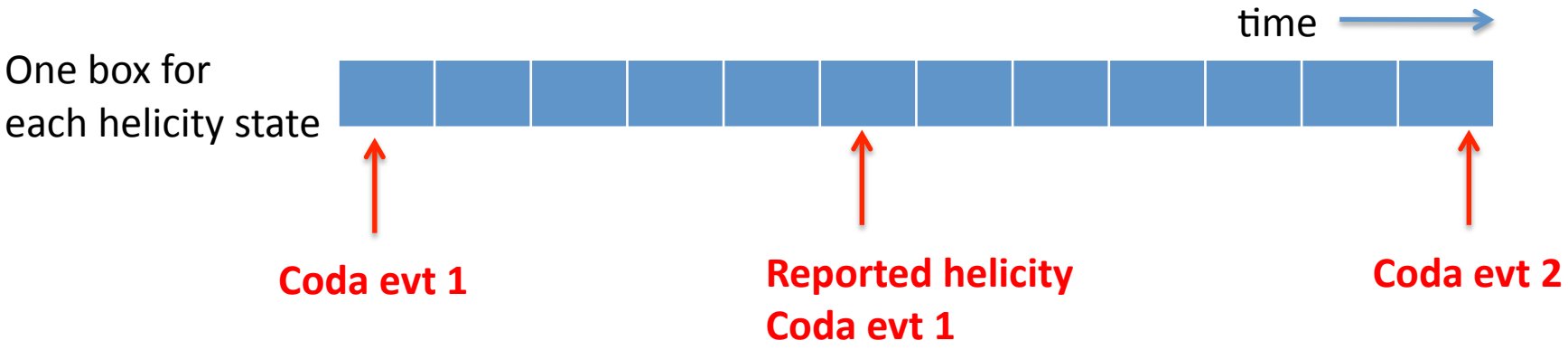
J. Roche, December 8, 2010

DVCS high rate helicity flip scheme.

The issue:

- QWEAK runs with a helicity flip rate as high as f = 1kHz.
- DVCS registers CODA events at rates of the order of 100 Hz.
- The helicity reported by the injector at a given time is
  - NOT the actual helicity of the beam at that time.
  - the helicity of the beam N helicity windows before this moment.



(Hall A in QWEAK Helicity Mode)

The technical documentation by R. Michaels is available at

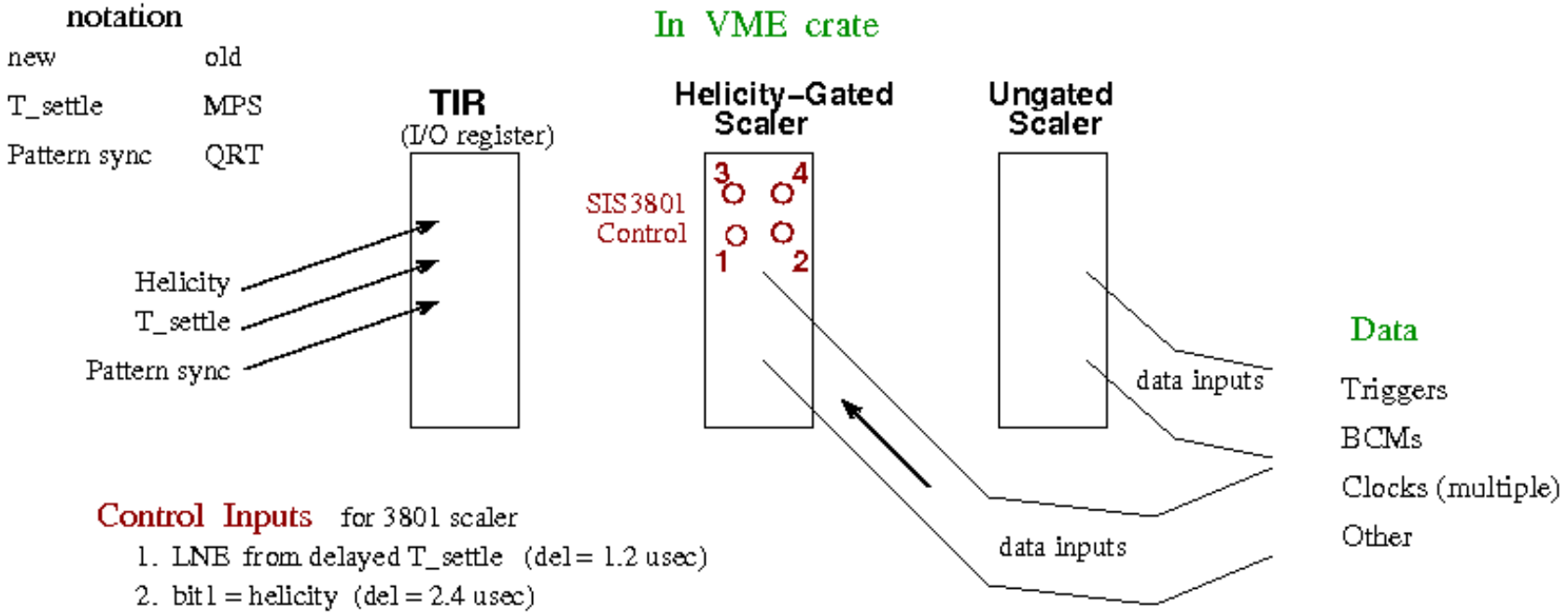
http://hallaweb.jlab.org/equipment/daq/qweak_helicity.html

A new helicity board (installed November 2009) features:

- several modes: Beam sync (30, 120 & 240 Hz) , free-running mode
- several patterns: e.g.: (+--+) (+--+ +--+ +--+ +--+ -++- -++- -++- -++-)
- several delays: the reported helicity is not the actual helicity. The actual helicity at the time of one event will be reported N helicity windows later.
- new pseudo-random generator: 30 bits. The polarity of each pattern is predictable given a seed. The seed can be extracted from 30 consecutive measurements of the pattern polarity.

Schematic of Electronics for Helicity Info in Hall A HRS during Qweak

R. Michaels
July 27, 2010



Control Inputs for 3801 scaler

1. LNE from delayed T_settle (del = 1.2 usec)
2. bit1 = helicity (del = 2.4 usec)
3. bit2 = pattern sync (del = 2.4 usec)
4. Gate = T_settle with appro. polarity and undelayed. (TRUE disables counting)

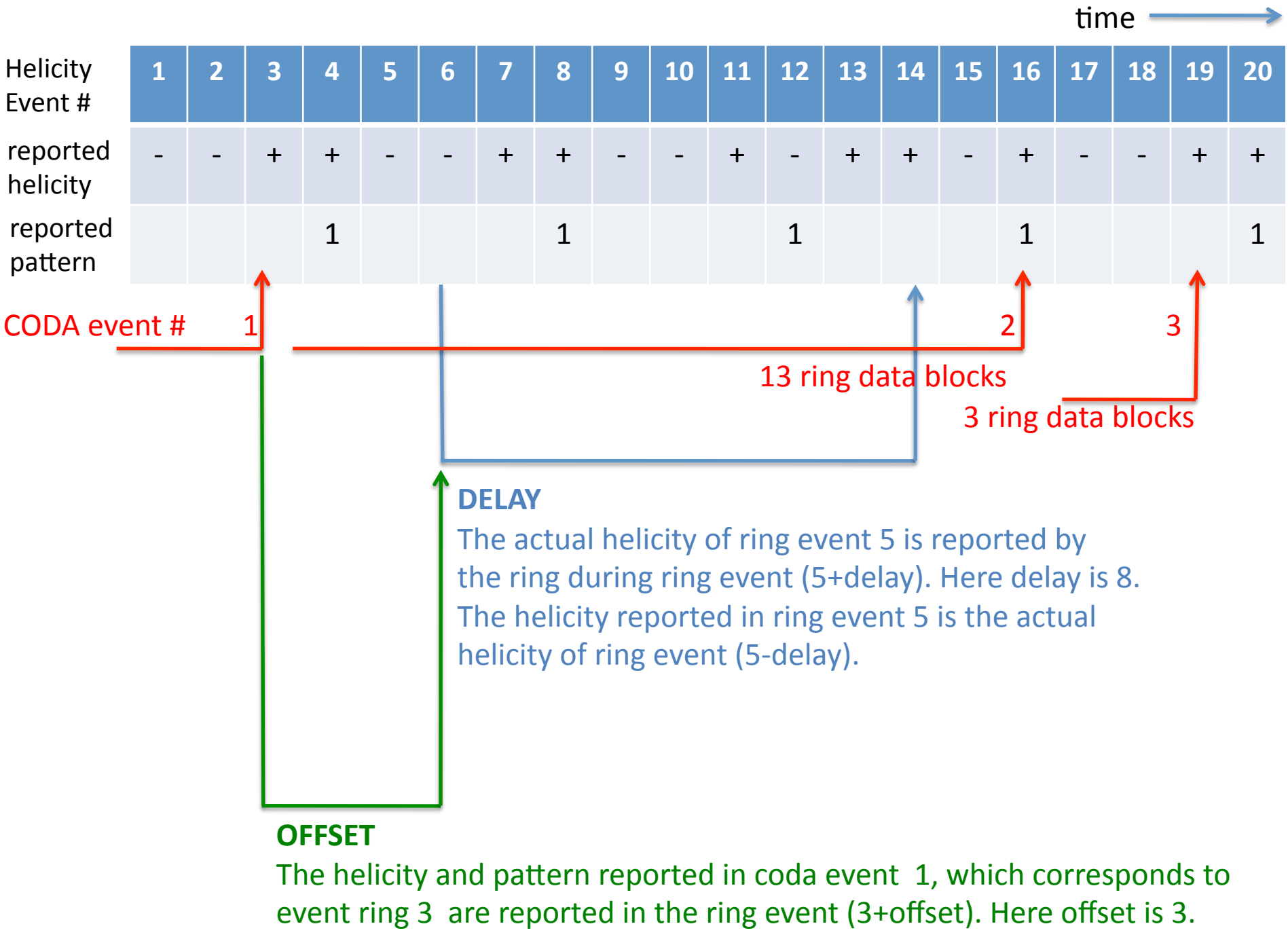
At frequency "f" (f = 1 kHz normally) we must:

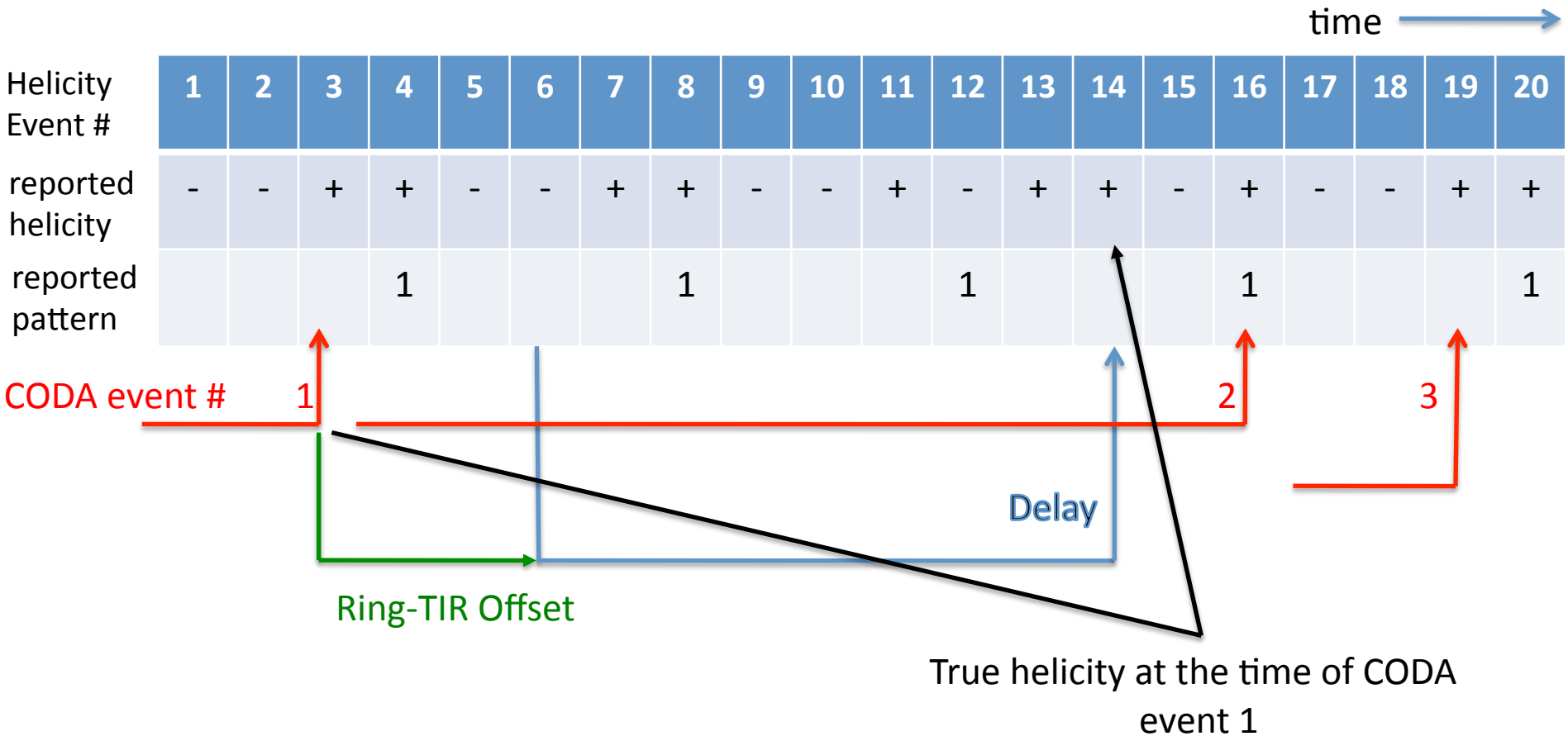
1. Read helicity–gated scaler with zero deadtime.
2. Keep a queue in memory of the cpu of [helicity, pattern sync, and clock(s)] ← Ring
3. Sort the plus and minus helicity, adding the data to "virtual" scalers for online consumption.
4. Check data for errors. Issue warnings.

For each trigger accepted by Trigger Supervisor, we must:

1. Read the (a) TIR bits and (b) clocks from ungated scaler. Also ADCs backup signals
2. Flush the memory queue (see 2 above), to have full sequence of helicity data.

Virtual scalers (R. Michaels) have been so far our main monitoring systems for helicity related data.


time
Helicity Event #
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
reported helicity
- - + + - - + + - - + - + + - + - - + +
reported pattern
1 1 1 1
CODA event #
1
2
3
13 ring data blocks
3 ring data blocks
DELAY
The actual helicity of ring event 5 is reported by the ring during ring event (5+delay). Here delay is 8. The helicity reported in ring event 5 is the actual helicity of ring event (5-delay).
OFFSET
The helicity and pattern reported in coda event 1, which corresponds to event ring 3 are reported in the ring event (3+offset). Here offset is 3.



Two solutions to determine the actual helicity of a CODA event:

- Keep the event in memory up to when the real info is extracted from the CODA file.
  Can become quickly cumbersome.

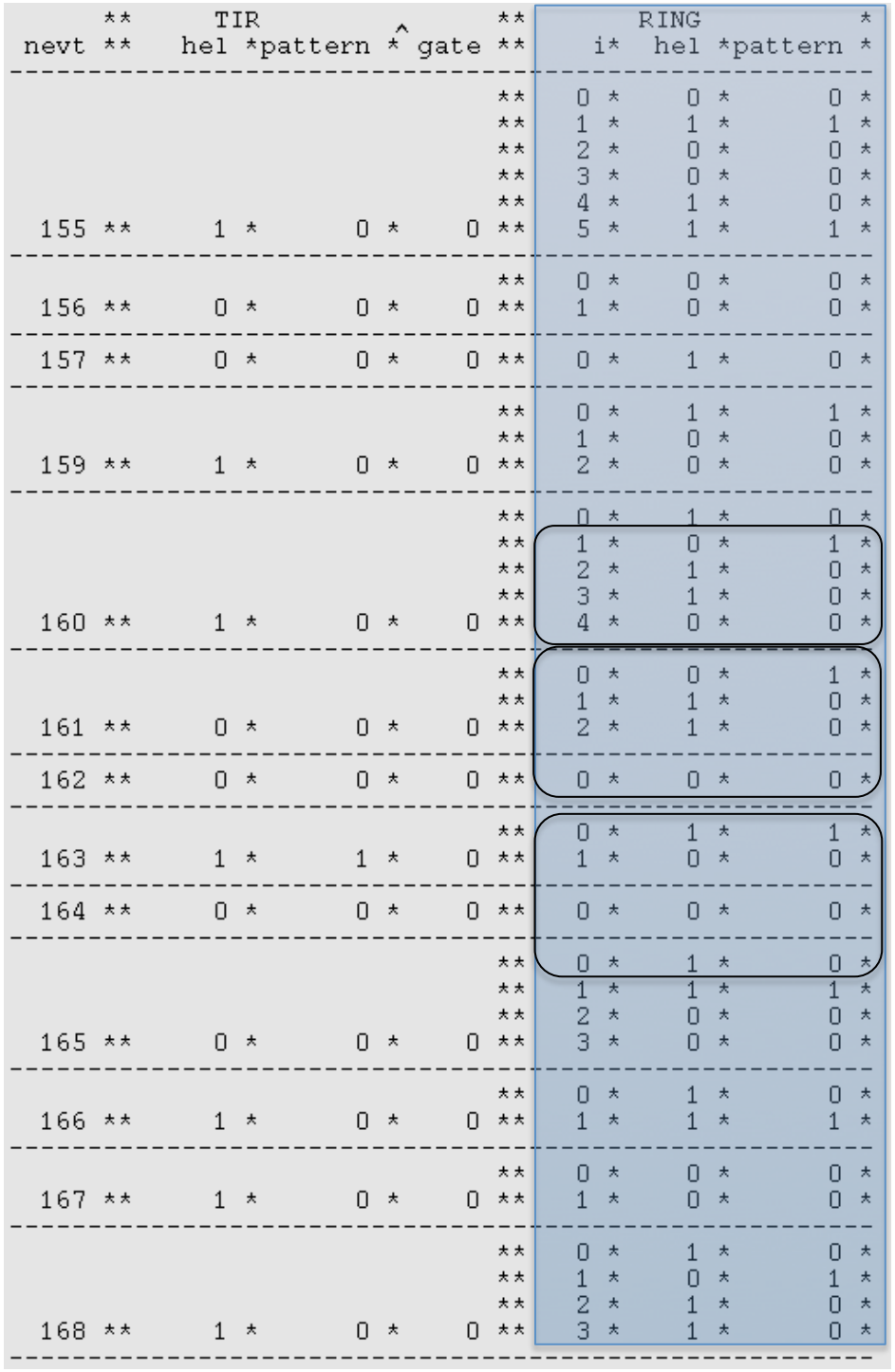
- Predict the helicity state based on the pseudo-random helicity generator behavior.
  Need to implement serious checks of the validity of the seed of the generator and the offset.

  This is what we choose to do

nevt	TIR hel	pattern	gate	RING i	hel	pattern
				0	0	0
				1	1	1
				2	0	0
				3	0	0
				4	1	0
155	1	0	0	5	1	1
				0	0	0
156	0	0	0	1	0	0
157	0	0	0	0	1	0
				0	1	1
				1	0	0
159	1	0	0	2	0	0
				0	1	0
				1	0	1
				2	1	0
				3	1	0
160	1	0	0	4	0	0
				0	0	1
				1	1	0
161	0	0	0	2	1	0
162	0	0	0	0	0	0
				0	1	1
163	1	1	0	1	0	0
164	0	0	0	0	0	0
				0	1	0
				1	1	1
				2	0	0
165	0	0	0	3	0	0
				0	1	0
166	1	0	0	1	1	1
				0	0	0
167	1	0	0	1	0	0
				0	1	0
				1	0	1
				2	1	0
168	1	0	0	3	1	0

Typical CODA events (run 8640)

- The ring also records 4 other words
  - Time elapsed between two ring loads
  - Charge accumulated
  - Triggers (T10 and T3)

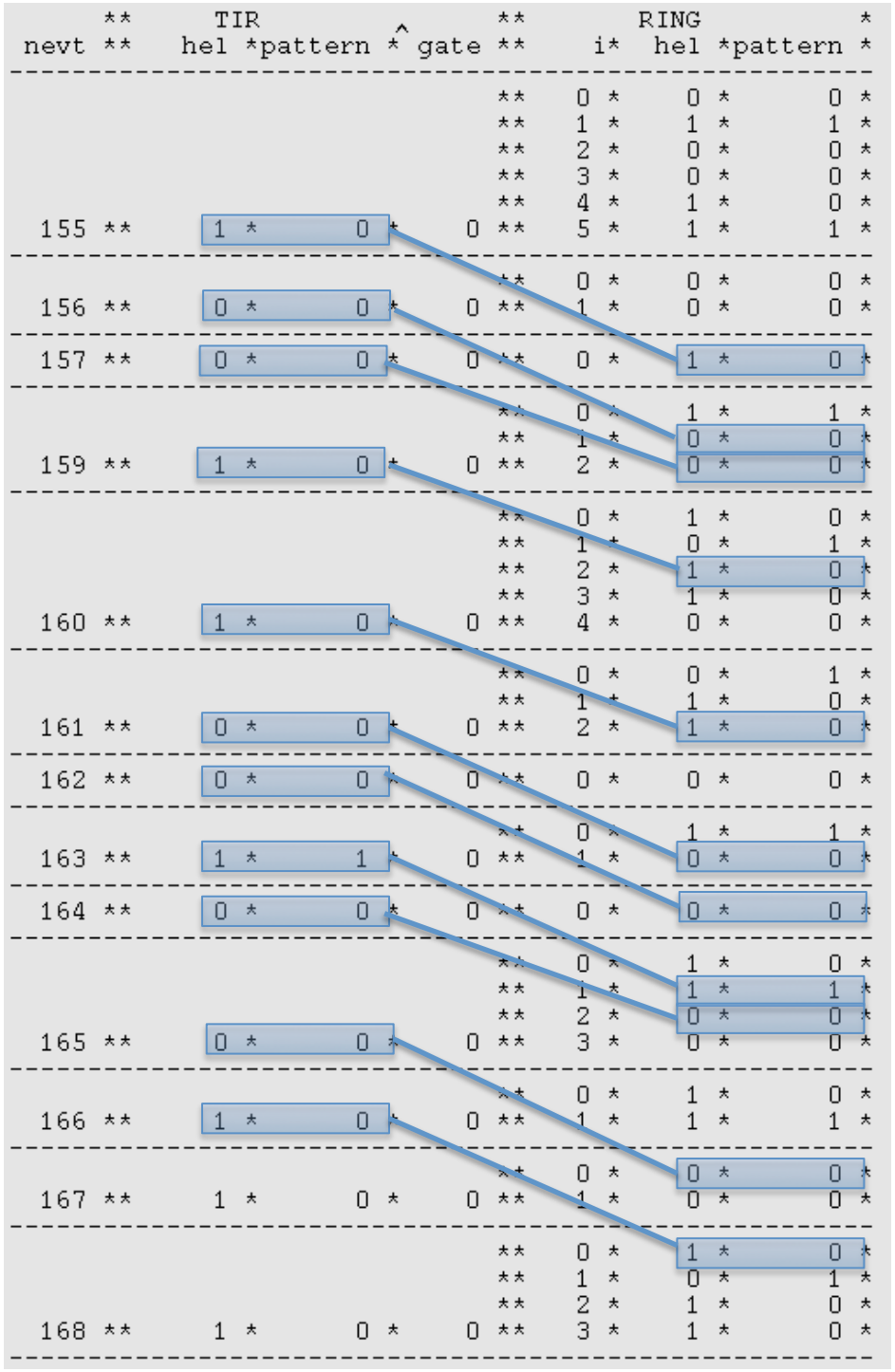


nevt	TIR hel	TIR pattern	TIR gate	RING i	RING hel	RING pattern
				0	0	0
				1	1	1
				2	0	0
				3	0	0
				4	1	0
155	1	0	0	5	1	1
				0	0	0
156	0	0	0	1	0	0
157	0	0	0	0	1	0
				0	1	1
				1	0	0
159	1	0	0	2	0	0
				0	1	0
				1	0	1
				2	1	0
				3	1	0
160	1	0	0	4	0	0
				0	0	1
				1	1	0
161	0	0	0	2	1	0
162	0	0	0	0	0	0
				0	1	1
163	1	1	0	1	0	0
164	0	0	0	0	0	0
				0	1	0
				1	1	1
				2	0	0
165	0	0	0	3	0	0
				0	1	0
166	1	0	0	1	1	1
				0	0	0
167	1	0	0	1	0	0
				0	1	0
				1	0	1
				2	1	0
168	1	0	0	3	1	0

Typical CODA events (run 8640)

- The ring also records 4 other words
  - Time elapsed between two ring loads
  - Charge accumulated
  - Triggers (T10 and T3)

- The ring measures the helicity variables without any deadtime
  - Checked with Tsettled as trigger
  - Pattern polarity follows pseudo-random algorithm
  - The max. depth of the ring need to be adjusted to match the helicity flipping rate and CODA trigger rate (current depth=500)



Typical CODA events (run 8640)

- The ring also records 4 other words
  - Time elapsed between two ring loads
  - Charge accumulated
  - Triggers (T10 and T3)

- The ring measures the helicity variables without any deadtime
  - Checked with Tsettled as trigger
  - Pattern polarity follows pseudo-random algorithm
  - The max. depth of the ring need to be adjusted to match the helicity flipping rate and CODA trigger rate (current depth=500)

- The ring and the TIR events are offset one compared to the other one.
  - Non trivial offset of 3 for DVCS
  - Checked the stability of the offset for various helicity rates (from 120Hz to 1kHz)

The ongoing implementation into Podd follows the THaG0Helicity scheme

- class THaQWEAKHelicity : public THaHelicityDet, public THaQWEAKHelicityReader
  - THaQWEAKHelicityReader: Reads the TIR and Ring information from the CODA file
  - THaQWEAKHelicity:
    - Extracts seed from 30 successive pattern recorded by the ring
    - Compares the measured sequence of the pattern polarities against the predictions of the pseudo random generator (ring)
    - Predicts the actual helicity of the ring and triggered event (taking into account the delay for the reporting and the offset between the ring and the TIR reporting)
    - Checks event by event the offset between the ring and the TIR reporting
  - THaHelicityDet: interfaces these routines with the main analyzer

```
THaApparatus* B = new THaIdealBeam("B","Idea Beam, for Test Only");
    gHaApps->Add( B );
    B->AddDetector(new THaQWEAKHelicity("qweakhel.L","Left arm QWEAK helicity") );
```

The ongoing implementation into Podd: follows the THaG0Helicity scheme

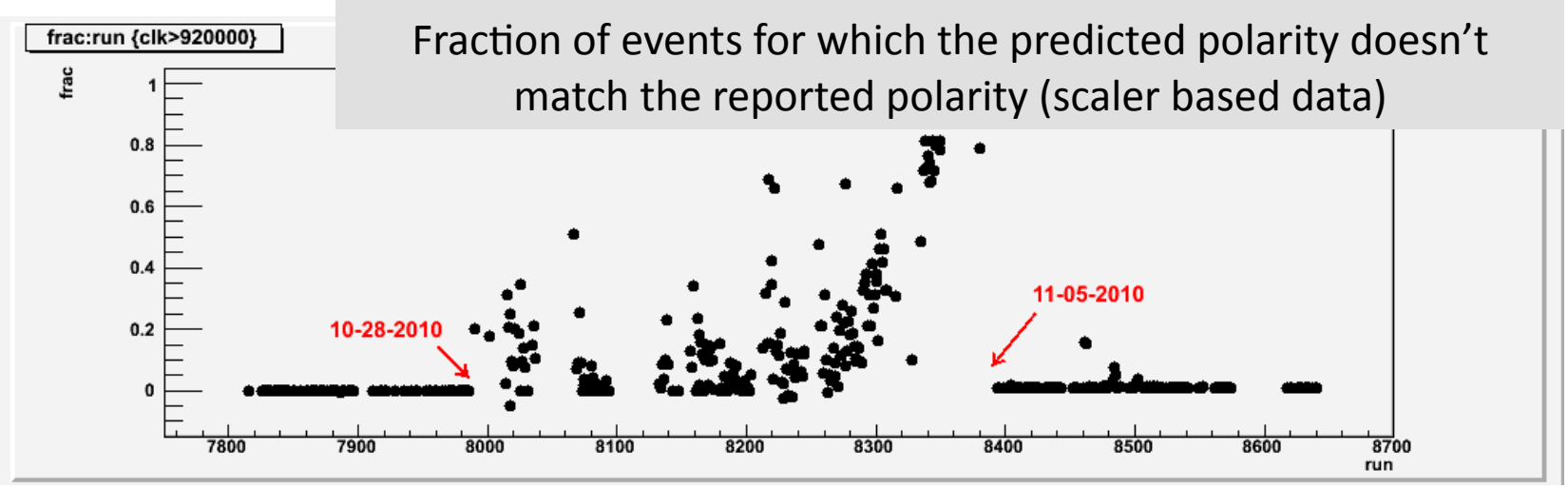
- class THaQWEAKHelicity : public THaHelicityDet, public THaQWEAKHelicityReader
  - THaQWEAKHelicityReader: Reads the TIR and Ring information from the CODA file
  - THaQWEAKHelicity:
    - Extracts seed from 30 successive pattern recorded by the ring
    - Compares the measured sequence of the pattern polarities against the predictions of the pseudo random generator (ring)
    - Predicts the actual helicity of the ring and triggered event (taking into account the delay for the reporting and the offset between the ring and the TIR reporting)
    - Checks event by event the offset between the ring and the TIR reporting
  - THaHelicityDet: interfaces these routines with the main analyzer

```
THaApparatus* B = new THaIdealBeam("B","Idea Beam, for Test Only");
    gHaApps->Add( B );
    B->AddDetector(new THaQWEAKHelicity("qweakhel.L","Left arm QWEAK helicity") );
```

- Status (12-08-2010)
  - Debugging phase for core routines (see above)
  - More work needed to interface with text-based database
  - J.R will keep on working on this up to submission into cvs.

Caveat to this neat scheme: when the ring "rings".

- When every thing is fine, the seed one extracts with 30 patterns lets you predict the polarity of every following pattern. Unless something goes wrong.



Fraction of events for which the predicted polarity doesn't match the reported polarity (scaler based data)

The effect was due to double pulsing in the LNE of the ring:

- Symptoms: End of run routines reports this fraction, time elapsed between two ring loads close to 0
- The problem was fixed with a longer LNE gate. This longer gate will need to be adjusted if the helicity is flipped at more than 1kHz.

- Podd analysis:

  Need to implement backup plans that doesn't use the ring info.

  Will do something similar to what is in the THaG0Helicity routine.

The ongoing implementation into Podd: follows the THaG0Helicity scheme

- class THaQWEAKHelicity : public THaHelicityDet, public THaQWEAKHelicityReader

- Status (12-08-2010)
  - Debugging phase for core routines that relies on the RING info only.
  - More work needed to interface with text-based database.
  - Need to implement the back up analysis when the ring info is garbage (ie ring “ringing”).
  - J.R will keep on working on this up to submission into cvs.