Helicity Decoder Module

Ed Jastrzembski
Jeff Wilson
FE-DAQ Group
2/12/2025

Contents

Introduction

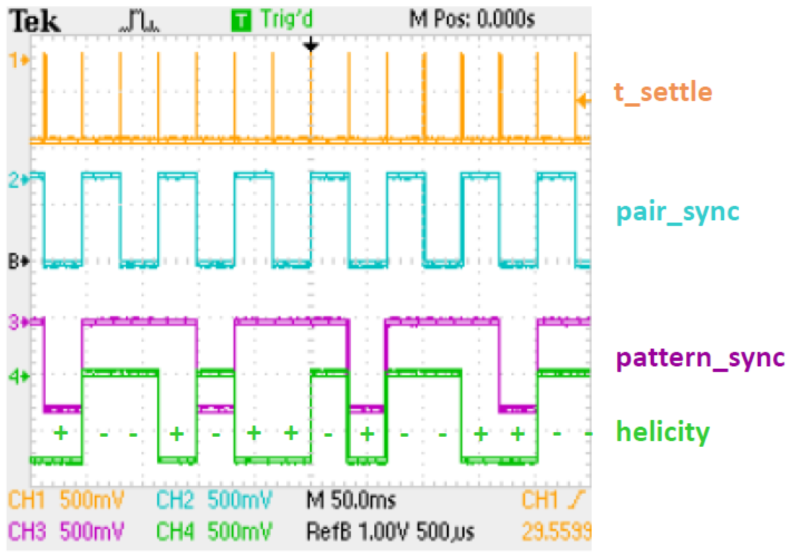
The function of the Helicity Decoder is to record information necessary to determine the helicity of the incoming electron for the recorded event. The source of that information is the Helicity Control board located in the CEBAF injector [1][2][3]. The Control board is sometimes referred to as the Helicity Generator board.

The Helicity Control board is used to control the high-voltage of the Pockels Cell on the Laser Table in the CEBAF injector. A polarity change in Pockels Cell HV changes the circular polarization of laser light which in turn changes the direction of spin of the photo-emitted electrons relative to its momentum. The electron spin is either aligned parallel (+) or anti-parallel (-) to the electron momentum; this is called electron helicity. The electron's original helicity state is maintained throughout the CEBAF acceleration process (90% polarization).

Because the effects of the electron's helicity on its interaction with matter is a very subtle one, serious effort has been made to control collection of data in both helicity states in a way that minimizes systematic effects from introducing errors. The Helicity Control board changes the helicity of the electrons over time according to a set of programmable complementary patterns (e.g. + - - +, - + + -). The initial helicity of a pattern is determined by a pseudo-random generator. This serves to remove correlations between the helicity of the beam and any other device of the accelerator or in the experimental halls. Furthermore, the helicity state reported by the Control board can be delayed from the actual electron helicity state. In this way no devices of the beamline or in the halls know what the real time helicity is. When analyzing data the helicity is adjusted to obtain the true helicity of the electron for the event. (See details in [2].)

Figure 1 shows the set of signals that the Helicity Control board sends to the Helicity Decoder modules located in the experimental halls. A Quartet helicity sequence is shown here.

Figure 1



Quartet Pattern (signals asserted low)

Signal *t_settle* is asserted low when the helicity of the beam is stable – i.e. the helicity has *settled* into one of its two possible states. The signal is more descriptively known as *t_stable*. Time intervals of stable helicity are called helicity windows. Whenever *t_settle* is high the helicity *may* be changing. Events recorded during these *settling* periods cannot be assigned a definite helicity and so should be eliminated from consideration in any analysis whose aim is to compare results from electrons in the two helicity states.

Signal *pattern_sync* asserted low marks the beginning of a helicity pattern. In this example a Quartet helicity pattern is generated. The Quartet pattern consists of four helicity windows. There are two distinct complementary patterns available for the Quartet: **+ - - +** and **- + + -** . The choice of pattern is based on the value from a pseudo-random generator in the Control board. The position of a window in the pattern is called its phase (0, 1, 2, 3 for the Quartet). Three complete Quartet sequences are shown in **Figure 1**.

Other helicity patterns (Toggle, Pair, Octet, Hexo-quad, Octo-quad, …) can be driven from the Helicity Control board [2][3]. Each of these (except Toggle) defines a pair of complementary patterns of length N helicity windows (Pair: $N = 2$, Octet: $N = 8$, Hexo: $N = 24$, Octo: $N = 32$,…).

Signal *helicity* is the reported helicity of the electrons (+, -) for the window. The reporting delay of the Control board is programmed in units of helicity windows. Only when the reporting delay of the Control board is programmed to zero does the reported helicity correspond to the real time helicity of the electrons.

Signal *pair_sync* toggles with advancing helicity window and may be used as a diagnostic signal. When the Pair helicity pattern is selected the signals *pair_sync* and *pattern_sync* are identical.

Pseudo-random Helicity Generator

A 30-bit shift register in the Helicity Control board is used to generate the pseudo-random bits 0, 1 (or +, -) which determines the first window of a pattern (**Figure 2**). This removes any correlation between the helicity of the beam and any other device in the accelerator or in the hall. It is pseudo-random because it is deterministic. For any 30-bit value (seed) loaded into the register $2^{30} – 1$ bits (1,073,741,823) will be output before the entire sequence repeats.

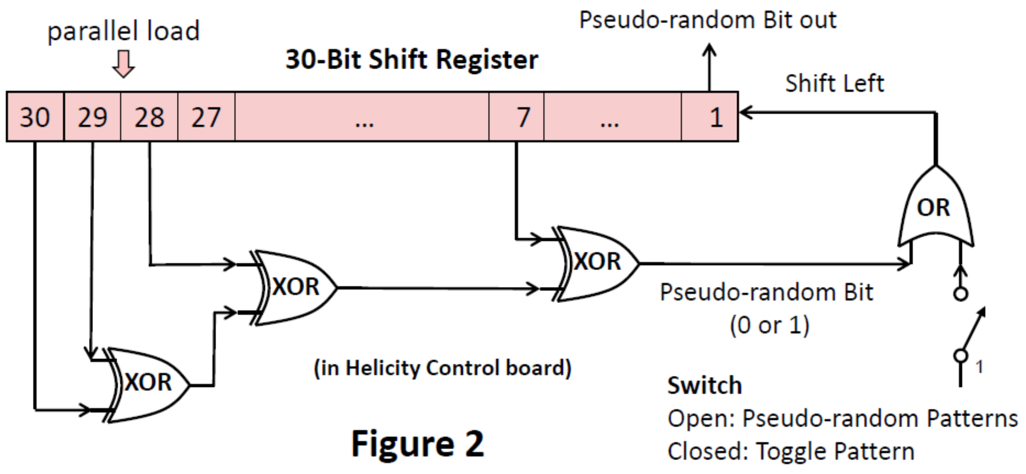


Figure 2

Consider the case when 30 consecutive bits from the output of the Control board shift register are recorded by a device (e.g. Helicity Decoder). This corresponds to measuring the reported helicity of the *first* window (phase 0) of 30 consecutive helicity patterns. The 30-bit value can serve as the *current seed* in an equivalent generator (manifested in software) so that all future bits from the Control board shift register can be predicted.

In this way the actual helicity of the electron for an event can be determined off line from a knowledge of the current seed, the pattern generated, the pattern phase for the event, and the reporting delay programmed in the Control board.

It is important to note that for the Decoder to correctly capture the *current seed* of the Generator the programmed reporting delay must be an integral number of helicity *patterns*. The Decoder assumes that the first window of a helicity pattern is always associated with the assertion of signal *pattern_sync*. Thus for the 4 window Quartet pattern only reporting delays of 0, 4, 8, … can be used.

Helicity Decoder I/O

The four helicity signals (*t_settle, pair_sync, pattern_sync, helicity*) are transmitted from the Control board to the Decoder module over multimode fiber optic cables. The AFBR-2418Z receivers of the Decoder are compatible with the HFBR-1412Z transmitters (820 nm) of the Control board and any intermediate fiber transmitters in fan out modules. The Decoder can instead accept the four helicity signals as NIM levels on standard LEMO connectors.

Data acquisition control signals (*clock, trigger, sync_reset*) can be accepted by two independent front panel ports (FP1, FP2). The connector and LVDS signal levels on port FP1 are compatible with the FADC250 Front-Panel Signal Distribution board. Port FP2 uses standard differential ECL signal levels on dual row 0.1” pin headers. See the Decoder front panel drawing in **Figure 3**. **Table 1** shows pin assignments for the signals.

The Decoder can also accept the control signals through the VXS port on the backplane. The VXS signal mapping is the standard one used for all VXS data acquisition boards at JLab.

The required input *clock* frequency for the Decoder is 250 MHz. This frequency is also used by the FADC250 boards and is readily available throughout the data acquisition system of JLab experiments. An internal 250 MHz *clock* signal on the Decoder may be used instead.

For standalone Decoder applications the internal *clock* signal and internally generated *sync_reset* signal may be used. The internal *sync_reset* signal is generated by writing a register on the Decoder. The assertion of *sync_reset* (external or internal) initializes the trigger time-stamp counter to zero.

An internal *trigger* signal can be generated by writing a register on the Decoder. This may be used for testing purposes.

Source selection for each of the control signals is done independently.

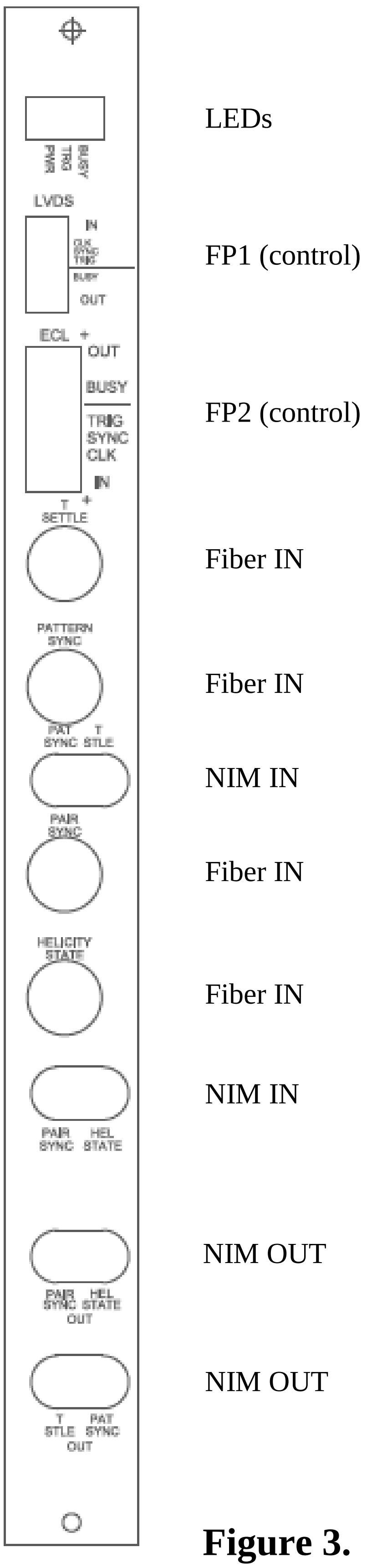
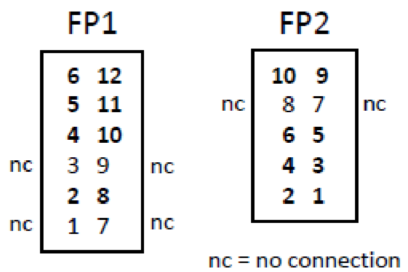


Figure 3.

Table 1.

signal	FP1 pin #	FP2 pin #
clock_p	6	1
clock_n	12	2
sync_p	5	3
sync_n	11	4
trig_p	4	5
trig_n	10	6
busy_p	2	9
busy_n	8	10



nc = no connection

Data acquisition output control signal *busy* is driven out of FP1, FP2, and the VXS interface with the appropriate signal levels for these ports. An asserted *busy* signal tells the acquisition system that the storage limit of the module is being approached. The system can respond by temporarily throttling back triggers to avoid any loss of data.

Helicity Data Path

The helicity data path in the Decoder is shown in **Figure 4**. The source of the four external helicity signals can be selected as Fiber or NIM. For flexibility the input signals can be inverted as a group. Optional filtering can be applied to suppress noise pulses. A digital delay can be programmed to account for the difference between the propagation time of the accelerated electrons from injector to hall and the propagation time of the helicity signals from Control board to the hall.

An internal helicity pattern generator has been implemented in the Decoder for test purposes.

The external or internally generated helicity data can be chosen for processing. The data to be processed may be delayed digitally to account for the trigger formation time (trigger latency). This insures that the Decoder is viewing the helicity signals at the time of the interaction when a trigger initiates event recording.

Helicity data is driven off the board as NIM signals. For flexibility the outputs can be inverted as a group. The output signals can be chosen to be the internally generated helicity signals, or the pre or post trigger latency helicity signals. Driving the external helicity data off the board allows a second independent device to record the helicity data for redundancy as well as allowing the helicity signals to be observed on an oscilloscope.

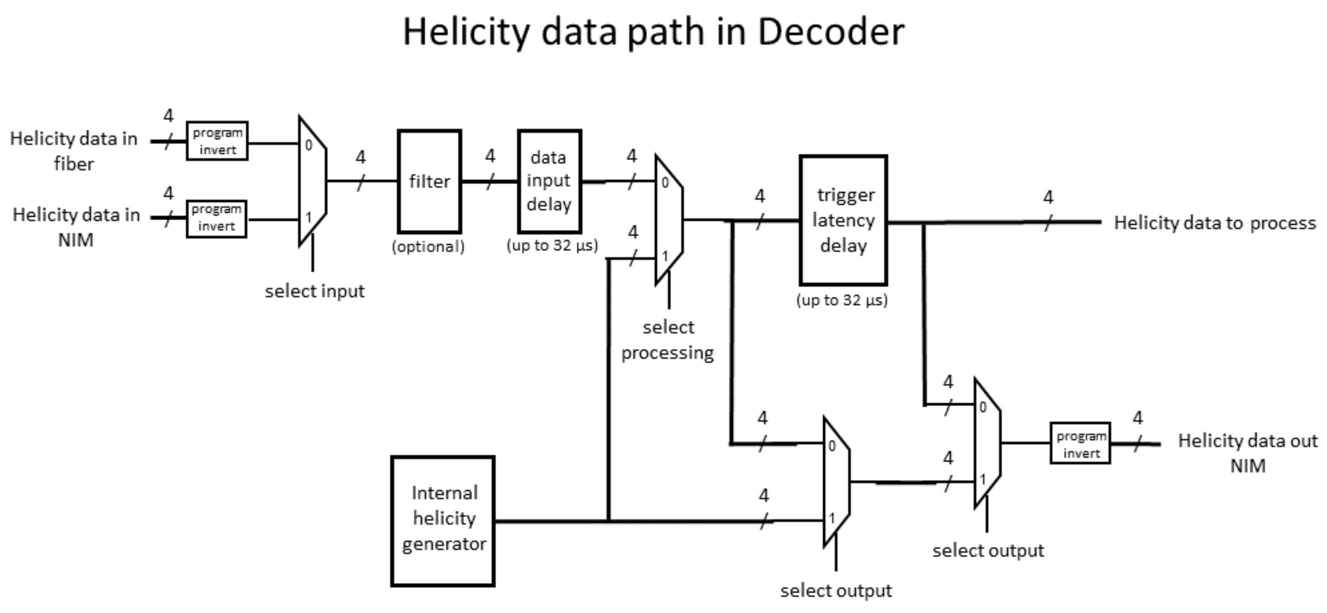


Figure 4.

Decoder Measurements

The conceptual specifications for the Helicity Decoder are found in Reference 4. Requested measurements are listed below. Many of these support consistency checks to confirm that the module is behaving correctly. (See the Data Format section in Appendix 2 for more details.) (Note: The *t_settle* signal is referred to as *t_stable* below and in Appendix 2.)

- Current seed of generator from past 30 *helicity* values at pattern start

- Count of falling edge *t_stable*

- Count of rising edge *t_stable*

- Count of *pattern_sync*

- Count of *pair_sync*

- Time of trigger from start of *t_stable* interval

- Time of trigger from end of *t_stable* interval

- Time duration of last complete *t_stable* interval

- Time duration of last complete *settling* interval (*t_stable* not asserted)

- Status at *trigger* time
  – *t_stable* state (live)
  – *pattern_sync* (live)
  – *pair_sync* state (live)
  – *helicity* state (live)
  – *helicity* state at pattern start (latched at *t_stable* start)
  – Event Polarity (XOR [*helicity*, *helicity* at pattern start])
  – Pattern phase count

- Last 32 windows of *pattern_sync*

- Last 32 windows of *pair_sync*

- Last 32 windows of *helicity*

- Last 32 values of *helicity* at *pattern_sync*

Data Acquisition and Module Features

The Helicity Decoder is a VMEbus Switched Serial (VXS) module [5]. When operating in a VXS crate it can use the standard JLab control signal distribution system. The module will also function in a VME64x crate when front-panel control signals are used. Earlier VME backplanes (with 3-row P1, P2 connectors) cannot be used as they do not provide the +3.3V power required by the module. A picture of the Decoder module is shown in **Figure 5**. More details about the board are found in Appendix 1.

The module is a high-performance data acquisition board with a specialized data payload. It is designed to work in the synchronous data acquisition systems of the halls (global *clock*). It can also be deployed in stand-alone mode using an internally generated *clock* signal. The module can accept and process data from consecutive triggers that are separated by 50 ns. Processed data is stored in memory for later readout. Readout of earlier events can occur simultaneously with the processing of new events. Dead time is negligible. The maximum data readout rate over the VMEbus is 200 MB/s (2eSST).

The module supports readout of event blocks. Rather than reading out data when each trigger is received, the Data Acquisition system configures modules to wait for a large number (< 256) of event fragments to accumulate in the module memory before readout is initiated. Reading out data in event blocks enhances readout efficiency. The Decoder reports 18 words (32-bits each) for each trigger. The 64K word on-board storage (FPGA memory) allows for 14 maximum size blocks to be stored on board (3570 events).

The time of arrival of the trigger is recorded and included in the data for the event. This is done so that the read out controller can check that the event fragments it collects from the modules are from the same event. The global system *clock* (250 MHz) and *sync_reset* signals are accepted by the module. At the beginning of a data run *sync_reset* is simultaneously sent to all modules in the system. This initializes the trigger time-stamp counter on all modules to zero. The 250 MHz *clock* is divided down to 125 MHz in the FPGA of the Decoder. All time measurements (*trigger,* helicity data) are thus measured to 8 ns precision.

The user can program the module to delay the helicity input signals by the trigger latency. The trigger comes later in time than the electron interaction in the target. We want to use and record the states of the helicity signals at the time of the interaction, not at the time of the trigger's arrival to the module.

An option exists to apply digital filtering on external helicity signals. A short duration noise pulse on *t_settle* can be mistakenly interpreted as the boundary of a helicity window and thus ruin the computation of the current seed of the Control board helicity generator. (See register CTRL_1 in Appendix 1.) After a glitch pulse thirty consecutive good patterns must occur before the seed is valid again. The available filtering levels and their minimum pulse durations are shown in **Table 2**. The filtering incurs a delay in the helicity data path equal to the minimum pulse duration of the selected filter level. The filtering is applied to all four helicity signals so that no skew is introduced between *t_settle* and the other signals. The total input data delay is the sum of the programmed data input delay and the filter delay.

Table 2.

Filter Level (register CTRL_1)	Minimum pulse duration = filter delay (ns)
0	0
1	32
2	64
3	128
4	256
5	512
6	1024
7	2048

Referring to **Figure 4** the total IN-to-OUT delay of the helicity signals (measured from the NIM inputs) can be approximated by:

Output before Trigger Latency:

DELAY (ns) = 64 + $(N_D - 1) * 8$ + filter_delay
$N_D = 1, 2, 3,\ldots, 4095$ = data delay count (register 0x1C)

Output after Trigger Latency:

DELAY (ns) = 88 + $(N_D + N_L - 2) * 8$ + filter_delay
$N_D = 1, 2, 3,\ldots, 4095$ = data delay count (register 0x1C)
$N_L = 1, 2, 3,\ldots, 4095$ = trigger latency count (register 0x1C)

Firmware for the Helicity Decoder may be downloaded into the EEPROM through VME access of the module’s registers (see the CONFIGURATION CSR & ADR/DATA register pair in Appendix 1.) A power cycle is required to load the new firmware image into the FPGA.

Testing the Helicity Control board using the Helicity Decoder

The Decoder can be run in a *test* mode that is useful to confirm the operation and stability of the Helicity Control board. In this mode an internal test trigger is generated at the beginning of each helicity window (*t_settle* assertion). Thus one can record and analyze the entire Control board helicity sequence over any time period. (See register CTRL_1 in Appendix 1.)

The helicity reporting delay of the Control board can be tested with the Decoder operating in *test* mode. For this test the *pair_sync* input signal to the Decoder is replaced by the (non-delayed) *helicity_flip* signal from the Control board. The Decoder delays this signal by a programmable number of helicity windows (just as the Control board does). If this delay is equal to the reporting delay of the Control board, the *pair_sync* and *helicity_state* signals should be identical. When the delay is enabled a counter is incremented whenever the signals do not agree at the time of their latching (beginning of each helicity window). The error count is available in a register. (See the Helicity Delay Test registers in Appendix 1.) To confirm on a scope that the delays are equal the helicity signals output from the module must include the trigger latency delay. For this test *all* reporting delays of the Control board (0, 1, 2, 4, 8, 16, 24, 32, 40, 48, 64, 72, 96, 112, 128, 256 helicity windows) can be tested no matter what helicity pattern is running,

Another way to test the helicity reporting delay again involves replacing the *pair_sync* input signal to the Decoder by the (non-delayed) *helicity_flip* signal from the Control board. In this test the signal is not delayed by the Decoder. Data is collected by the Decoder in test mode. In the analysis of the data the predicted helicity for each event is computed by advancing the helicity sequence by the known reporting delay of the Control board. The predicted helicity is compared to the actual helicity latched from the *pair_sync* input (*helicity_flip*). With this method one can only test delays that are integral numbers of the helicity pattern used. (For the quartet pattern all delays are valid except 1 and 2 windows; for 64-bit patterns the only valid delays are 0, 64, 128, or 256 windows.)

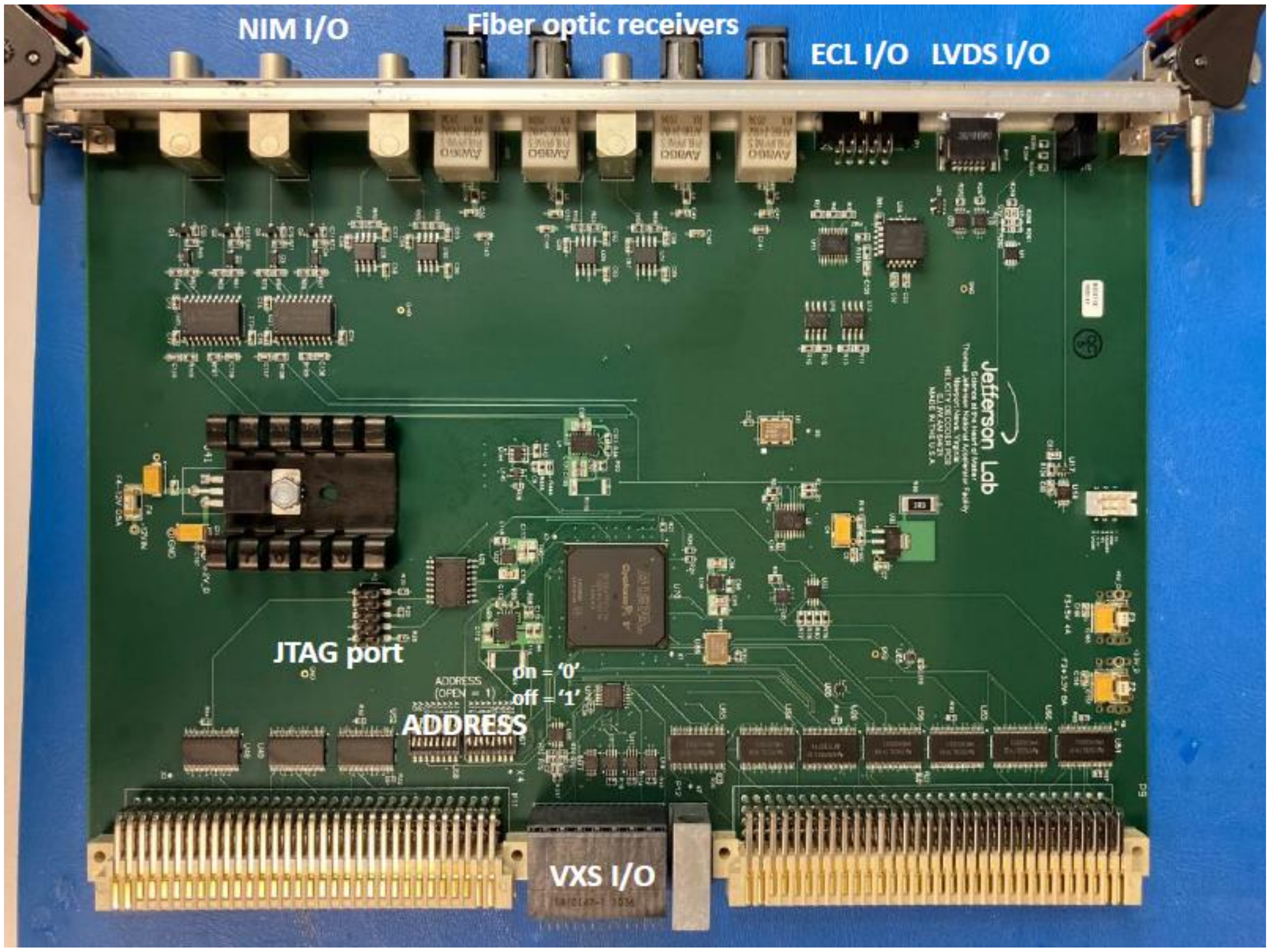


Figure 5

References

1. *Helicity Control Board User's Guide (Draft 2)*, Roger Flood, Scott Higgins, Riad Suleiman, February 4, 2010.

2. *Helicity Signal Generation at Jefferson Lab*, R. Suleiman, R. Flood, J. Hansknecht, S. Higgins, M. Poelker. June 23, 2014.

3. *MOLLER Helicity Board Settings*, Joe Grames, Paul King, Robert Michaels, Prasanna Nepal, Caryn Palatchi, Kent Paschke, Riad Suleiman. June 30, 2022.

4. *Conceptual Specs for Helicity Decoder Board (Draft 2)*, P. M. King, November 20, 2020.

5. VXS Base Standard – VITA 41.0 (ANSI ratified).

Appendix 1 - Using the Helicity Decoder Module (V10)

1.1 Controlling the Module

Communication with the module is by standard VME bus protocols. All registers and memory locations are defined to be 4-byte entities. The VME slave module has two distinct address ranges.

A24 – The base address of this range is set by a 16-element DIP switch on the board. It occupies 256 bytes of VME address space, organized in 64 32-bit words. Relative to the base address, this space is utilized as follows:

00-FF – Register space to control and monitor the module

A32 - The base address of this range is programmed into register ADR32. It occupies 256 Kbytes of VME address space, organized in 64K 32-bit words. A read of any address in this range will yield the next data word from the module. Even though the module is a FIFO, the expanded address range allows the VME master to increment the address during block transfers. This address range can participate in single cycle, 32-bit block, and 64-bit block reads. The only valid write to this address range is the data value 0x80000000 which re-enables the module to generate interrupts (after one has occurred). The address range must be enabled by setting ADR32[0] = 1.

1.2 Module Operation

After a reset of the module (CSR[31] = 1), the system clock source is set (CTRL_1[2..0]). The sync_reset source is set (CTRL_1[6..5]). The BLOCK SIZE register is loaded with the number of events (i.e. triggers) that constitute a *block*. The INTERRUPT register may be loaded with the interrupt ID and level if the module is to initiate an interrupt when the defined *block* of data is available for readout. The address for data access is loaded (ADR32). The event level interrupt (CTRL_1[16] = 1) is enabled if interrupt generation is desired. The BERR response is enabled (CTRL_1[17] = 1) if the module is required to indicate when the complete *block* of data has been read out. Select the trigger source (CTRL_1[4..3]). Enable the decoder function (CTRL_2[0] = 1) and event building function (CTRL_2[2] = 1). A sync_reset signal is generated (CSR[28] = 1) or delivered to the module. Enable the GO bit (CTRL_2[1] = 1) to allow triggers to be accepted by the module.

When the programmed number of triggers has been received, the Block Ready Flag (CSR[3]) will be set and an interrupt will be generated if enabled. The user should initiate a DMA block read (32 or 64-bit) from the address in stored in ADR32. The length of the block read should be programmed to be the expected number of words in the block if termination by BERR is not selected (CTRL_1[17]). Otherwise, the length of the block read should be chosen larger than the expected size of the data block. In this case the module will terminate the DMA transfer by issuing BERR when all data from the *block* has been transferred. Interrupt generation must be re-enabled by writing 0x80000000 to the address in ADR32.

A trigger accepted by the module will generate an event fragment that consists of **18** data words (32-bits each). Block Header and Trailer words enclose the event data for the block, so the smallest block has **20** words. Current data acquisition software allows for blocks as large as 255 events (4592 words). The 64K word on-board storage (FPGA memory) thus allows for 14 maximum size blocks to be stored on board (3570 events).

1.3 Module Registers

VERSION – board/firmware revision (0x00) **(reg 0)**

[7…0] – (R) – firmware revision

[15…8] – (R) – board revision

[31…16] – (R) – board type ("DEC0")

CSR – Control/Status (0x04) **(reg 1)**

0 – (R) – System clock PLL locked status (1 = locked)

1 – (R) – Module clock PLL locked status (1 = locked)

2 – (R) – Block of Events Accepted (Event Level Flag)

3 – (R) – Block of Events Ready for readout

4 – (R) – Events on board empty flag status (1 = no events)

5 – (R) – BERR status (1 = module asserted BERR)

6 – (R) – BUSY status (current)

7 – (R) – latched BUSY status ('1' means at least one occurrence). Clear latched status by writing '1' to this bit.

8 – (R) – Internal Buffer #0 empty flag

9 – (R) – Internal Buffer #1 empty flag

10 – (R) – Helicity sequence error

11 – (R/W) – TRIGGER TIME WORD ERROR – a mismatch of build event number and trigger number from time word occurred. Clear latched status by writing '1' to this bit.

[12…15] – Spare (read as zero)

16 – (W) – FORCE BLOCK TRAILER INSERTION – will be successful only
if there are NO triggers waiting to be processed

17 – (R) – Last FORCE BLOCK TRAILER INSERTION Successful

18 – (R) – Last FORCE BLOCK TRAILER INSERTION Failed

[19…27] – Spare (read as zero)

28 – (W) – Pulse software generated SYNC_RESET (if CTRL_1[7] = 1)

29 – (W) – Pulse software generated TRIGGER (if CTRL_1[7] = 1)

30 – (W) – Pulse soft reset

31 – (W) – Pulse hard reset

CTRL 1 – (0x08) **(reg 2)**

[1…0] – (R/W) – System clock select (0 = P0, 1 = FP 1, 2 = FP 2, 3 = internal)
(FP = front panel)

2 – (R/W) – Internal system clock chip enable (0 = OFF, 1 = ON)

[4…3] – (R/W) – TRIGGER source (0 = P0, 1 = FP 1, 2 = FP 2, 3 = soft)

[6…5] – (R/W) – SYNC_RESET source (0 = P0, 1 = FP 1, 2 = FP 2, 3 = soft)

7 – (R/W) – Enable Soft control signals (SYNC_RESET, TRIGGER)

8 – (R/W) – Use internal TEST trigger – overrides selection of bits 3,4
(currently generated from T_STABLE signal)

[9…11] – (R/W) – Spare

12 – (R/W) – ‘0’ – drive module BUSY signal out of front panel busy outputs
‘1’ – drive DEBUG signal out of front panel busy outputs

==
(Helicity signals are latched 4 us into the T_SETTLE period; all signals from the Helicity Generator are verified to be stable 1 us into this period. Noise on T_SETTLE can cause erroneous latching and corruption of Helicity sequence information. An erroneous latch will manifest itself as an extra Helicity window thus corrupting the computation of the Current Seed and History information. Note: at trigger time the status of the four ‘live’ Helicity signals is recorded, not the values at T_SETTLE latch time. (See Decoder Data word 10 bits 0-3).)
==

[15…13] – (R/W) – glitch filtering level applied to ALL helicity signals (8 ns clock period)
0 – no filtering
1 – 4 clock period filtering (32 ns)
2 – 8 clock period filtering (64 ns)
3 – 16 clock period filtering (128 ns)
4 – 32 clock period filtering (256 ns)
5 – 64 clock period filtering (512 ns)
6 – 128 clock period filtering (1024 ns)
7 – 256 clock period filtering (2048 ns)

16 – (R/W) – Enable Interrupt

17 – (R/W) – Enable BERR response

18 – (R/W) – '1' – use internal helicity generator signals for processing
'0' – use external helicity input signals for processing

19 – (R/W) – '1' – select copper cable helicity inputs as external inputs used
'0' – select fiber optic helicity inputs as external inputs used

20 – (R/W) – '1' – route internal generator signals to helicity front panel outputs
'0' – route helicity signals used for processing to front panel outputs
(Note: these signals do not include the trigger latency delay)
**** See bit 24 which can override this selection *****

21 – (R/W) – '1' – invert fiber helicity input signals
'0' – no signal inversion

22 – (R/W) – '1' – invert copper helicity input signals
'0' – no signal inversion

23 – (R/W) – '1' – invert copper helicity output signals
'0' – no signal inversion

24 – (R/W) – '1' – route processed signals to helicity front panel outputs
'0' – route selection from bit 20 to helicity front panel outputs
(Note: processed signals include the trigger latency delay)
**** '1' overrides the selection from bit 20 *****

[25…31] – (R/W) – Spare

CTRL 2 – (0x0C) **(reg 3)**

0 – (R/W) – Enable decoder

1 – (R/W) – GO – Enable triggers

2 – (R/W) – Enable event build

[3…7] – (R/W) – Spare

8 – (R/W) – Enable internal helicity generator (set after Helicity CONFIG 1, CONFIG 2, CONFIG 3 are written)

9 – (R/W) – Force BUSY output

[10…31] – (R/W) – Spare

ADR32 – Address for data access (0x10) **(reg 4)**

0 – (R/W) – Enable 32-bit address decoding

1 – 6 – (not used – read as 0)

[15…7] – (R/W) – Base Address for 32-bit addressing mode (8 Mbyte total)

INTERRUPT (0x14) **(reg 5)**

[7…0] – (R/W) – Interrupt ID (vector)

[10…8] – (R/W) – Interrupt Level [2..0]. Valid values = 1,..,7.

11 - 15 – (not used)

[20…16] – (R) – Geographic Address (slot number) in VME64x chassis.

21 – 22 – (not used – read as ‘0’)

23 – (R) – Parity Error in Geographic Address.

24 – 31 – (not used – read as ‘0’)

BLOCK SIZE (0x18) **(reg 6)**

[15…0] - (R/W) – Number of events defining a block

[31…16] – (not used = read – as ‘0’)

TRIGGER LATENCY and DATA DELAY (0x1C)   **(reg 7)**
(This register must always be programmed.)

[11…0] – (R/W) – latency value – **must** be non-zero (1 count = 8 ns)

[14…12] – (not used = read – as ‘0’)

15 – (R) – latency configured

[27…16] – (R/W) – data delay value – **must** be non-zero (1 count = 8 ns)

[30…28] – (not used = read – as ‘0’)

31 – (R) – data delay configured

==

Internal Helicity generator configuration registers: CONFIG 1 and CONFIG 2 must be written before CONFIG 3. (The write to CONFIG 3 loads values into generator.) After writing CONFIG registers enable generator by setting CTRL_2[8] = 1.

HELICITY CONFIG 1 (0x20) (internal generator)   **(reg 8)**

[1…0] – (R/W) – Pattern mode: 0 = pair, 1 = quartet, 2 = octet, 3 = toggle

[7…2] – (not used = read – as ‘0’)

[15…8] – (R/W) – Helicity delay in windows

[31…16] – (R/W) – Helicity settle time (1 count = 40 ns)

HELICITY CONFIG 2 (0x24) (internal generator)   **(reg 9)**

[27…0] – (R/W) – Helicity stable time (1 count = 40 ns)

[31…28] – (not used = read – as ‘0’)

HELICITY CONFIG 3 (0x28) (internal generator)   **(reg 10)**

[29…0] – (R/W) – Initial pseudorandom sequence seed

[31…30] – (not used = read – as ‘0’)

==

TEST REGISTER / INTERNAL TEST TRIGGER DELAY (0x2C) **(reg 11)**

[31…0] – (R/W) – test value (used to confirm data integrity to/from module)

When CTRL_1[8] = 1:
[17…0] – (R/W) – test trigger delay value – (1 count = 8 ns; max = 2097 us)

TRIGGER 1 SCALER – (0x30) **(reg 12)**

[31…0] – (R) – total event count

TRIGGER 2 SCALER – (0x34) **(reg 13)**

[31…0] – (R) – total TRIGGER 2 count

SYNC_RESET SCALER – (0x38) **(reg 14)**

[31…0] – (R) – total SYNC_RESET count

EVENTS ON BOARD (0x3C) – Number of events currently stored on board **(reg 15)**

[23…0] - (R) – event count[23…0]

[31…24] – (not used – read as ‘0’)

BLOCKS ON BOARD – (0x40) **(reg 16)**

[31…20] – not used

[19…0] – (R) - number of event BLOCKS on board (non-zero → CSR[4] = 1).

HELICITY SCALER 1 – (0x44) **(reg 17)**

[31…0] – (R) – count of falling edge T_STABLE

HELICITY SCALER 2 – (0x48) **(reg 18)**

[31…0] – (R) – count of rising edge T_STABLE (latched on read of HELICITY SCALER 1)

HELICITY SCALER 3 – (0x4C) **(reg 19)**

[31…0] – (R) – count of PATTERN_SYNC (latched on read of HELICITY SCALER 1)

HELICITY SCALER 4 – (0x50) **(reg 20)**

[31…0] – (R) – count of PAIR_SYNC (latched on read of HELICITY SCALER 1)

==

Processing Clock Test Register: The processing clock (125 MHz) that may be derived from an external source is validated. A write of any data to the register initiates a 10.24 us time interval over which the processing clock periods are counted. The time interval is generated by a fixed free-running on-board clock (50 MHz). A subsequent read of the register should yield a value of **1280** counts if the processing clock frequency is correct.

PROCESSING CLOCK TEST REGISTER – (0x54) **(reg 21)**

[31…0] – (R/W) – count of processing clock periods in a10.24 us time interval

==

RECOVERED SHIFT REGISTER VALUE (0x58) **(reg 22)**

[29…0] – (R/W) – Current recovered value

[31…30] – (not used = read – as ‘0’)

GENERATOR SHIFT REGISTER VALUE (0x5C) (internal generator) **(reg 23)**
(Note: this value is latched when the RECOVERED value is read)

[29…0] – (R/W) – Current internal generator value

[31…30] – (not used = read – as ‘0’)

==

Delay Confirmation Registers: The two digital delays programmed in the TRIGGER LATENCY and DATA DELAY register (0x1C) are implemented with dual-port memory structures. The difference between the WRITE address (wraddr) and the READ address (rdaddr) is the dynamic measure of delay in counts (1 count = 8 ns). To be precise:

```
if (wraddr > rdaddr)  delay = wraddr – rdaddr
else                  delay = 4096 + wraddr – rdaddr
```

The computed delay values should **always** match the values programmed into the TRIGGER LATENCY and DATA DELAY register. A read of the TRIGGER LATENCY CONFIRMATION REGISTER latches data for both confirmation registers and thus should be read first.

<u>TRIGGER LATENCY CONFIRMATION REGISTER</u> (0x60) **(reg 24)**

[11…0] – (R) – Memory READ address

[15…12] – (not used = read – as ‘0’)

[27…16] – (R) – Memory WRITE address

[31…28] – (not used = read – as ‘0’)

<u>DATA DELAY CONFIRMATION REGISTER</u> (0x64) **(reg 25)**

[11…0] – (R) – Memory READ address

[15…12] – (not used = read – as ‘0’)

[27…16] – (R) – Memory WRITE address

[31…28] – (not used = read – as ‘0’)

==

<u>Helicity History Registers</u>: Bit 0 is the most recent value. A read of REGISTER 1 latches data for all 4 registers and thus should be read first.

<u>HISTORY REGISTER 1</u> (0x68) **(reg 26)**

[31…0] – (R) – Last 32 windows of PATTERN_SYNC

<u>HISTORY REGISTER 2</u> (0x6C) **(reg 27)**

[31…0] – (R) – Last 32 windows of PAIR_SYNC

<u>HISTORY REGISTER 3</u> (0x70) **(reg 28)**

[31…0] – (R) – Last 32 windows of reported HELICITY

HISTORY REGISTER 4 (0x74) **(reg 29)**

[31…0] – (R) – Last 32 values of reported HELICITY at PATTERN_SYNC

===

SPARE REGISTER (0x78) **(reg 30)**

SPARE REGISTER (0x7C) **(reg 31)**

===

Helicity Delay Test Registers: To test the helicity reporting delay of the Helicity Generator two registers are implemented. For the test the PAIR_SYNC input signal to the Decoder is replaced by the (non-delayed) HELICITY_FLIP signal from the Generator. The Decoder delays this signal by a programmable number of helicity windows (pair delay). If this delay is equal to the reporting delay of the Generator, the PAIR_SYNC and HELICITY_STATE signals should be identical. If the delay is enabled a counter is incremented whenever the signals do not agree at the time of their latching at the beginning of each helicity window. The count is available in a register and can be reset anytime.

DELAY SETUP REGISTER (0x80) **(reg 32)**

31 – (R/W) – Enable delay of PAIR_SYNC signal

[30…4] – (not used = read – as ‘0’)

[3…0] – (R/W) – Pair delay selection

0 – no delay
1 – 1 window delay
2 – 2 window delay
3 – 4 window delay
4 – 8 window delay
5 – 16 window delay
6 – 24 window delay
7 – 32 window delay
8 – 40 window delay
9 – 48 window delay
10 – 64 window delay
11 – 72 window delay
12 – 96 window delay
13 – 112 window delay
14 – 128 window delay
15 – 256 window delay

DELAY ERROR COUNT (0x84) **(reg 33)**

31 – (W) – Writing ‘1’ initiates a pulse that resets the error count

[31…0] – (R) – Error count since last reset.

==

SPARE REGISTER (0x88) **(reg 34)**

SPARE REGISTER (0x8C) **(reg 35)**

==

CONFIGURATION CSR (0x90) – (Firmware Update) **(reg 36)**

31 – (R/W) – Write Enable (1 = write mode, 0 = read mode)

30 – (R/W) – Bulk Erase (bit 31 = 1 also required)

29 – (R/W) – Sector Erase (bit 31 = 1 also required)

[28…16] – (R/W) – Reserved

[15…9] – (R) – Reserved

8 – (R) – Busy (operation in progress)

[7…0] – (R) – Last Valid Data Read

CONFIGURATION ADR/DATA (W) (0x94) – (Firmware Update) **(reg 37)**

[31…8] – EPROM address

[7…0] – EPROM data to write

==

Appendix 2 - Data Format

2.1 Data Word Categories

Data words from the module are divided into two categories: Data Type Defining (bit 31 = 1) and Data Type Continuation (bit 31 = 0). Data Type Defining words contain a 4-bit data type tag (bits 30 - 27) along with a type dependent data payload (bits 26 - 0). Data Type Continuation words provide additional data payload (bits 30 – 0) for the *last defined data type*. Continuation words permit data payloads to span multiple words and utilize bits 30 - 27 as data. Any number of Data Type Continuation words may follow a Data Type Defining word. The decoder data header type is an exception. It specifies the number of 32-bit data words that follow.

2.2 Data Type List

0 – block header
1 – block trailer
2 – event header
3 – trigger time

8 – decoder data header

14 – data not valid (empty module)
15 – filler (non-data) word

2.3 Data Types

Block Header (0) – Word 1 indicates the beginning of a block of events.

(31) = 1
(30 – 27) = 0
(26 – 22) = slot number (set by VME64x backplane)
(21 – 18) = module ID (‘D’ for Helicity Decoder)
(17 – 8) = event block number
(7 – 0) = number of events in block

Block Trailer (1) – indicates the end of a block of events.

(31) = 1
(30 – 27) = 1
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = total number of words in block of events

Event Header (2) – indicates the start an event.

(31) = 1
(30 – 27) = 2
(26 – 22) = slot number (set by VME64x backplane)
(21 – 12) = trigger time (bits 9 – 0 (see below))
(11 – 0) = trigger number

Trigger Time (3) – time of trigger occurrence relative to the most recent global reset. Time is measured by a 44-bit counter that is clocked by the 125 MHz system clock. The six bytes of the trigger time

$$\text{Time} = T_A\ T_B\ T_C\ T_D\ T_E\ T_F$$

are reported in two words (Type Defining + Type Continuation).

Word 1:

(31) = 1
(30 – 27) = 3
(26 – 24) = T_C bits 2 – 0 (duplicated in Word 2)
(23 – 16) = T_D
(15 – 8) = T_E
(7 – 0) = T_F

Word 2:

(31) = 0
(30 – 20) = reserved (read as 0)
(19 – 16) = T_A (lower 4 bits)
(15 – 8) = T_B
(7 – 0) = T_C

Decoder Data Header (8) – indicates the beginning of a block of Decoder Data words. The number of 32 bit data words that will immediately follow it is provided in the header. The fixed order of the data words is used to identify them (see below). Currently there are 14 words reported for each event.

(31) = 1
(30 – 27) = 8
(26 – 6) = reserved (read as 0)
(5 – 0) = number of decoder data words to follow (14 = current)

Data Not Valid (14) – module has no valid data available for read out.

(31) = 1
(30 – 27) = 14
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = undefined

Filler Word (15) – non-data word appended to the block of events. Forces the total number of 32-bit words read out of a module to be a multiple of 2 or 4 when 64-bit VME transfers are used. **This word should be ignored**.

(31) = 1
(30 – 27) = 15
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = undefined

2.4 Decoder Data Words

1. Helicity seed
   31 – Expected HELICITY_STATE value at *next* PATTERN_SYNC
   (XOR of current seed (29 – 0) bits 29, 28, 27, 6)
   30 = 0
   (29 – 0) – recovered seed from past 30 HELICITY_STATE values at PATTERN_SYNC

2. Count of falling edge T_STABLE (32 bits)

3. Count of rising edge T_STABLE (32 bits)

4. Count of PATTERN_SYNC (32 bits)

5. Count of PAIR_SYNC (32-bits)

6. Time of trigger from start of T_STABLE interval (32 bits, 1 count = 8 ns)

7. Time of trigger from end of T_STABLE interval (32 bits, 1 count = 8 ns)

8. Time duration of last complete T_STABLE interval (32 bits, 1 count = 8 ns)

9. Time duration of last complete settling interval (T_STABLE not asserted) (32 bits, 1 count = 8 ns)

10. Status at trigger time
    (0) – T_STABLE state (live)
    (1) – PATTERN_SYNC state (live)
    (2) – PAIR_SYNC state (live)
    (3) – HELICITY state (live)
    (4) – HELICITY state at pattern start (latched at T_SETTLE start)
    (5) – Event Polarity (XOR [HELICITY, HELICITY at pattern start])
    (7 – 6) – ‘0’
    (15 – 8) – Pattern phase count
    (31 – 16) – ‘0’

11. Last 32 windows of PATTERN_SYNC (32 bits)

12. Last 32 windows of PAIR_SYNC (32 bits)

13. Last 32 windows of HELICITY_STATE (32 bits)

14. Last 32 values of HELICITY_STATE at PATTERN_SYNC (32 bits)

NOTES

(1) let T1 = Time of trigger from start of T_STABLE interval,
T2 = Time of trigger from end of T_STABLE interval

If trigger occurred during stable interval (T_STABLE asserted), then
T2 > T1 and T2 – T1 = settling duration value.

If trigger occurred during settling interval (T_STABLE not asserted), then
T1 > T2 and T1 – T2 = stable duration value.