh_n2", the number of events that belong to the same group of this particular event (for the event in the trigger region of lower ID number). If "mh_n" is 1, then no other events arrived in a region of 20mm within a 100 ns time window.
- "mh_j2", the number ID (from 0 to "mh_n"-1) of this event in the group it belongs to (ordered from higher to lower energy, for the event in the trigger region of lower ID number). For example, one event with "mh_j" set to 1 means this event has the second highest energy in the group it belongs to. By default this number is allways 0, unless the option "–writeMultipleHits" is selected.

- Time of detection in picoseconds (for the event in the trigger region of lower ID number).
- Energy of the pulse, for the event in the trigger region of lower ID number (in QDC mode, in ADC units; in TOT mode, in nanoseconds).
- Absolute channel ID, for the event in the trigger region of lower ID number (see appendix A).

An example of the output data stored in “data_folder/data_coincidences.txt” (by default, writing only the more energetic event in each group - mh_j=0):

```
...
11 0 301480472837 185.294998 199 5 0 301480465259 167.621994 13
12 0 301532891318 226.013000 196 3 0 301532893866 152.274002 15
15 0 301738429998 221.503998 193 4 0 301738430771 164.977997 10
4 0 301738430052 289.398987 206 4 0 301738430771 164.977997 10
3 0 301918815624 176.210999 216 7 0 301918814401 208.921997 14
10 0 301931404962 215.347000 180 4 0 301931405908 171.488007 9
11 0 301942022635 217.341995 179 6 0 301942023468 172.139008 10
3 0 301996556877 176.731995 171 3 0 301996556753 167.464996 3
12 0 302024405940 223.718994 181 8 0 302024405463 205.020004 6
10 0 302130348655 211.975006 206 6 0 302130349774 199.153000 1
14 0 302153241603 221.628998 200 3 0 302153243423 158.725006 11
8 0 302176578134 236.889008 163 5 0 302176579154 216.787003 15
7 0 302239978904 200.735992 184 7 0 302239980521 194.102005 7
13 0 302316883175 237.466003 202 7 0 302316883190 212.334000 1
3 0 302338280882 186.028000 182 7 0 302338280899 199.207993 11
7 0 302395147844 179.563995 202 9 0 302395147466 235.246002 5
13 0 302408236333 202.229996 200 6 0 302408238547 201.173996 7
6 0 302437555834 209.151001 210 10 0 302437555777 271.506989 7
14 0 302465167662 202.238007 182 7 0 302465166982 209.720001 6
10 0 302490604579 209.537994 192 8 0 302490605299 225.447006 7
9 0 302536237352 195.199005 195 7 0 302536237768 230.324997 2
9 0 302557333164 212.367004 185 9 0 302557333902 173.858002 5
11 0 302698969034 217.651001 195 5 0 302698969442 197.901993 12
10 0 302699934611 220.119003 222 8 0 302699935309 222.557007 1
10 0 302745318600 227.632996 207 7 0 302745318053 227.115005 12
9 0 302757036740 227.098999 196 5 0 302757038323 229.815994 15
8 0 302774357728 178.496994 191 5 0 302774355897 201.186996 6
14 0 302778056665 170.263000 199 9 0 302778055246 214.335007 14
16 0 302804616432 210.240005 172 10 0 302804616836 213.227005 9
8 0 302811200959 210.856995 202 8 0 302811200378 198.110001 14
7 0 302814200700 179.029999 203 4 0 302814200615 220.423996 0
10 0 302928019824 225.815994 183 5 0 302928018938 235.091995 8
7 0 302970035403 192.188004 221 6 0 302970035107 189.781006 10
6 0 302970034195 201.929993 222 6 0 302970035107 189.781006 10
15 0 302981848884 212.018997 189 11 0 302981848937 180.059998 6
15 0 302985861610 224.072006 200 6 0 302985864050 194.565002 7
...
```

Again, using the option “–writeBinary”, the output file will be a compact binary with 10 variables per event and the data format will be: 64 bit integer (time in picoseconds), 32 bit floating point (ToT or Energy) and 32 bit integer (channel ID), 64 bit integer (time in picoseconds), 32 bit floating point (ToT or Energy) and 32 bit integer (channel ID). The full C code structure for each event is:

```c
struct Event {
        uint8_t mh_n1;
        uint8_t mh_j1;
        long long time1;
        float e1;
        int id1;
        uint8_t mh_n2;
        uint8_t mh_j2;
        long long time2;
        float e2;
        int id2;
};
```

Also, a third output data type is possible with the application “convert_raw_to_coincidence”, consisting of a ROOT file with the data in the form of a ROOT TTree named “data” with one coincidence event per entry. There will be two sets of “time”, “channelID”, “tac”, “tot”, “energy”, “mh_n”, “mh_j”, “xi”, “yi”, “x”, “y”,“z”, one for each event in a coincidence, as branches of the TTree “data”.

Several other command line options are available to control the filtering of coincidence events during processing and when writing to output file. Option “–help” prints to screen all the arguments and flags for “convert_raw_to_coincidence”.

5.3 Raw to ROOT application

It was mentioned above that the raw data (time and energy) originated from the ASIC and FPGA needs to be corrected with calibration functions, hence the need for the processing aplications. Nevertheless, as a debuging way to check if there is some problem in the raw data, the user may want to check this data before the application of any calibraiotn. To do so, and since the raw data is in binary format, PETSYS Electronics also provides an application that converts the raw data into a ROOT TTree format.

In order to use it, just type:

```
./convert_raw_to_root --config config.ini -i data_folder/data_prefix
-o data_folder/data_raw.root
```

The full TTtree structure on th output data is:

- “step1” and “step2” are the values of step1 and step2 from the acquisition with multiple parameters. If no scan is performed (default data acquisition script), both this variables are set to 0.
- “channelID” is the absolute channel ID of the event (see appendix A).
- “tacID” is the TAC number for this event, ranging from 0 to 3 [2].
- “tcoarse” is the coarse time stamp measured at the t1 discriminator, in clock periods, from 0 to 1023.
- “tfine” is the fine time stamp measured by the TDC at the t1 discriminator, in ADC units, typically ranging from 200 to about 400.
- “ecoarse”, in TOT acquisition mode, is the coarse time stamp measured at the E discriminator, in clock periods, from 0 to 1023; in QDC acquisition mode, is the coarse time stamp corresponding to the charge integration ending, in clock periods, from 0 to 1023.
- “efine” in TOT acquisition mode, is the fine time stamp measured by the TDC at the E discriminator, in ADC units, typically ranging from 200 to about 400; in QDC acquisition mode, is the charge intergration value, in ADC values.
- “frameID” identifies the frame of data to each the event belongs. Each frame contains 1024 clock periods.

[2]described in page 4 of the TOFPET ASIC v1 - Short Data Sheet

Appendix A

Absolute ID of Channels, ASICs and HV DAC channels

The software and firmware assign an absolute ID to each ASIC, channel, and HV DAC channel. This ID numbering is related directly to the PAB connector (or FEB/D port) where a given TOFPET2 ASIC Test board is plugged.

The absolute IDs are constructed in the following way:

$$\text{Absolute Channel ID} = 64 \times \text{chipID} + \text{ChannelID}$$

where:

- chipID: The ASIC ID in respect to the position where it is connected in the FEB/D (or PAB). Its value is 0 for a TOFPET2 ASIC Test board connected in port F1 of the FEB/D (or connector J15 of the PAB), 2 if the Test board is connected in port F2 of the FEB/D (or connector J16 of the PAB). If the test board is connected in ports F3, F4, F5, F6, F7 or F8 of the FEB/D, the value for chipID is 4, 6, 8, 10, 12 or 14, respectively.
- channelID: The channel ID inside each ASIC, from 0 to 63.

The above formula implies the following:

Test board in port F1 of FEB/D (or J15 in the PAB) $\Rightarrow$ chipID=0 $\Rightarrow$ Absolute channel ID from 0 to 63
Test board in port F2 of FEB/D (or J16 in the PAB) $\Rightarrow$ chipID=2 $\Rightarrow$ Absolute channel ID from 128 to 192
Test board in port F3 of FEB/D $\Rightarrow$ chipID=4 $\Rightarrow$ Absolute channel ID from 256 to 320
Test board in port F4 of FEB/D $\Rightarrow$ chipID=6 $\Rightarrow$ Absolute channel ID from 384 to 448
Test board in port F5 of FEB/D $\Rightarrow$ chipID=8 $\Rightarrow$ Absolute channel ID from 512 to 576
Test board in port F6 of FEB/D $\Rightarrow$ chipID=10 $\Rightarrow$ Absolute channel ID from 640 to 704
Test board in port F7 of FEB/D $\Rightarrow$ chipID=12 $\Rightarrow$ Absolute channel ID from 768 to 832
Test board in port F8 of FEB/D $\Rightarrow$ chipID=14 $\Rightarrow$ Absolute channel ID from 896 to 960

Regarding the absolute HVDAC Channel ID, the user may refer to the mapping tables provided in the Client Package Folder for PAB (32 channels) and FEB/D (64 channels).