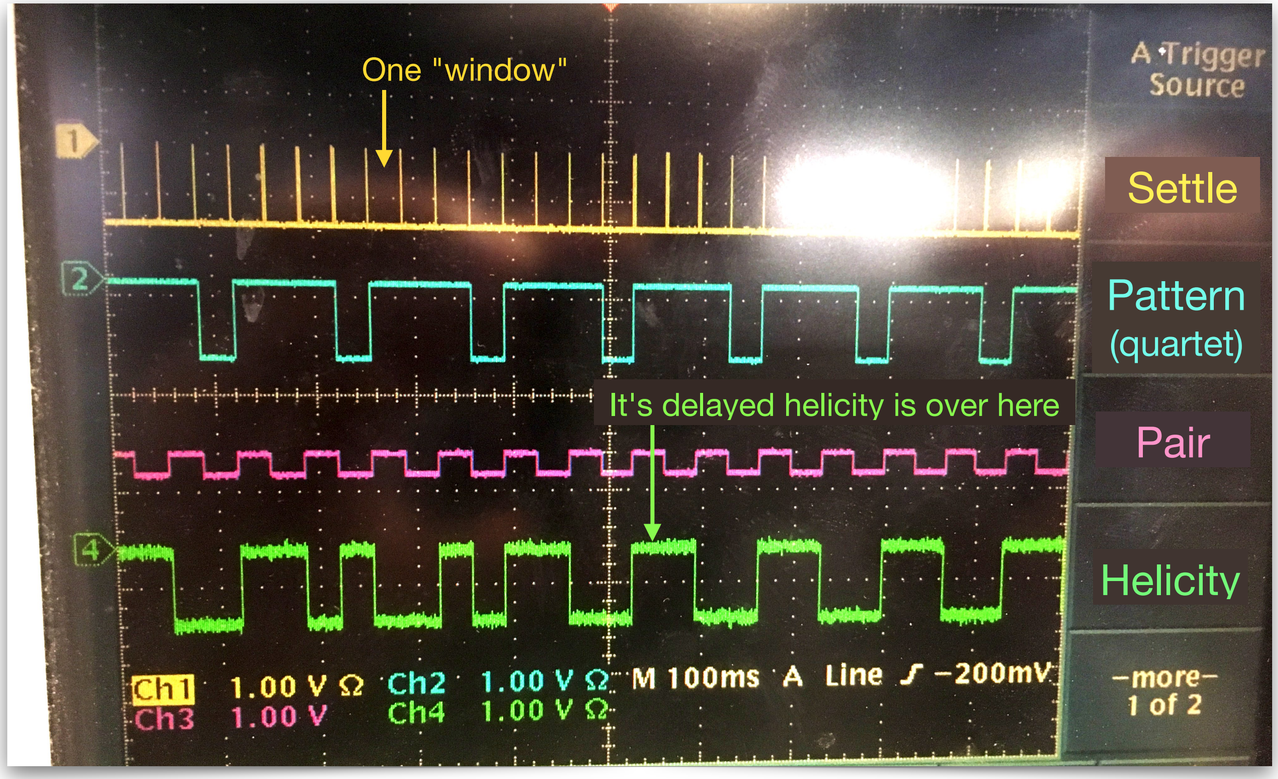
Helicity Updates

CLAS Collaboration Meeting

N. Baltzell - March 12, 2024

Introduction

- The polarization of CEBAF's electron beams is generated and controlled in the injector, with helicity the polarization direction, parallel or anti-parallel
- To reduce systematic errors, helicity is changed rapidly and automatically, systematically and pseudorandomly
    - different, configurable pattern types and clock frequencies, most commonly "quartet" and 30 Hz
- To get the current helicity state in experiments, optical fibers are routed from the injector to each hall's DAQ
    - this includes the helicity state, of course, plus 3 "control" signals for, e.g., vetoing during transitions, selecting state pairs, integrity checking the sequence, seeding the pseudorandom generator
- To further reduce systematic errors, the signals can also be (heavily) delayed, called "delayed reporting"
    - by a multiple of the clock frequency, most commonly 8 "windows" (1/4 second at 30 Hz)
    - that's a complication for trigger/event-based paradigm, traditionally addressed in software



Helicity Delay Correction

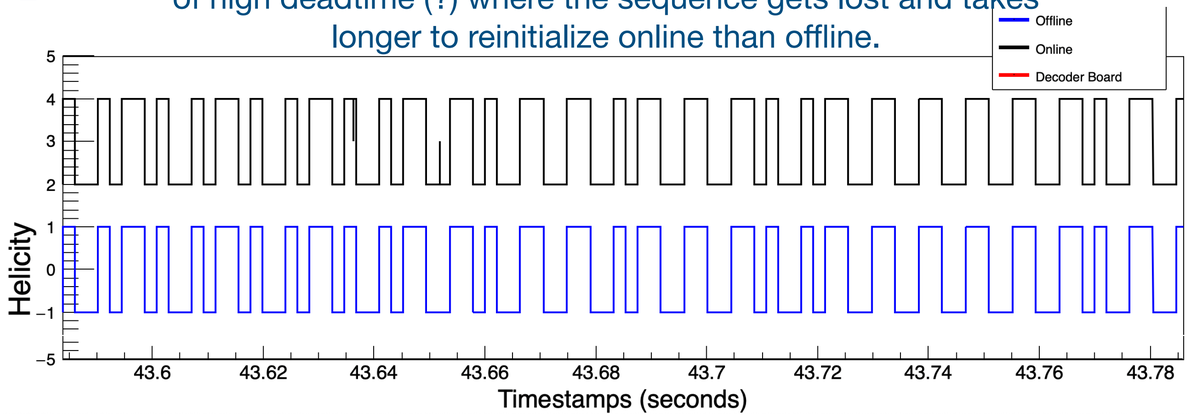
- With delayed reporting, one needs more information than just a snapshot in time of the "live" helicity signals
  - Minimally, ***N, time-ordered, helicity windows***, and counting state changes and integrity checking ...
  - One can use the pseudorandom generator sequence, where N = 120 or 3.5 seconds, before the given event
  - Or, if you can see into the future, walk forward in the measured sequence, where N = 8, or 0.25 seconds, right after the given event

    * for the most common delay=8, pattern=quartet, clock=30 Hz

- We do both, based on FADC readout of the "live" helicity signals in every trigger/event (***which requires trigger rate >> helicity clock***)
  - Online: `HEL::online`, based on the pseudorandom generator, done in "L3"
  - Offline: `HEL::flip` → postprocessing → `REC::Event`, based on the measured sequence, if it matches the pseudorandom generator if seeded

- We also have other approaches available, e.g., triggering on helicity state changes, using a multi-channel, helicity-latching scaler, but all have their own complications and never really get rid of the "more information" problem above

The "online" and "offline" corrections normally give the same result, with some minor glitches in the online one, except in cases of high deadtime (?) where the sequence gets lost and takes longer to reinitialize online than offline.



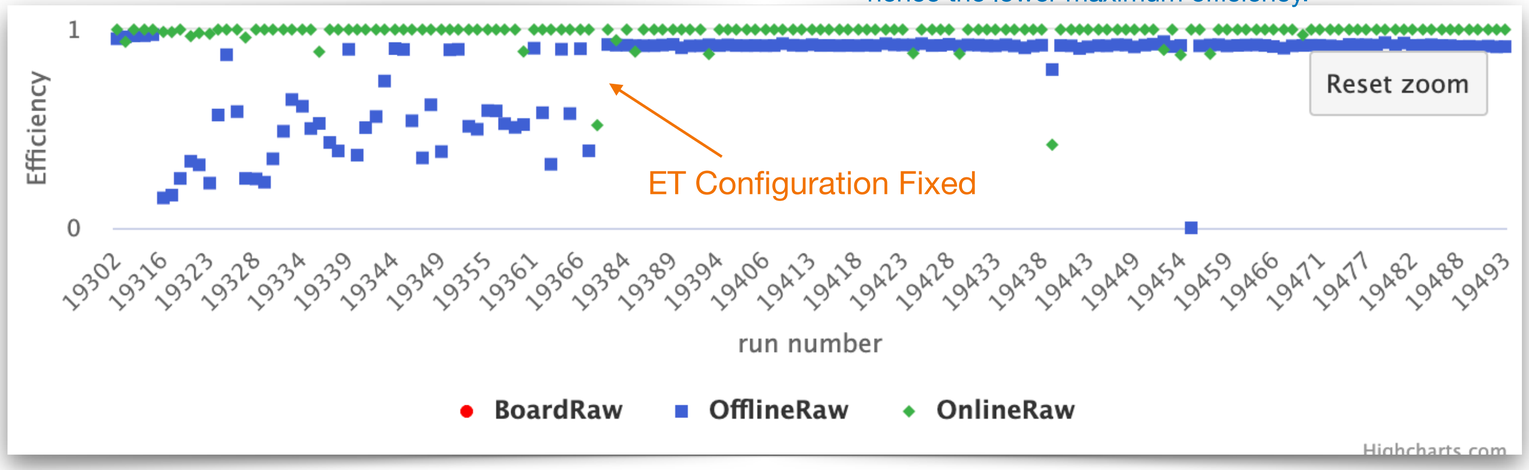
The pseudorandom sequence is based on a 30-bit seed from the previous helicity states, and then predicts into the infinite future, based on some fast bitwise manipulations and number of windows.

```
/**
 * Pattern delay correction, as specified by JLab's injector group.
 * @param helicities the first helicity of the previous SEQUENCE_LENGTH patterns
 * @param patternDelay number of patterns
 * @return delay-corrected helicity of the first window in the pattern
 */
public static byte getPatternHelicity(int helicities, byte patternDelay) {
    int bit = 0;
    int register = 0;
    for (int i=RNG_REGISTER_SIZE-1; i>=0; --i) {
        register = ( ((helicities>>i)&1) | (register<<1) ) & 0x3FFFFFFF;
    }
    for (int i=0; i<patternDelay; ++i) {
        int bit7  = (register>>6)  & 1;
        int bit28 = (register>>27) & 1;
        int bit29 = (register>>28) & 1;
        int bit30 = (register>>29) & 1;
        bit = (bit30^bit29^bit28^bit7) & 1;
        register = ( bit | (register<<1) ) & 0x3FFFFFFF;
    }
    return (byte)bit;
}
```

Event Ordering (1)

- The CODA system, at least the way Hall B uses it, is designed such that events recorded in EVIO files should always be time-ordered
    - We have seen misordering in the past, on occasion, but not enough to have significant impact
- But between October 2023 and January 2024, one ET ring, the one used for online monitoring, was in an incorrect configuration, blocking versus non-blocking.
    - **The result is large event misordering, quantized in blocks of 10 events. Even seen mixing events across runs (probably, hopefully very rare)!**
    - There's also some event loss, not yet fully quantified, seen to be up to ~1% in some cases.

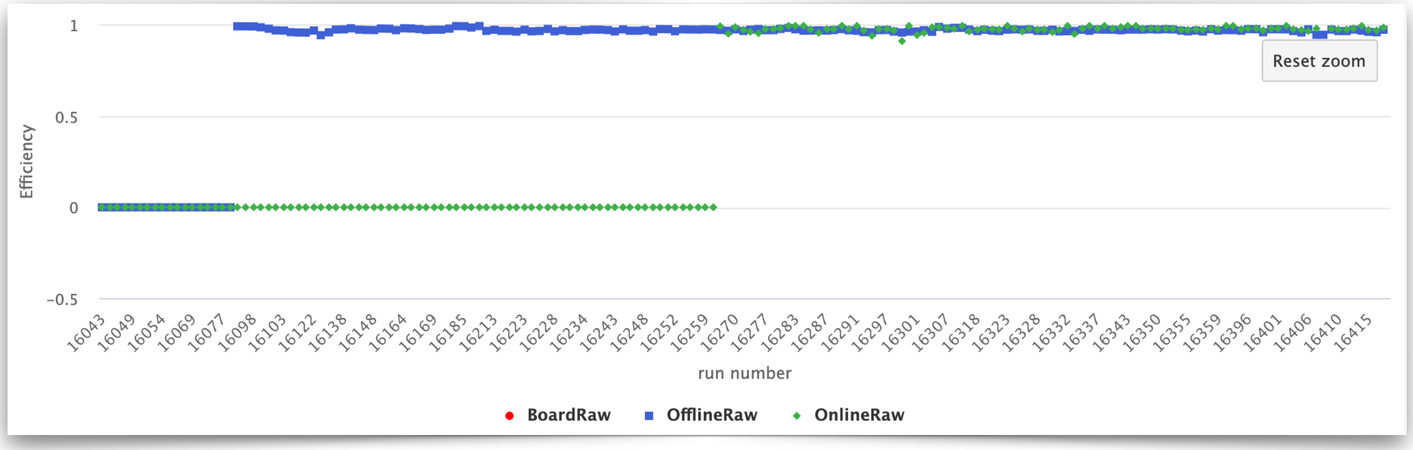
Note, here the offline correction was is performed on single files, hence the lower maximum efficiency.



- **First noticed in low efficiency of the offline helicity delay correction, which requires time-ordered events.**
    - Online monitoring efficiency is currently too low to really be sensitive to it. Would need much more performant EVIO→HIPO conversion.
    - And the online delay correction sees the events prior to this ET-induced misordering, so it's also insensitive.

Event Ordering (2)

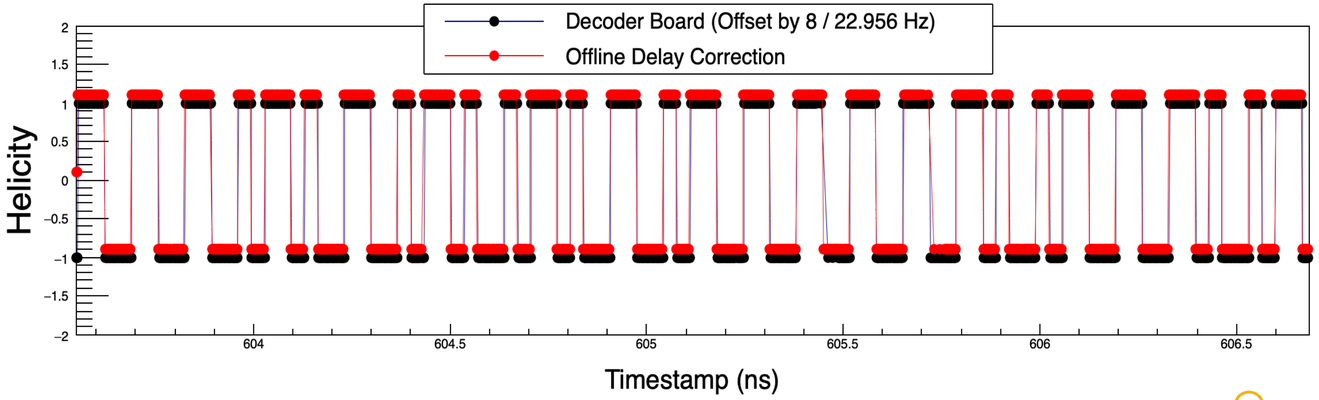
- Why not "just forget about the offline" correction and rely only on the unaffected online?
- Well, we've already got it, and it's required for multiple CLAS12 data sets when the online version isn't available, so it's best to try to keep it working for all, at least as a cross check.

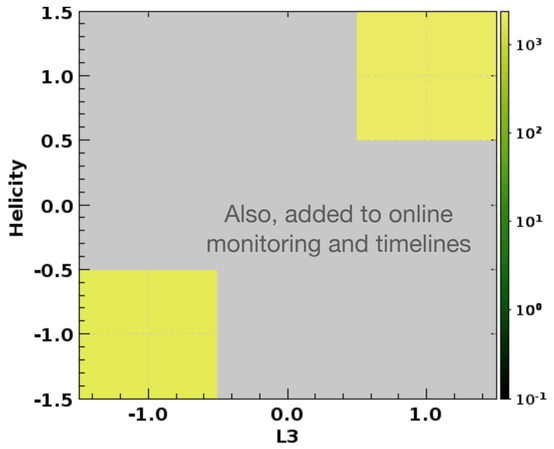


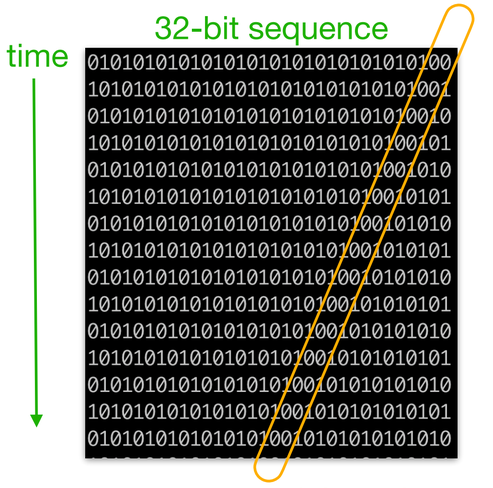
- So, as of COATJAVA 10.0.6, https://github.com/JeffersonLab/coatjava/releases/tag/10.0.6
  - Now time-sorting helicity states during the EVIO→HIPO transition (and ignoring events from other run numbers), essentially by reading (the minimal information) from every event twice and buffering it in between. Negligible computing resource overhead. Should have been doing it anyway ...
  - Considered also writing out the HIPO events sorted, but since CLARA will unsort them downstream anyway, decided it better to minimize data manipulation and preserve their EVIO ordering.
- **Also added efficiency of the online and offline delay corrections to clas12mon timelines.**

Helicity Decoder (HD) Board

- Developed by Ed Jastrzembski in the Fast Electronics group at JLab
- Intended to be the end of all helicity delay corrections at Jlab, and critical to support the large helicity clock rates required for high-precision, parity experiments (where helicity clock ~ trigger rate in Hall B)
- Latches on helicity control signals, buffers the previous N windows of all helicity signals, and on every readout provides everything necessary to delay correct the current event based on the pseudorandom sequence, plus lots of additional information for debugging, integrity checking,
- Using it in Hall B
  - EVIO→HIPO decoding by Raffaella and Maurik
  - On first analysis, we (re)learned our signals are inverted
    - → HD firmware update #1 - configurable input/output polarity (thanks Ed!)
  - After that, things look great and agree with our other delay corrections, most of the time.
  - We see something consistent with occasional noise, every few minutes, causing false state advance and corrupting 4 seconds (with an 8 ns clock, it's much more sensitive than our trigger-based approaches)
    - → HD firmware update #2 - optional noise filtering
  - Next, more testing, closer inspection of the optical signals ...







Summary

- Helicity delay corrections have been working for a few years, but hit a surprise with heavily misordered events during RG-D
    - The root cause was eventually found and fixed, meanwhile it corrupted our offline helicity delay correction, addressed in COATJAVA 10.0.6
    - Also added better monitoring of online delay correction and new helicity decoder board, both in mon12 and timelines
- The new helicity decoder board seems very close to ready for production, and is expected to make helicity delay corrections a non-issue