Manual for GRANDproto35 operation

Olivier Martineau-Huynh[a], Jacques David[a], Junhua Gu[b], David Martin[a], Patrick Nayman[a], Vincent Voisin[a]

[a] *LPNHE, Université Pierre et Marie Curie, Université Paris Diderot, CNRS/IN2P3, Paris, France.*

[b] *National Astronomical Observatories of China, Chinese Academy of Science, Beijing 100012, P.R. China.*

1 Introduction

In this document we give details needed to set up, run and maintain the GRAND-proto35 acquisition, and (tentatively) fix possible issues.

2 The GRANDProto35 detection unit

Here we give a brief description of the structure of the GRANDProto35 detection unit and provide informations needed to make sure it runs in a proper state and identify possible problems when encountered.

2.1 Antenna

To be done

2.2 Electronic Box Set-Up

The electronic Box (EB) should be fixed to its antenna pole as shown in Fig. 1: the front pannel is vertical, with the pressure valve facing ground. The box may be fixed inside the lower vertical arm of the antenna.

2.3 Connections

Connections to the EB should be carried out in the following order:

i) First connect the three outputs of the antenna to the associated inputs *ANT X*, *ANT Y*, *ANT Z* of the EB with the three dedicated 75Ω coaxial cables. Make sure here that you connect the 75Ω N-type connector end (the thinner one), and not the 50Ω one (thicker). Inversion would destroy the plug. Also note that there is a 12 V DC level on these cables, which is used to power the antenna LNAs (see section 2.4.1 for details). It is therefore safer to plug/unplugg them only when the EB power is off, in order to avoid possible short-cuts.

Email address: `martineau@lpnhe.in2p3.fr` (Junhua Gu)

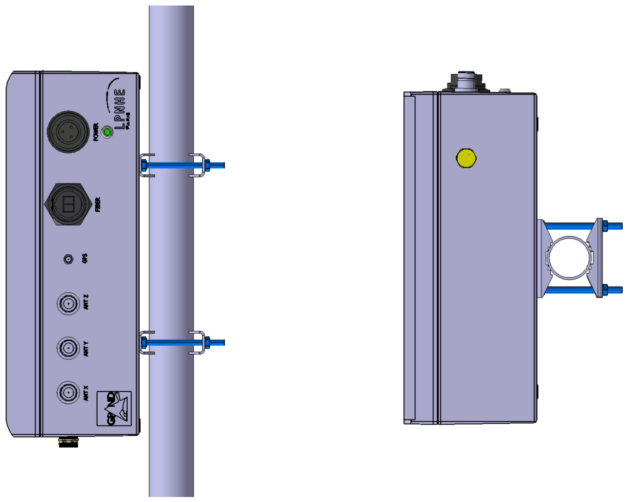
Figure 1: Front (left) and bottom (right) views of a GRANDProto35 Electronic Box casing once fixed to an antenna pole.

ii) Then connect the GPS antenna. By no means should the GPS antenna be connected when the power is on. This may destroy the GPS unit inside the board (see section 2.4.2) and make it unusable. The GPS antenna should preferably be placed above ground. A magnet inside the GPS antenna allows to fix it to a close-by metalic surface.

iii) Then connect the optical fiber (which can however be plugged/unplugged at any time, with no risk of damage).

iv) Finally plug the power cable. Nominal value of the power supply should be 15 V, but values between 9 and 20 V are still OK. A green LED on the front panel just below the power plug allows to check if the board is powered on.

Cables between the box and the antenna should be tied to the pole to avoid dangling and stress on the connectors.

2.4 Electronic Board description

A picture of the board is shown in figure 2. It can be divided in 4 functionnal sections: analog, digitial, communication and power supply.

2.4.1 Analog part

The description of the analog part of the board is given in details in [1] (in French). Here we just give a brief overview.
The analog part lies in the top-left corner of the board. Signal input to the board is done for each of the three channels through an MMCX 75Ω jack connected to the corresponding N-type connector. From top to bottom we have X, Y and Z channels. The signal is adapted to a 50Ω impedance thanks to a dedicated amplifier (G=4) placed just after the connector, then fed into the 30-100 MHz filters tagged *KR ELECTRONICS*. Signal is finally processed through a power detector AD8310 [2], which acts as an envelop detector. Note however that the poer detector also performs a logarithmic

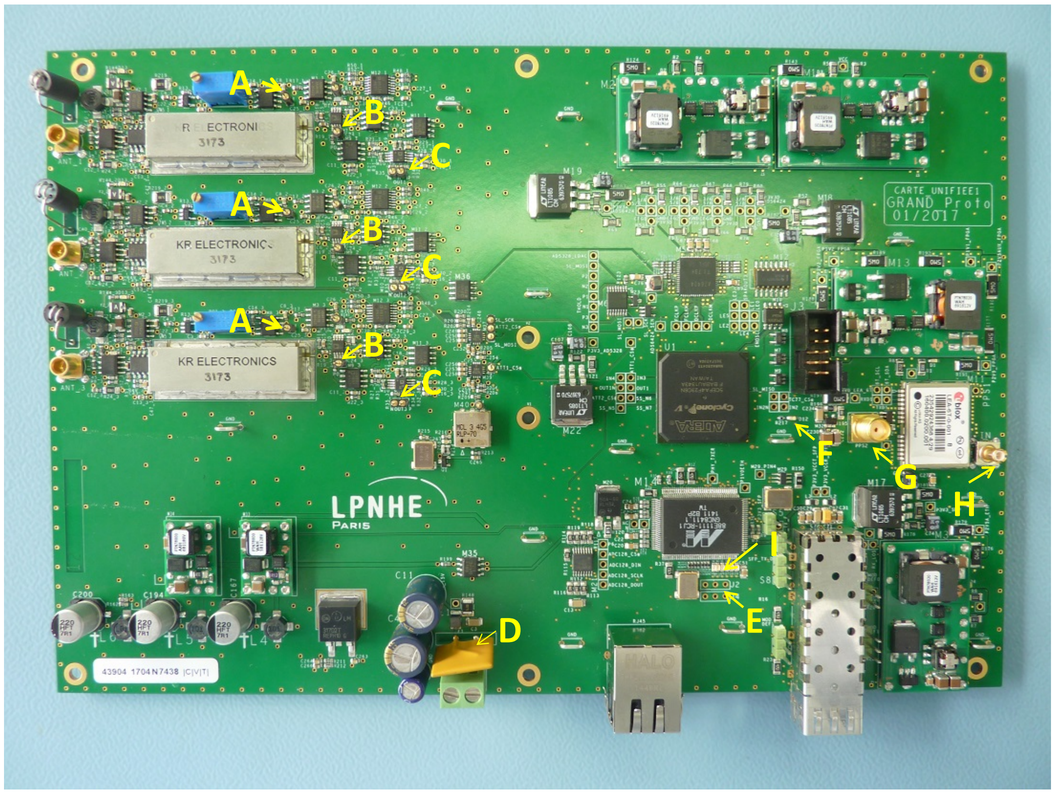

Figure 2: Picture of the GRANDProto35 Electronic Board. Meaning of the letter tags: "A" stands for the measurement point for the the power dectector common-mode voltage level. The screw on the blue) potentiometer can be used to adjust the voltage value. "B": measurement point for the signal at output of the power detector. "C": differential measurement points for the signal at input of the ADC. "D": re-amarable fuse for board power input (see text for details). "E": jumper to switch between ethernet plug (left position) or optical (right position). "F": FPGA configuration flag, should lit in red when FPGA is properly configured. "G": GPS PPS signal output: should deliver a periodic pulse (T=1 s) when GPS is properly set. "H": connector to the GPS antenna. "I": light signals for communication with board. **Specify here normal beahaviour of lights.**

amplification of the signal, which allows for an increased dynamic range **(be more quantitative here)**, but also distords the shape of the signal (smaller amplification for larger signals). The power detector runs in differential mode. Its common mode voltage has a nominal value of 0.9 V adjusted thanks to a potentiometer (blue square, just above the filter). The common-mode level can be measured between ground and the test point shown on Fig. 2 by the letter tag "A".
The analog section also provides the power supply to the LNAs placed inside the antenna nut through a bias-tee system implemnted in the analog section. This means that in normal operation, a 12 V DC voltage is applied on the input plugs, which the user should be aware of when performing tests on a powered board. Note however that the LNAs power supply can be switched on and off by the user using the remote command program (see section 3.4.2).

In addition to this, the analog part includes a trigger section, where the signals at the output of the filters are first amplified by an additionnal factor 10 thus bringing the total amplification for the trigger channel to $4 \times 10 = 40$. The signal level is then compared to threshold values set by the user through the remote control program. A trigger flag is generated and sent to the FPGA (see section 2.4.2) if one channel exceeds it. Note that there are six independant trigger channels, corresponding to two polarities for each of the three channels. The user can activate/inhibit independantly each of these trigger channels and set their respective threshold values, again through the remote control program (see section 3.4.3).
Finaly an internal calibration system is also included in this analog part: its core is a 66.6660 MHz quartz oscillator (placed at the center of the board), which generates a sine wave. Its amplitude can be moderated thanks to two attenuators, with attenuation values adjustable by the user (see section 3.4.2). When the user uses the DAQ in calibration mode, the input of the signal treatment chain detailed above is switched from the MMCX connector input to this calibration signal. This allows to calibrate the full DAQ chain.

2.4.2 Digital part

The digital treatment of the signal is detailed in [3]. It is performed on the right part of the card, starting with a 4-channels ADC [4] running at a nominal frequency of 50 MHz, ajustable up to 100 MHz through the FPGA firmware. The ADC continously digitizes the signal coming from the three analog channels on 12 bits and within a dynamic range [-1 V,+1 V]. The 4^{th} ADC channel is used in calibration mode to digitize the signal at the output of the quartz oscillator. It delivers dummy data in other run modes. The digital signal of each of the four channels is buffered in a circular register inside the FPGA. When one trigger is received (see previous section), a subset of length $2 \times OFFSET$ (where $OFFSET$ is a parameter set by the user at run start) is saved for each of the four channels to form one event, with the trigger position being usually few samples after the middle position (index 105 for 180 samples). A GPS time-tag is also build by the FPGA (see section 2.4.3) and embeded in the event header and sent to the board output (see section 2.4.4).

2.4.3 Timing

A resolution of the order of 10 ns on the antenna trigger time is requested in order to achieve a $\sim 1^\circ$ resolution on the reconstructed direction of origin of the radio wave. The trigger time tagging in GRANDProto35 is done in the following way:

- the GPS unit delivers a pulse-per-second (PPS) signal to the FPGA. If the position of the GPS unit is known with good precison (<1 m), a resolution of the order of 10 ns is achievable on the PPS period (see [5]). However the EB design only allows for a very basic communication with the GPS receiver, and the information on its true position cannot be provided to the GPS (note that similarly, no information other than the PPS signal can be retrieved from the GPS). The position of the GPS unit is therefore taken as its averaged value, provided the rms of its measurement do not exeed 10 m. In case this condition is not reached, then no PPS signal is emitted. So far however the timing info has always been present in the data after a few minutes, meaning that the PPS signal was indeed emitted. The actual timing precision achieved through this method is still to be determined.

- A counter inside the FPGA, running at a nominal 125 MHz frequency, then provides a 8-ns granularity on the trigger time. The counter is reset at each new PPS signal. The counter value is a 32-bits integer named $TS2$, available in the header of the event data sent to the DAQ (see section 3.2). Note however that the counter frequency is not stable, and depends in particular on its temperature. In order to get a reliable time tag from the counter, the actual number of clock counts achieved between two PPS signals is recorded by the FPGA every second. This information is available in the slow control data (see section 2.4.6), under the name *MAXCOARSE*.
- Additionnaly two more fields ($TS1PPS$ and $TS1Trigger$) provide a nanosecond granularity of the time tag.

Eventually, the absolute time tag for a trigger can be computed as :

$$t = t_0 + \left(TS2 + \frac{TS1PPS - TS1Trigger}{4}\right)\frac{1}{MaxCoarse}$$

where t_0 is the UTC second in which the trigger was received and the second term of the addition corresponds to the time elapsed between the beggining of that specific UTC second and the trigger. In order to compute t_0, the SSS info is is used. Its raw value corresponds to the number of seconds since the board received its first PPS signal. As their GPS units may not start issuing the PPS signal at the same instant, there may be offsets between SSS values for different EBs. The SSS fields are therefore corrected online in the DAQ program, based on the DAQ unit GPS time info.

2.4.4 Communication

Communication with the DAQ computer is done through a Marvell interface chip, using the GEDEK Communication Engine. Messages are exchanged using UDP protocol. Either a standard ethernet plug can be used, or optical transfer through an SFP connector. A jumper placed just below the Marvell chip allows to switch from one to the other (left: ethernet plug, right: optical, see letter tag "E" in Fig. 2)). The latter is the standard communication system in GRANDProto35 operation, with a bi-modal SFP ($\lambda_1 = 1310\,\text{nm}$ & $\lambda_2 = 1550\,\text{nm}$) allowing downstream (data) and upstream (commands) communication on one single fiber.
The EB is identified through a unique MAC adress given by a specific chip (Dallas DS2502). At boot time, IP adresses of all units are identically set to 192.168.1.18 by default. A software program however allows to map each MAC adress to an IP adress of format 192.168.1.1xx (see section 4.1) where xx is the EB ID, ranging between 01 and 35. This IP adress is used for any further communication with the DAQ PC, whose IP should be 192.168.1.10 within a local network (submask 255.255.255.0).

2.4.5 Power supply

The nominal power supply of the EB is 15 V DC, but powers ranging between 9 and 30 V are acceptable. The standard power consumption is ∼10W in normal

run conditions. In standard operation at Ulastai, this DC voltage is provided by the external AC/DC converter installed at the pod level. There is also an internal AC/DC converter fixed on the internal side of the EB casing, but it is not used.
A re-armable fuse is installed at the input of the board power supply (letter tag "D" on Fig. 2). It will disrupt power supply in case a current surge is detected (**level ?**). It is re-armed after power cycling.
DC voltages of +4 V and -3.3 V are generated by DC-DC converters located on small mezanines placed at 3 corners of the board.

2.4.6 Slow control

Various slow-control parameters are measured on the board, and available for the user upon request (see section 3.3.5). In particular various key voltage values are monitored (master board voltage, +4 V, -3 V and LNA power suplies), as well as the temperature on the board.

2.4.7 Board control

There are several tools to check that the board is in a proper state and identify possible issues. **To be completed.**

3 Data acquisition & communication with board

3.1 General structure

The GRANDProto35 DAQ software structure is detailed in the README file and doc section of its dedicated gitHub repository [6]. Here we simply give a brief overview.

The DAQ software has a multi-layer structure. At the core of it is a very basic system of formated words exchanged between the EB and the DAQ PC. These are briefly described in section 3.4, and in more details in [3]. The central engine of the DAQ system is composed of two programs originally written in C, now translated in RUST, and running on the DAQ PC:

- `trend_server` runs on the DAQ PC and collects data sent by the Electronic Boards (EB) on a given port of the PC (by defaut 1235 for slow-control data, and 1236 for antenna data). This is the "ear" of the DAQ.
- `send_msg` is its "mouth". It issues commands to the EBs on another port (by default 1234).

By construction `trend_server` and `send_msg` do not communicate with each other. This is a built-in feature of the DAQ system, which directly derives from the fact that the DAQ is based on socket communication in UDP protocol. However in the RUST version of the program, an inner process makes sure that the acknowledge statement expected from the EB in response to a message from the DAQ PC is indeed received, and warns the user otherwise (`"NoAck"` message displayed on screen).
A third layer of code, which consists of shell scripts, is in charge of properly starting relevent processes, and sending appropriate commands in a timely manier so that

all runs smoothly. The standard user only accesses this layer in principle. **It is presently in a very preliminary form and should be improved**. These scripts are saved in the `gp_daq/scripts` directory, and some of them are detailed in section 4.

3.2 Data format

Data are in principle in binary format, following a structure very much inspired by the AERA data [7]. It can be accessed in a python session thanks to tools provided by a dedicated package called `pyef` [8]. Examples of data retrieval with `pyef` are given in the script `readData.py` [9] for example.
Pattern (see section 3.3.1) and slow control data is however saved in YAML format. This in particular allows the user to open it with a text editor and quickly check the normal beahavior of DAQ and units. Examples of YAML file handling in Python are given for example in the script `anaSLC.py` [9].
Note finaly that the standard data may be saved under YAML format as well, for debugging purpose. **Explain how this can be done**. The script `readDataYaml.py` [9] then allows to read these YAML data with python. Data directory is specified as an argument to the `trend_server` program. It is in principle */mnt/disk* and is associated with the $DATADIR$ variable in the DAQ machine.

3.3 Acquisition modes

The different types of data acquisition are the following:

3.3.1 Pattern mode

Here we request that the ADC generates a specific 12-bits pattern which can be:

1) **zeros**, a series of 8 null bits.
2) **ones**, a series of 8 "1" bits, corresponding to a numeric value of 4095.
3) **toggle**, corresponding to an alternance between sequences '010101010101' and '101010101010', equal to numeric values of 1365 and 2730 respectively.
4) **deskew**, corresponding to a sequence '110011001100', equal to a numeric value of 819.
5) **sync**, corresponding to a sequence '111111000000', equal to a numeric value of 4032.

A soft trigger is issued, resulting in the EB sending one event only. This mode allows to check that the digital section of the unit works properly. Pattern data is saved in YAML format, allowing to do this check by eye, simply opening the file with a text editor.

3.3.2 Calibration mode

Here the signal processed by the EB is the sine wave from the calibrator (see section 2.4.1), with a programable attenuation given by two coefficients Att1 and Att2, according to the following formula:

$$Att(dB) = (254 - (Att1 + Att2)) \times 0.25 \qquad (1)$$

Again a soft trigger is issued, but usually repeated several times in a single acquisition, resulting in a run composed of several events (typically 100). This mode allows to test that the analog part of the board works properly, and moreover to calibrate the response of the DAQ chain, as the amplitude of the input signal is known here.

3.3.3 Physics mode

This is the standard acquisition mode for cosmic-ray detection. Here the input of the EB is the antenna signals, and the acquisition is triggered by them, with ajustable threshold values (see section 3.4.3). The run will go on (and data will flow) as long as no other command is issued.

3.3.4 Minimal bias mode

In this mode, acquisition of the signals at antenna output is triggered by a soft trigger. This allows to record a baseline signal at a given instant, and is used for monitoring purposes. The command may be looped (200 times typically) in order to collect larger statistics.

3.3.5 Slow control acquisition

This mode allows to record the slow control information of the EB. It can be run in parallel to the others, as slow control is associated with a specific port (see section 3.1). In addition to the monitoring informations measured on the board for monitoring (voltage of the 3 LNAs, voltage at the unit input, absolute value of the +3V and -4V DC levels, temperature, see section 2.4.6 for details), the overall trigger rate of the unit, as well as the individual trigger rates of the six channels averaged over the last second are also sent in a SLC message, together with the channels thresholds values and the $MAXCOARSE$ value (see section 2.4.3).

3.4 DAQ messages

The DAQ is monitored through DAQ messages written in configuration files in YAML format, (thus readable through a standard text editor) and stored in the `gp_daq/cfgs` directory. When executed, the `send_msg` program reads the configuration file given as a parameter and sends the messages composing it to the EB. We detail below these messages and their different values for the various types of runs.

3.4.1 TRENDADC message

Each parameter of a *TRENDADC* message participates in the configuration of the ADC (see details in [4]). The message is the same for all acquisition modes except for pattern, where the word with adress 0xa (corresponding to the register 2 of the ADC) differs, with a value depending on the type of pattern chosen (see section 3.3.1), while it is 0 for the other modes.

3.4.2 TRENDDAQ message

The main parameters of the DAQ are defined in the *TRENDDAQ* word. They are:

- **DAQon**, a boolean which switches acquisition on and off.

- **AntOn**, a 3-bit-pattern switching on and off the output of channels X,Y and Z. It is set to 0 in calibration or pattern mode, and 7 (all channels on) in standard operation for the other modes.
- **EnOSC**, a boolean switching on and off the calibration oscillator. It is 1 in calibration mode, 0 otherwise.
- **Offst**, a scalar corresponding to half the trace length, in sample units. Default value is 90, corresponding to a 3.6 μs trace for a 50 MHz sampling rate.
- **EnablePD**, a 3-bit-pattern switching on and off the power detectors of channels X,Y and Z. Its default value is 7.
- **DisLNA**, a 3-bit-pattern switching off or on the LNAs of channels X,Y and Z. The default value for this parameter is 7 (all LNAs off). It is also set to this value in calibration or pattern mode, and 0 in standard operation for the other modes.
- **Attr1** and **Attr2** are the values for the 2 attenuators of the calibrator. Their value range between 0 (full attenuation) and 127 (no attenuation). These values do not matter for modes others than calibration.

Also note that the SSS info is reset each time a $TRENDDAQ$ command is issued.

3.4.3 TRENDTRIG message

In the *TRENDTRIG* message, we define the parameters relative to the trigger. Most important for the user are:

- **ST**, a boolean sending a soft trigger order to the EB.
- **TrgEn**, a 3-bit-pattern enabling the different channels. The order of the channels in the pattern is Z-, Y-, X-, Z+, Y+, X+, where the sign stands for the polarity of the trigger signal.
- **ThXs** (where X=1,2,3 is the channel, and s=+,- the polarity) is the threshold (in mV) of the 6 trigger channels at the comparator level (should be divided by a factor 40 to have an estimate at the antenna output level, and an additionnal factor 27 (24 dB) to detecrmine the corresponding voltage level at antenna input. Obviously these values are relevant only if the corresponding bits in the TrgEn pattern are set to 1.

4 Operation

Here we describe how to start and run the GRANDproto35 setup. This is a very preliminary structure, as the scripts are very likely to evolve. THis is therefore not too much detailed.

4.1 Start up

1- First set-up, connect and power the detection unit as described in section 2.

2- Then issue the command `./set_addr.sh` on the DAQ PC from the `gp_daq/scripts` directory. This command will call the subprocess `set_addr` located in directory `gp_daq/target/release`. This program will attribute a dedicated IP adress

192.168.1.1xx to each board of ID number xx. `set_addr` communicates with the board through its MAC adress, stored in the dedicated file `gp_daq/scripts/addr.yaml`. See [6] for details.
Success in establishing proper communication to an Electronic Board can be checked through a ping command to the unit IP adress. Note that the `set_addr` command also informs the units about the DAQ PC ID (IP adress and MAC adress), as well as the communications ports to be used for data and slow control data transfer.

3- Then the status of the unit can be requested through the following command:

```
./slcreq.sh BOARDID
```

from the same directory `gp_daq/scripts`. A slow-control event file should appear in the `$DATADIR` directory (in principal equal to `/mnt/disk`). Edit it and check that all voltage values are correct: vpower1 should be in range 9 to 24 V, vpower2 $\simeq$ 3 V, vpower3 $\simeq$ 4 V, vpower4, 5 and 6 = `min`(vpower1,12V). The temperature should also be in the range between 0 and 50°.
If all is OK at this stage, this means that DAQ communication with the board works fine and that the detection unit is properly powered.

4- Then launch:

```
./pattern.sh BOARDID 3
```

Note again that *BOARDID* is a two-digit integer (`"03"` for board n°3). This executes a *pattern* run in toggle mode (see section 3.3.1) and tests if the logical section of the board is running properly. Check that a file with name `P`*RUNID*`_b`*BOARDID*`.data.yaml` is indeed created in the *DATADIR* directory. Edit it and check that data values in the file are all equal to 819, corresponding to the **deskew** sequence (see section 3.3.1 for details).
If all is OK at this stage, this means that ADC and FPGA are working properly.

5- Then a complementary stage consists in launching:

```
./loopCalib.sh BOARDID 60 60
```

This execute a calibration run with $Att1 = Att2$=60, corresponding to a total attenuation of 33.5 dB to the quartz sine signal (see Eq. 1). Open the run file with python script with `anaData.py` and check that the average value of the signal is around a value of -0.4 V, with a standard deviation not above 80 mV. This allows to check that analog section of the board is also running properly.
The tools for this stage (scripts+analysis) have to be implemented again with the new data format.

6- Then launch:

```
./soft.sh BOARDID
```

This script launches an infinite loop recording data in the *minBias* mode (see section 3.3.4).
In the offline treatment, open the run file `M`*RUNID*`.data.bin` with the python script `readData.py` and display the signals. Check that the average value of the signal is ∼0.2 V, with a standard deviation of the baseline ∼0.2 V as well, for all three channels.
This allows to test if the antenna + analog part of the EB are also working fine. This furthermore gives an insight on the electromagnetic environment of the antenna. If this stage fails, the previous one (calibration) should allow to determine if the problem comes from the EB or the antenna.
You may also open the `R`*RUNID*`.data.bin.yaml` file to check in the event header that the GPS info is different from 0.

The detection unit can be considered as fully functionnal if all previous steps were successful. It may then be included in the Physics acquisition by adding the *BOARDID* value to the file `boardsIn.txt` in the `gp_daq/scripts` directory.

4.2 Normal operation

Launch `./loopPhys.sh`.
This will start a *Physics* run on all units listed in `boardsIn.txt`, with triggers on X and Y channels only. **ToDo: find a proper way to set individual triggers on channels & units independently (with a default value common to all)**. This script also requests slow control data every two minutes. Together with other slow control monitoring infos (see section 3.3.5 for details), the *MAXCOARSE* value is therefore saved to disk, and made available for a precise computation of the trigger time in the offline analysis (see section 2.4.3 for details).
Note that EBs will remain configured in this mode even though the DAQ process and data writting to disk are interrupted. To stop data emission from the boards, you will have to explicitly request it. This can be done by calling the `./loopStop.sh` script.
In the offline treatment, coincidences can be searched for with the `readData.py` script. The produced `R`*RUNID*`_coinctable.txt` file can then be analysed with a C software computing the best wave direction (plane wave hypothesis) and source position (spherical wave hypothesis) associated with each coincidence [10]. These reconstructions can eventually be analysed with the `readRecons.py` [9] script.

4.3 Monitoring data

To be completed: MinBias and Galactic fluctuation measurements, as well as trigger rate monitoring.

4.4 Calibration

To be completed: based on Calib procedure (see section 3.3.2).

5 Troubleshooting

Below are summarized the most frequent problems encounter on the GRAND-proto35 setup, and how they may be solved.

TO BE UPDATED

- **no ping:** check network settings on the PC (should be local network, with IP: 192.168.1.1). Check cable and fiber connection. Cycle power on the Front-End unit (may need to be done several times). If this does not work, go on field, and connect directly to unit from local fiber. If this works, then the problem comes from the fiber. If this does not work, then open the casing lid and check the Marvell leds (see section 2.4.7): first two should be blinking, two following should be on. If not, check that jumper is set on the right slot and that SFP is properly set in its slot. If nothing works, switch jumper to left slot (with power off) and test connection on field with Ethernet cable. If this works, then problems comes from the SFP module. If this does not work then the communication module has a problem and the unit should be brought back to lab for further test.
- **no DAQ communication (ie no response to `slcreq.sh`):** Ping Front-End unit. Issue the command `echo $DATADIR` in a terminal window of the DAQ PC, and check that the variable is indeed set to the folder where you want data to be written. Check that IP and MAC adress of the PC actually correspond to those given in `setAdress.sh`. Relaunch `setIP.sh`. Launch the `Wireshark` program. When a `SLCreq.sh` command is issued, a response from the Front-End Unit should appear in the packet traffic, with appropriate IP and MAC adresses from both sides.
- **Antenna signal baseline with value below 0 V:**
- **Antenna signal baseline with value above 0.7:**

6 References

[1] J. David, Caracteristiques de la carte frontale (2016).
URL `http://www.iap.fr/grand/wikigrand/images/6/6b/GRANDProto_CarteAna.pdf`

[2] Analog-Devices, Ad8310 data sheet (2017).
URL `http://www.analog.com/media/en/technical-documentation/data-sheets/ad8310.pdf`

[3] P. Nayman, Daq for trend (2016).
URL `http://www.iap.fr/grand/wikigrand/images/8/87/TRENDDAQv23.pdf`

[4] Texas-Instrument, Ads6424 data sheet (2013).
URL `http://www.ti.com/lit/ds/symlink/ads6424.pdf`

[5] u blox, u-blox 6 receiver description (2013).
URL `http:////www.u-blox.com/sites/default/files/products/documents/u-blox6_Receiv`

[6] J. Gu, Grandproto35 daq (2019).
URL `http://github.com/TREND50/pyef`

[7] O. Martineau, Grandproto35 data format (2018).
URL `http://www.iap.fr/grand/wikigrand/images/e/ee/DAQStructure.pdf`

[8] J. Gu, The pyef package (2018).
URL `http://github.com/TREND50/gp_daq`

[9] O. Martineau, Analysis tools for grandproto35 data (2018).
URL `http://github.com/TREND50/gp_ana`

[10] V. Niess, Source reconstruction tools for grandproto35 data (2009).
URL `http://github.com/TREND50/gp_recons`