Helicity Decoder for E08-027

Chao Gu

University of Virginia

cg2ja@virginia.edu

August 9, 2014

The E08-027 (g2p) experiment uses inclusive electron scattering cross-section differences to determine the g_2 structure function. The cross-section differences are defined between two different helicity states of the incoming electrons. Thus the helicity scheme of the beam needs to be recorded and decoded correctly to obtain the true helicity states of the incoming electrons. This technical note will summarize the helicity scheme during the experiment and the helicity decoder for the offline analysis.

1 Introduction to the Helicity Scheme

At Jefferson Lab, polarized electrons are generated by sending circular polarized laser to the photocathode of the electron gun. The spin of the photo-emitted electron is correlated to the circular polarization state of the photon. It can be either aligned parallel (1 or +) or anti-parallel (0 or -) to the electron momentum direction, which are the two helicity states of the electron. A programmable logic generator known as the Helicity Control Board is installed at the injector to control the helicity of the electron beam [1]. It generates a logic signal called Helicity Flip to control the polarity of the high voltage of the Pockels Cell on the Laser Table in the injector. The Pockels Cell is a crystal that acts as a quarter-wave retardation plate when a high voltage is applied on it, thus, flipping the polarity of the HV of the Pockels Cell changes the circular polarization state of the laser and hence changes the helicity of the electron beam [2].

The actual sequence of the beam helicity is a series of identical length helicity windows in which helicity

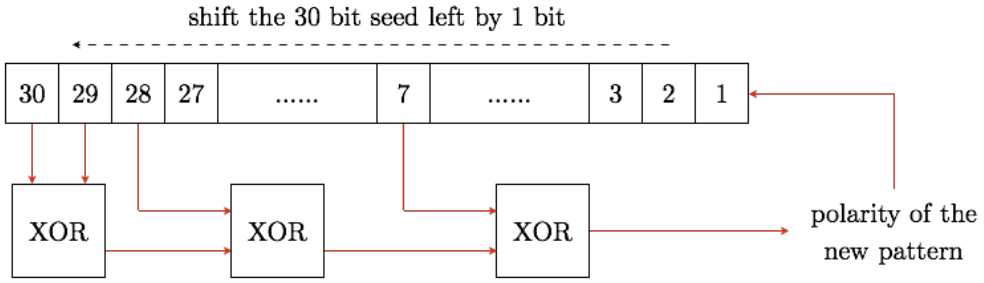


Figure 1: The 30-bit shift register in the Helicity Control Board. The polarity of a new pattern is calculated by applying an XOR (exclusive disjunction) operation to the bit 30, bit 29, bit 28 and bit 7 of a 30-bit register. Then the register is left-shifted by one bit and the new bit 1 is set by the XOR result. The repeat length of this generator is $2^{30} - 1 = 1,073,741,823$ bits.

is stable. To minimize the low frequency systematic uncertainty, the helicity signal always shows up in some symmetric multi-window patterns, like a double-window Pair or a four-window Quartet. For example, the helicity sequence in a Quartet pattern can either be $(+--+)$ or $(-++-)$ so any linear background is cancelled out. The helicity of the first window of each pattern is determined by a pseudo-random generator in the Helicity Control Board. The generator is a 30-bit shift register, the algorithm of which is shown in Figure 1. Any correlation between the helicity of the beam and other DAQ components is removed by using the pseudo-random generator. However, the helicity can still be predicted if a sequence of 30 helicity patterns is known since it is not real random. To minimize any other possible systematic effects, the helicity signal received by the experiment DAQ is delayed by 8 windows (adjustable in the Helicity Control Board). The actual helicity of a physics event is predicted with the Delayed Helicity signal offline by the same pseudo-random algorithm. Due to the non-zero response time of the Pockels Cell to the HV change, the helicity during the transition time between two helicity window is not stable. A $\mathrm{T_{settle}}$ signal is generated to deal with this problem. This signal is composed by a T_{settle} part and a T_{stable} part. $T_{\mathrm{settle}} + T_{\mathrm{stable}}$ equals to the time length of a helicity window and the T_{settle} is chosen to be slightly longer than the transition time of the Pockels Cell. Any data taken during the T_{settle} part is excluded from the helicity related analysis. Aside from the Delayed Helicity and the $\mathrm{T_{settle}}$ signal, the experiment DAQ also receives two more signals from the Helicity Control Board. The Pattern Sync signal indicates the start of a helicity pattern with a logic 1 and remains 0 in other helicity windows. The Pair Sync signal begins with a logic 1 at the first window of a helicity pattern and then toggles between 0 and 1. These two signals are useful to help predicting the actual helicity. Figure 2 is an example of these signals and their time sequence.

The helicity scheme generated by the Helicity Control Board can be various. The g2p experiment shares

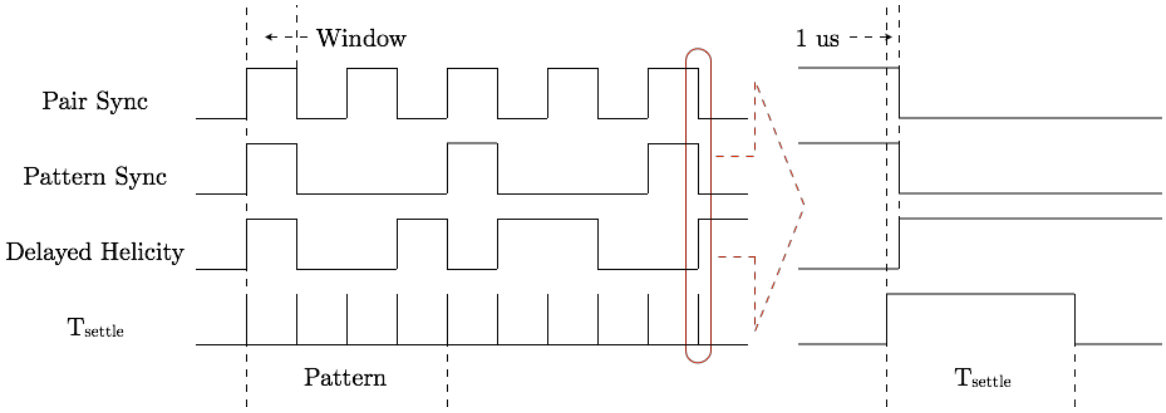


Figure 2: Helicity signals received by experiment DAQ. The right side shows the time sequence of these signals, notice that the T_{settle} signal is 1 μs prior to the other three to avoid misalignment.

the same helicity scheme with Hall C QWEAK experiment. During the QWEAK era, the helicity pattern is set to be Quartet. The T_{settle} and T_{stable} is set to 70 μs and 971.65 μs respectively so the helicity reversal rate is 960.02 Hz. The typical DAQ rate of g2p experiment is about $5 \sim 6$ kHz. The existed helicity decoder for the QWEAK helicity scheme in the Hall A analyzer (THaQWEAKHelicity) does not work at this DAQ rate. Thus a new helicity decoder is required for the g2p experiment.

2 DAQ Setup

To calculate asymmetry, each recorded event is sorted by the helicity of the electron beam. The number of accepted events in each helicity state is normalized by the total charge and the DAQ live time with the same helicity. Thus, the BCM signal, the trigger (T1$\sim$T8) and L1A signal of Hall A DAQ also need to be sorted by the beam helicity. In g2p experiment, these signals and the event stream are addressed as two different issuses.

The helicity signals described in section one are copied to three different electronics in g2p experiment. The helicity of the physics event is recorded by the trigger interface (TI). The TI has 12 state registers (TIR). The logical state of these registers is combined to two bytes and written to the raw data file whenever a trigger is received. The electronics setup is based on the note of R. Michaels [3] but is slightly different. Four registers (instead of the three in previous experiments) are used to record all of the four helicity signals respectively. The recorded helicity is referred as TIR helicity in the rest of this article. Besides, decoding the TIR helicity requires some timing information to determine whether the helicity sequence misses some

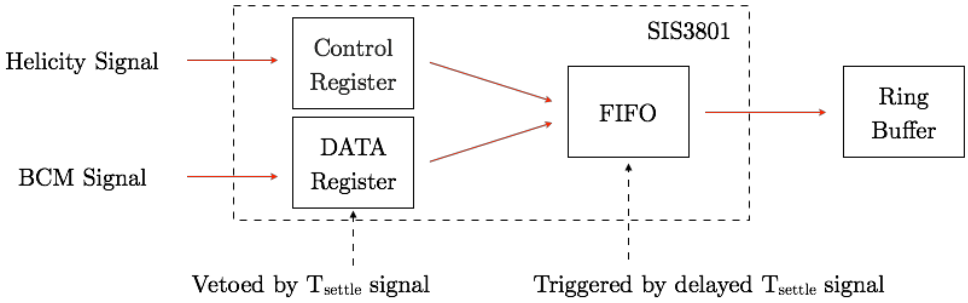


Figure 3: Workflow of a SIS3801 scaler. The data registers stops integrating for a TRUE level of T_{settle} signal. The same T_{settle} signals is delayed by 1.2 μs to trigger the FIFO to read the registers.

windows or not. So the standard 103.7 kHz fast clock signal is used to set a timestamp for each event written by CODA.

The triggers and L1A signals are pulse signals. The original outputs of BCM are analog signals but they are also converted to pulse signals. The helicity-gated SIS3801 scalers are used in the experiment to integrate these signals. Figure 3 shows the workflow of a SIS3801 scaler. The major components of SIS3801 are 32 data registers to do integration, 8 control registers and a FIFO (First-In-First-Out) data buffer. The T_{settle} signal is sent to one of the control registers to make a veto gate. The data registers only integrate in the T_{stable} part of a helicity window. A delayed T_{settle} signal is also sent to the control registers. Once the integration of one helicity window is finished, the delayed T_{settle} signal triggers the FIFO to read the integration results from the data registers and save them temporally. The delay time is carefully adjusted to ensure the integration has finished when FIFO reads the data registers. Two additional control registers are used for the Delayed Helicity and Pattern Sync signals. The FIFO also reads them and records the helicity state of each helicity window. However, the FIFO is not capable to store a large amount of data so a ring buffer, which is able to store the integration results of 1000 helicity windows, is set in the memory of the controller of the VME crate in which the scaler is installed. Thus, the scaler keeps integrating BCM and other signals and the ring buffer saves the results for an asynchronous readout. During the experiment, the ring buffer is set to be read by CODA every 50 events to reduce DAQ dead time. The CODA is also set to read the ring buffer whenever the ring buffer is filled by 50 helicity windows. The memory is cleared after each readout to prepare for new data. With SIS3801 scalers, a full sequence of helicity is saved as well as the helicity-gated integration results.

In addition to the SIS3801 scalers, the original analog BCM signals are also sent to the HAPPEX DAQ for helicity-gated integration. HAPPEX DAQ is installed to read the beamline electronics because of the

high resolution of its ADC [4, 5]. It is triggered by the helicity signal and it also uses a ring buffer to save data for readout. The helicity information recorded by the HAPPEX DAQ can be decoded in the same way of the SIS3801 scalers. This provides a cross-check of the scaler results. The helicity recorded by SIS3801 scalers and HAPPEX DAQ are referred as ring buffer helicity in the rest of this article.

3 Helicity Decoder

As mentioned in Section 1, The helicity written to the raw data file is delayed by 8 helicity windows. To get the actual helicity, the idea is to read 30 continuous helicity patterns to generate the random seed, and use this seed to predict the reported helicity, i.e. the delayed helicity, and the actual helicity afterward. The prediction is compared to the reported helicity of the first window of each pattern in the helicity sequence to make sure things are correct.

The previous helicity decoder is integrated into the Hall A analyzer, which has some benefits. However, since the ring buffer helicity is not saved event-by-event in the raw data file, it is easier to dump the helicity information out and do the prediction separately. Thus the new helicity decoder is written as a standalone package. The package contains 5 programs: "decode", "tir", "ring", "align" and "gen". Program "decode" reads the raw data file and dump the TIR and the ring buffer helicity information as well as the helicity-gated data like BCM into different ASCII files. Program "tir", "ring" and "align" are the major part to do the prediction of the actual helicity. The details of these 3 programs are described in section 3.1, 3.2 and 3.3 respectively. Program "gen" writes the decoded actual helicity and the helicity-gated data to the normal ROOT file generated by the Hall A analyzer. The usage of the package is also described in section 3.4.

3.1 Predict Actual Ring Buffer Helicity

The ring buffer saves a full sequence of the helicity as mentioned in Section 2. The sequence only breaks if no event written by CODA during the time period of 1000 helicity windows that in our case is about a second. It is because the capacity of the ring buffer is 1000 helicity windows, and the newly coming data flushes the old one out if CODA does not read the ring buffer to clear it. However, this is not likely to happen in this experiment because a 20 Hz pulse trigger (T8) is set for CODA to clear the buffer even when beam trips. This actually makes the decoding much easier. The random seed collected at the beginning can be used for

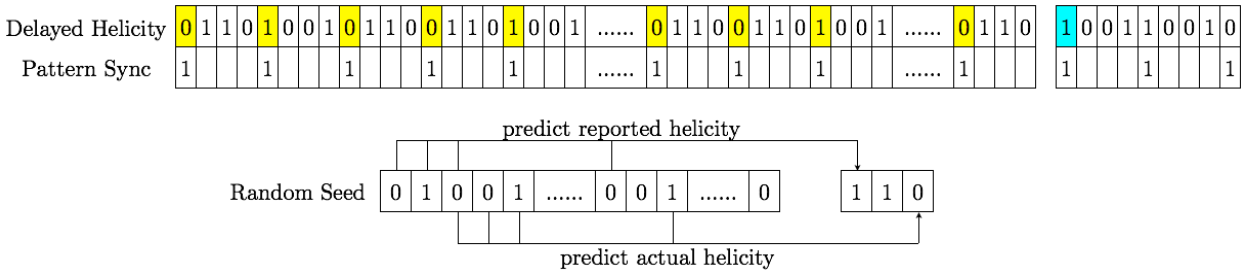


Figure 4: Predict actual ring buffer helicity. The windows marked as yellow are used to generate the random seed. The seed is used to predict the reported helicity and the actual helicity of the newly coming window (marked as cyan). In this example, the reported helicity of the cyan window is 1 but the actual helicity is 0.

the whole run.

The method of decoding the ring buffer helicity is shown in Figure 4. If the random seed is not set, the program reads the ring buffer helicity in sequence and selects out the windows of which Pattern Sync is 1. The helicities of these windows are appended to the random seed. The seed is used to predict the reported helicity and the actual helicity of the next helicity pattern once 30 bits is collected. The prediction algorithm is shown in Figure 1. For a Quartet pattern, only the helicity of the first window is directly predicted by the seed. The helicities of the second and the third window are always opposite to the first window and the helicity of the forth window is equal to the first one. After prediction of four windows in one pattern, the random seed is left-shifted by one bit and the new bit 1 is set with the reported helicity of the first window of the predicted pattern. The prediction is verified with the reported helicity sequence. If the prediction fails for any reason, each window in the pattern is marked as bad and the random seed is reset immediately and generate again.

The above method can not get the actual helicity of the windows used to generate the random seed. However, the actual helicity of these windows is saved in the helicity sequence. For the nth helicity window in the sequence, the actual helicity of this window is the reported helicity of the $(n+8)$th window. Thus a special procedure called backward prediction is added to the decoder. Mathematically the pseudo-random algorithm in Figure 1 can be approved equivalent to a 30-bit right-shift register which replaces the new bit 30 with the exclusive disjunction of bit 1, bit 8, bit 29 and the old bit 30. The backward prediction uses this right-shift register to predict the reported helicity from the last window in the helicity sequence to the first one. If the prediction of the nth window does not fail, the actual helicity is assigned by the reported helicity of the $(n+8)$th window. It covers those windows used to generate the random seed of the normal prediction and also provides a cross-check for the whole helicity sequence.

In Ring Buffer

Helicity		0	1	1	0	1	0	0	1
Helicity-gated Data	105	100	100	105	100	105	105	100	105

Actual Sequence

Helicity	0	1	1	0	1	0	0	1	
Helicity-gated Data	105	100	100	105	100	105	105	100	105

Figure 5: Misalignment between control registers and data registers of SIS3801 scalers.

During the test, it is found that the control registers of the SIS3801 scalers are delayed by exactly one window comparing to the data registers. As shown in the Figure 5, the helicity-gated data (like BCM) is one window ahead of the correlated helicity states. This is corrected in the program. The registers of HAPPEX DAQ also have this issue, but in this case it is the data registers delayed by one window.

3.2 Predict Actual TIR Helicity

For TIR helicity, the decoding algorithm is still based on the prediction method. However, it is possible that several CODA events are saved in one helicity window, or no CODA event is saved during several helicity windows. Figure 6 shows an example of TIR helicity. It does not show any obvious pattern, which makes the decoding more difficult. It is critical to find a method to locate each event in the helicity sequence before any prediction can be proceed. Since the helicity windows all have identical time length, it is possible to identify each CODA event in the helicity sequence if they are labeled by some kind of timestamp. Thus, a 103.9 kHz fast clock signal is used to set the timestamp during the experiment. The clock signal is integrated by an ungated scaler (which means it is not helicity gated), and read by CODA for each event. The helicity reversal frequency is 960.02 Hz in the experiment, so the time length of each window is about $103900 \div 960.02 \approx 108.2$ scaler counts. Some of the tir event may be saved during the T_{settle} part of a helicity window. These events is excluded from the decoding process and marked as "unstable" by the decoder.

The first step to decode the TIR helicity is still to generate the random seed. For convenience, the helicity windows of which Pattern Sync is 1 are referred as Pattern Sync windows in the rest of this article. Any events in these Pattern Sync windows are also referred as Pattern Sync events. The helicity of Pattern Sync events is used to generate the random seed, however, there are 3 different situations for the TIR helicity:

1. The event is the first event of a Pattern Sync window, and no Pattern Sync window is missed before this event. In this case, the helicity of this event should be appended to the random seed.

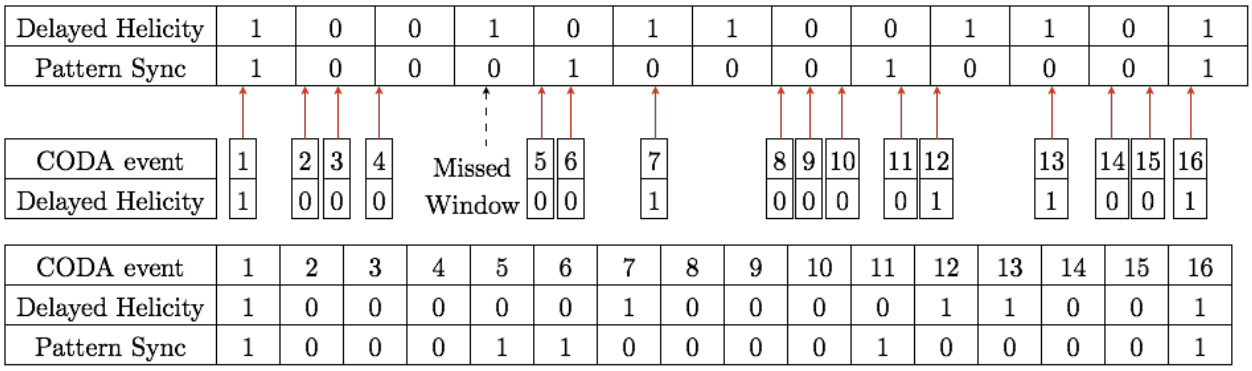


Figure 6: TIR helicity example. The sequence on the top is the normal helicity sequence. The CODA event stream at the bottom shows how the TIR helicity breaks the normal helicity sequence.

2. The event is the second (or third, ...) event of a Pattern Sync window. In this case it is ignored because the first event of this window has been appended to the random seed.

3. One or more Pattern Sync windows are missed before this event. In this case, the existed bits of the random seed is reset.

In the decoder, the time interval $\Delta T_{\text{pattern}}$ is calculated to determine these 3 situations. Assuming the timestamp of the current event is T and the previous Pattern Sync event is $T^{\text{last}}_{\text{pattern}}$, $\Delta T_{\text{pattern}}$ can be expressed as $\Delta T_{\text{pattern}} = T - T^{\text{last}}_{\text{pattern}}$. Figure 7 shows the restrictions on $\Delta T_{\text{pattern}}$ for these 3 situations. If $\Delta T_{\text{pattern}} < 2\,T_{\text{w}}$, the previous Pattern Sync event and the new one are in the same window, and it is case 1 described above. If $\Delta T_{\text{pattern}} > 6\,T_{\text{w}}$, at least one Pattern Sync window is missed, and this is case 3. Notice that the possible value of $\Delta T_{\text{pattern}}$ are $0 \sim 1\,T_{\text{w}}$, $3 \sim 5\,T_{\text{w}}$, $7 \sim 9\,T_{\text{w}}$ or etc. The restrictions are chosen to be just inbetween two ranges to avoid any possible error due to the scaler fluctuation.

If the ring buffer helicity is decoded when the TIR helicity is processing, it is possible to use ring buffer helicity to reduce the amount of Pattern Sync events used to generate the random seed. When the length of the random seed exceeds 10 bits, the existed bits is used to run a pattern search among the saved random

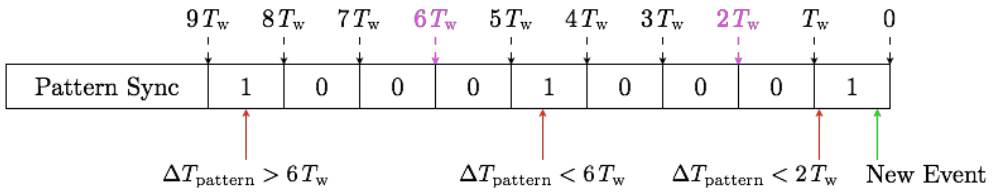


Figure 7: Restrictions on $\Delta T_{\text{pattern}}$ to determine whether a Pattern Sync window is missed or not if the new event is a Pattern Sync event. Here T_{w} is the time length of a helicity window in the unit of scaler counts. The red arrows are different possibilities of the previous event.

seeds of the ring buffer helicity. If a matched seed is found during the search, it is considered as a candidate of the random seed for TIR helicity. To test it, this seed is used to make several predictions. If all of the predictions matches the real helicity sequence, it is used as the random seed for TIR helicity.

Once the random seed is generated, the second step is to predict the reported and actual helicity for each event with the seed. Since the random seed need to be left-shifted by 1 bit whenever a helicity pattern is finished, it is critical to determine n, the number of missed Pattern Sync windows. Besides $\Delta T_{\text{pattern}}$, another time interval ΔT is used to determine n. Assuming the timestamp of the previous event is T^{last}, ΔT can be expressed as $\Delta T = T - T^{\text{last}}$. The random seed is left-shifted by n bits before the prediction. When shifting, the new bits are always calculated with the algorithm in Figure 1. Due to the particularity of the Pattern Sync event, three different situations need to be considered separately:

1. Both the new event and its previous event are Pattern Sync events. In this case, the restrictions on $\Delta T_{\text{pattern}}$ shown in Figure 7 still works. If $\Delta T_{\text{pattern}} < 2\,T_{\text{w}}$, the new event is in the same window of the previous one. The same random seed is used to predict the actual helicity. If $(4 \times n + 2)\,T_{\text{w}} < \Delta T_{\text{pattern}} < (4 \times n + 6)\,T_{\text{w}}$, n Pattern Sync windows are missed (n can be 0). After prediction, the seed is left-shifted by 1 bits to prepare for next prediction.

2. The new event is a Pattern Sync event but its previous event is not one. In this case, the time interval ΔT is used to determine the number of missed Pattern Sync windows. The integer part of $\Delta T/(4\,T_{\text{w}})$ is n. After prediction, the seed is left-shifted by 1 bits to prepare for next prediction.

3. The new event is not a Pattern Sync event. In this case, the time interval $\Delta T_{\text{pattern}}$ is used to determine the number of missed Pattern Sync windows. The integer part of $\Delta T_{\text{pattern}}/(4\,T_{\text{w}})$ is n. The prediction only tells the actual helicity of the first window in this Quartet pattern. The actual helicity of this particular event can be found in Table 1 according to its Pair Sync and reported helicity value. Unlike case 1 and 2, the seed does not need to be left-shifted after prediction, but the $T^{\text{last}}_{\text{pattern}}$ need to be increased by $n \times 4\,T_{\text{w}}$ in case the next event is still not a Pattern Sync event.

Prediction of the First Window is +			Prediction of the First Window is −		
Reported Helicity	Pair Sync	Actual Helicity	Reported Helicity	Pair Sync	Actual Helicity
1	1	+	0	1	−
0	0	−	1	0	+
0	1	−	1	1	+
1	0	+	0	0	−

Table 1: Find the actual helicity of a event according to its reported helicity and Pair Sync value.

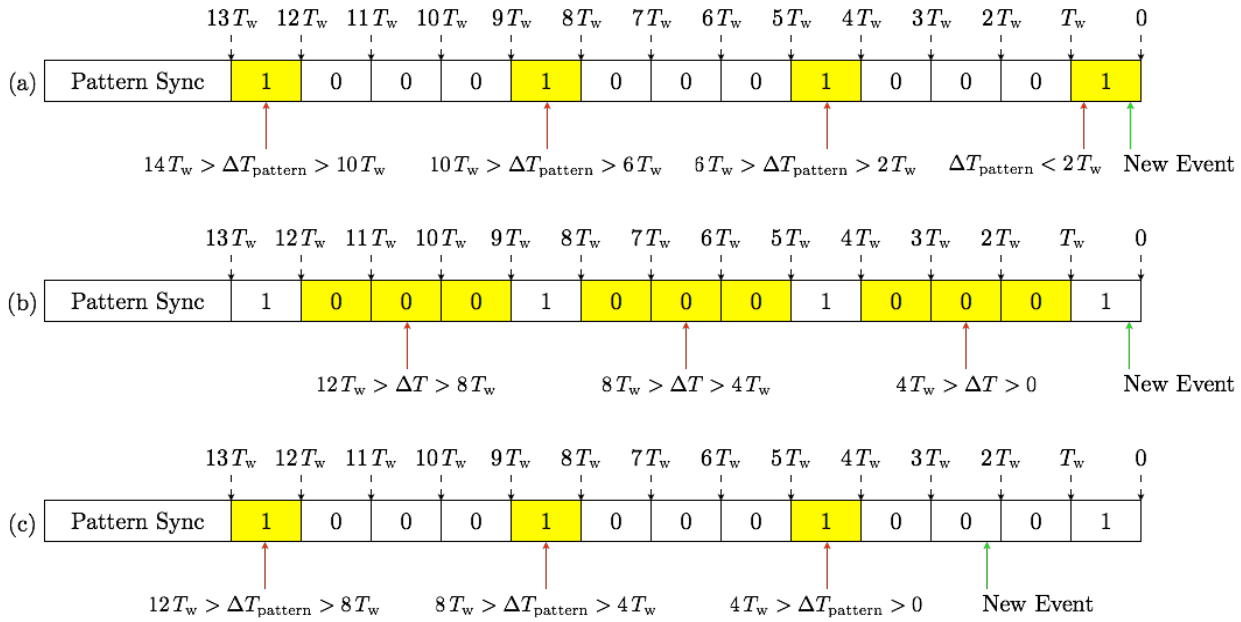


Figure 8: Restrictions to determine the number of missed Pattern Sync windows: (a) Both the new event and its previous event are Pattern Sync events; (b) The new event is a Pattern Sync event but its previous event is not one; (c) The new event is not a Pattern Sync event.

Figure 8 shows how to set restrictions on ΔT and $\Delta T_{\text{pattern}}$ to determine n for these 3 situations. The prediction of the reported helicity of each event is checked with the reported helicity written in the raw data file. If the prediction fails, any events in this pattern are marked as bad. The random seed is reset and regenerated.

Due to the fluctuation of the scaler, the time interval ΔT and $\Delta T_{\text{pattern}}$ may not satisfy the restrictions described above sometimes. The excluded T_{settle} events is used to calibrate T^{last} and $T^{\text{last}}_{\text{pattern}}$ since the time length of helicity windows is quite accurate. As shown in Figure 9, a T_{settle} event which is right before a Pattern Sync window is selected out for calibration. Assuming the timestamp of this T_{settle} event is T, the T^{last} is set to $T - T_{\text{w}}$ and the $T^{\text{last}}_{\text{pattern}}$ is set to $T - 3\,T_{\text{w}}$. Once calibrated, T^{last} and $T^{\text{last}}_{\text{pattern}}$ are not used to

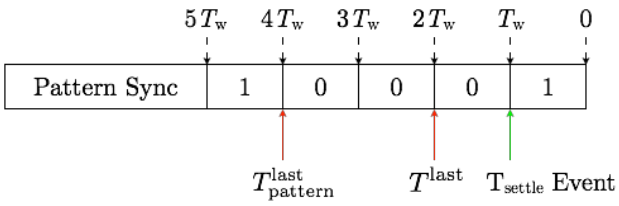


Figure 9: Calibrate T^{last} and $T^{\text{last}}_{\text{pattern}}$ with T_{settle} events.

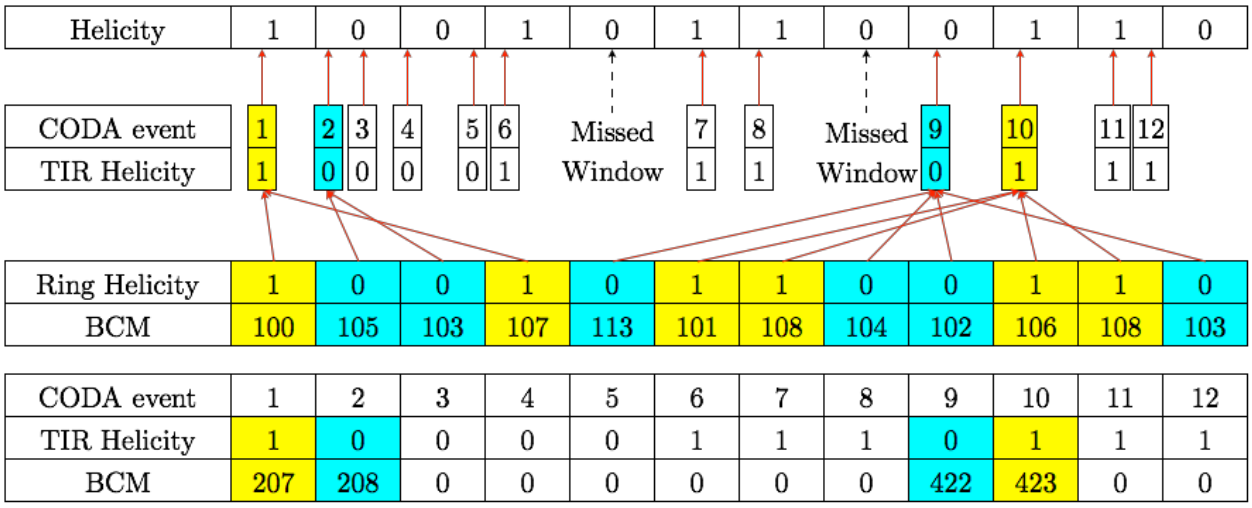
store the time stamp of previous events any more. The values of T^{last} and $T^{\text{last}}_{\text{pattern}}$ are increased by $n \times 4\,T_{\text{w}}$ if n patterns are finished during the prediction. And any qualified T_{settle} events are used to recalibrate their values. The fluctuation of ΔT and $\Delta T_{\text{pattern}}$ is reduced by half at least if calculated with calibrated T^{last} and $T^{\text{last}}_{\text{pattern}}$ and hence the number of missed Pattern Sync windows can be determined more accurately.

3.3 Align TIR Helicity with Ring Buffer Helicity

The purpose to align TIR helicity with ring buffer helicity is to insert the helicity-gated data into the CODA data stream since the helicity-gated data is only saved with the ring buffer helicity. As mentioned in Section 1, the helicity random seed never repeats during one particular run, so it can be used as "fingerprint" to do this alignment.

Before alignment, the quality of the prediction result is checked to avoid false asymmetry. If the actual helicity of a event is not predictable due to some error, events in the same helicity pattern are also marked as bad during the checking. For the ring buffer helicity, the BCM information is used to determine beam trips. The data taken during the beam trip and within 30 helicity patterns before or after the beam trip is excluded from the data analysis to prevent any systematic error.

Figure 10 shows an example of the alignment. Here BCM is selected as an example of the helicity-gated



CODA event	1	2	3	4	5	6	7	8	9	10	11	12
TIR Helicity	1	0	0	0	0	1	1	1	0	1	1	1
BCM	207	208	0	0	0	0	0	0	422	423	0	0

Figure 10: Align TIR helicity with ring buffer helicity. Take BCM as an example of the helicity-gated data. The second pattern in the helicity sequence missed two 0 helicity windows, so the BCM values of this pattern is added to the values of next pattern to be saved in the CODA event stream.

data. For each helicity pattern in the ring buffer helicity, two BCM values C_+ and C_- are calculated for + and − helicity. The random seed saved for this pattern is compared with all the random seeds saved in the tir helicity. If a matching pattern is found and the pattern contains at least one event with + helicity and one event with − helicity, C_+ is saved to the first event with + helicity and C_- is saved to the first event with − helicity. If no pattern matches, C_+ and C_- is added to the BCM values of the next helicity pattern in the ring buffer helicity. This method keeps the asymmetry and the sum of each helicity-gated data unchanged in the CODA data stream so they can be used in helicity-related calculation.

3.4 Usage of the Package

The programs in the package are standalone executables. The library "libconfig" is required to compile and execute the programs. "decode" and "gen" also requires ROOT and Hall A analyzer libraries. All the parameters used in the programs can be adjusted in a configuration file. The "libconfig" library is used to parse this configuration file so the content of the file must obey the grammar of "libconfig". The package already provides two configuration files for left and right HRS. The programs can also read some parameters from command line, such as the input and output directories. A list of acceptable parameters can be obtained by executing the program with parameter "-h". The package also provides a shell script "helicity.sh" to execute these 5 programs automatically in the proper sequence.

The actual helicity and the helicity-gated data are written to the corresponding root files of the same run if the files exist. The TIR helicity is written to the existed "T" tree as several new leaves. The ring buffer helicity is written as a new tree "hel_ring" for SIS3801 scalers or "hel_happ" for HAPPEX DAQ. Table 2

Name	Meaning
hel.L.hel_rep	reported Delayed Helicity, 1 means +, 0 means −
hel.L.hel_act	actual helicity, 1 means +, −1 means −
hel.L.qrt	Pattern Sync
hel.L.pairsync	Pair Sync
hel.L.mps	T_{settle}
hel.L.timestamp	TIR timestamp
hel.L.seed	saved TIR random seed
hel.L.error	error code
hel.L.*	helicity-gated data inserted from ring buffer, * can be time, bcmup, bcmdown, L1A, T3, T4 (or T1, T2 for RHRS)

Table 2: List of TIR helicity variables inserted in the "T" tree. For right HRS, replace "L" with "R".

Name	Meaning
hel.L.evnum	CODA event number which contains the raw data of this ring buffer window
hel.L.hel_rep	reported Delayed Helicity, 1 means +, 0 means −
hel.L.hel_act	actual helicity, 1 means +, −1 means −
hel.L.qrt	Pattern Sync
hel.L.seed	saved ring buffer random seed
hel.L.error	error code
hel.L.*	helicity-gated data, * can be time, bcmup, bcmdown, L1A, T3, T4 (or T1, T2 for RHRS), heldata

Table 3: List of ring buffer helicity variables inserted in the "hel_ring" tree or "hel_happ" tree. For right HRS, replace "L" with "R".

and 3 are full lists of the variables written to the root files and their definitions.

The datas which should be excluded from the helicity-related analysis is marked with some error code during the prediction and the alignment. The error code is a binary number. Each bit of this number represents a different kind of failure during the decoding procedure. 0x000F, 0x00F0, 0x0F00 are reserved for TIR helicity, SIS3801 helicity and HAPPEX helicity respectively. 0xF000 are reserved for errors during alignment. The meaning of each bit is shown in the Table 4. For example, if the helicity-gated charge is calculated with the BCM information in the SIS3801 scaler ring buffer, the cut " Error&0x10FF == 0" should be used to select proper events. If the BCM information in HAPPEX DAQ is used, the cut should be changed to " Error&0x2F0F == 0".

Error code	Meaning
0x0001	event used to generate TIR random seed
0x0002	prediction does not match the reported TIR helicity
0x0004	TIR event taken during a beam trip
0x0008	T_{settle} event
0x0010	event used to generate scaler ring buffer random seed
0x0020	prediction does not match the reported ring buffer helicity
0x0040	ring buffer event taken during a beam trip
0x0F00	only used for HAPPEX helicity, the meaning is same as 0x00F0
0x1000	failed to find a corresponding pattern in scaler ring buffer helicity
0x2000	failed to find a corresponding pattern in HAPPEX helicity

Table 4: Meanings of the error codes.

DataSet	Left HRS	Right HRS	LHAPPEX	RHAPPEX	Moller	Hall C
1	-0.91%	-0.91%	-0.92%	-0.91%	-0.92%	-0.91%
2	-0.56%	-0.56%	-0.56%	-0.56%	-0.56%	-0.56%
3	-0.092%	-0.095%	-0.097%	-0.097%	-0.090%	-0.094%

Table 5: Beam charge asymmetry calculated with different DAQs

4 Test with Charge Asymmetry

The helicity decoder is tested with beam charge asymmetry during the experiment. The beam charge asymmetry A_Q can be expressed as

$$A_Q = \frac{Q_+ - Q_-}{Q_+ + Q_-} \tag{1}$$

Here $Q_\pm$ are the beam charge with helicity ± 1. The beam charge asymmetry can be adjusted in the injector. For the test, beam with large charge asymmetry is required from the injector and is measured with HRS DAQ (SIS3801 scaler), HAPPEX DAQ, Moller DAQ and Hall C DAQ. The Moller DAQ and Hall C DAQ is used as reference of this test. The results of the test are listed in the Table 5. The calculation result of the new helicity decoder agrees with the Moller DAQ and Hall C DAQ. To conclude, the helicity decoder is able to be used in helicity related analysis.

References

[1] R. Flood, S. Higgins, and R. Suleiman. *Helicity Control Board User's Guide.* Jefferson Lab, 2010.

[2] J. Hansknecht. *Injector Laser Table Element Definitions.* Jefferson Lab, 2007.

[3] R. Michaels. *Schematic of Electronics for Helicity Info in Hall A HRS during QWEAK.* Jefferson Lab, 2010.

[4] R. Michaels. *Precision Integrating HAPPEX ADC.* Jefferson Lab, 2003.

[5] B. Hahn. *Tests of the PREX 18-bit ADCs.* Jefferson Lab, 2008.