

TPG Validation & DTP-Pattern Review

Kunal Kotheekar

17/03/2021



Scope of the talk

- We use DTP-Simulations (A python based library) to validate the firmware after every release.
- For quick comparisons, it was decided that we should have some “ready” patterns available.
- Over the time these patterns has grown themselves into a big library.
- You can check all the patterns and their description listed here:
[dune-daq/readout/dtp-patterns](https://github.com/dune-daq/readout/dtp-patterns)
- We have in total 17 no. of patterns available right now.
- The scope of this talk is to review, analyze, and discussion on how to improve these “dtp-patterns”.
- In the larger scheme of things this talk also concerns itself with a validation framework for firmware, and in general complete upstream DAQ system.



- **How the patterns are defined?**

→ The easy way to understand the patterns is in terms of a 64 ticks (adc values) packet, which is a unit on which hit finding works. A hit finder responds whenever the adc values in a packet crosses a predetermined threshold.

→ Based on this the text files which contains various combination and sets of adc values are produced at each stage of firmware units, collectively known as **ntp-patterns**. The different text files at different firmware units are described in table 1.

- **How are the patterns categorized?**

→ The patterns are categorized mainly into four categories:

1. Fixed ADC
2. Fixed Hits
3. Unique ADC
4. Unique Hits

They are then divided into sub category (A,B,C ...) based on various combinations, but inheriting the properties of their main class.

- For the complete description of patterns, please refer to this document:

[DTP-Patterns Description](#)



DTP-Patterns

- **These four pattern categories are used for following primary purposes:**
 - FixedADC: Verifying the functionalities of pre-hit finding components.
 - FixedHits: Verifying basic functionalities of the hit finding module.
 - UniqueADC: Ensure the data-flow through the whole chain.
 - UniqueHits: These patterns are closer to a random adc generated in the experiment and are used to verify the functioning of the complete chain.
- **How were the DTP-patterns validated in first place?**
 - All these patterns were validated against LarSoft framework, which used the same seed file as that of DTP-simulation.
 - There is a 100% agreement in simpler patterns, but in complicated patterns the agreement is less.
 - These corner cases are listed here: [TPG LArSoft Validation#Issue57](#)



- **Are all the patterns tested in firmware (in v1.0.0 & v1.1.0)?**
 - Yes, the issues are also listed. Please refer [Testing_V1.0.0](#) & [Testing_V1.1.0](#)
 - **Suggestion:** The testing needs to be more organized, the format should also contain the results with each pattern.
- **Are more patterns needed?**
 - FixedADC, FixedHits, UniqueHits & UniqueADC patterns are good to check the basic functionalities of firmware chain.
 - But for more robust validation, we need more simulated data. The reason is that we have seen some shortcomings when we operated with protoDUNE-I data.
 - **Proposal:** To have a more simulated data with LArSoft, need to convert it to wib33b though.
 - **Proposal2:** check with saved ProtoDUNE-I files again and again (if the raw data is saved)

TPG Validation



- **How did the data validation with protoDUNE-I go?**
 - Joel Greer found no. of bugs in the firmware in his analysis of protoDUNE-I data. The summary can be found here: [PD1HitAnalysis](#)
- **Are the bugs still present in the firmware?**
 - There is no confirmation as such study was not repeated on latest firmware versions.
- **What should be the next steps in validation?**
 - Someone need to repeat all the tests performed by developers and should present a report after every major release.
 - Hit Analysis should be ongoing exercise and should be part of validation framework.
 - Need to set-up a validation framework for US-DAQ. This would consist firmware tests, hit analysis, identifying some plots/information which will be produced at the end of the release (kind of a data quality monitoring)

Validation Framework



- **TP implementation performance characteristics:** How many are the actual hits present in the data to how many are we getting through firmware? Might need aggregation of TPs from large simulated data
- Need to answer questions like how well the firmware TPs convert into offline selection quantities.
- Recording of the raw data and comparing them with the firmware TPs
 - The test stands should also be part of validation framework, should be able to run the simulated data.
 - On more system level timing and TPG integration and validation should be done.
- **Do we need TPG algorithm performance evaluation?**
 - May be. At the moment things look fine with simple threshold algorithm. How we determine this threshold and how we should optimize can be a topic of research.
 - For that LArSoft framework has relevance, it can give us a parallel model to test the simulated data and at the same time allow us to experiment with trigger algorithm.

Plans



- On my level, I am going to start with Hit Analysis taking guidance from JG's last year efforts.
- Another thing I am keen on is comparing firmware TPs to off-line selection quantities. (This comes under data-selection task, and need to contact relevant people for this)
- Trigger Primitive implementation performance characteristics:
→ This will need ADC values (in binary form) and finding hits through them and then comparing them to actual firmware TPs.
- Putting some of this information into some characteristic plots which can be put into DQM.



Back-up



DTP-Patterns

- Table 1: Different data-types in firmware

Firmware Components	Firmware: data-type	Description	DTP pattern files
Data Reception			



- Fixed ADC:
→ In these patterns files, the adc values across all the packets are fixed. The combinations are set such that a part of all the values across the channels are fixed at some niche value, the remaining at some other value. These pattern files **do not produce any hit. The primary use of such patterns is verifying the functionalities of pre-hit finding components in firmware.**
- Fixed Hits:
→ In these pattern files, there are identical hits (occurring in same packet, same tick) across all channels. The sub-patterns (A,B,C,D...) then plays on idea of shifting this hit and changing the hit features across the channels. The pattern file **produce reoccurring identical hit. The use of such pattern is a primary inquiry on hit-finder and it's sub-modules.**
- Unique ADC:
→ In these pattern files, each channel has same adc in a packet, but the pedestal value in each channel is proportional to it's channel no. **These patterns are used primarily to ensure that data is flowing correctly through the whole firmware chain, these may or may not produce hits.**
- Unique Hits:
→ In these pattern files, a unique hit (every occurrence of a hit has a different feature such as adc peak value) is placed on different wires, and across different packets. The sub patterns then plays on the idea of having one kind of unique pattern varying the features of the hit and it's occurrence. They produce the hits at a predetermined locations verifiable by output files. **These patterns are closer to a random adc generated in the experiment and are used for verify the functioning of complete chain.**