Argonne NATIONAL LABORATORY

Digital Gammasphere Firmware User's Manual

For experts and firmware nerds only

-- PRELIMINARY --

September 13, 2021

Version 1.0

This revision of the document describes firmware at SVN revision #6185.

Originator: J. Anderson

Document #TBA

Table of Contents

TABLE OF FIGURES 4

INTRODUCTION 5

DETAILS OF EACH CHANNEL 5

Memory Structures within Xilinx FPGAs 5
Important FPGA generalities 6
Channel Initialization 7
Best initialization practice 8

SPECIFICS OF DISCRIMINATOR OPERATION 9

Operation in the leading-edge discriminator mode 10
Discriminator Hold-off 11
Preamp Reset Kill 12
Peak Detector 12
Difference in operation of the Constant-Fraction mode 13
Time interpolation using the CFD Sample values 15
Detecting improper CFD operation 15
Details of Pileup Logic 16
Handling of Pileup 18
Relationship of pileup to event header formation 18
PEHQ latch timing requirements 20
Holdoff time versus pileup time 20
Event Formation at the end of the pileup inspection time 21
Triggered modes vs. "Internal Accept All" 22
Event expiry 22
Event status bits in the PEQ 23
External Discriminator Modes 23
External Discriminator Signals 24
Timestamp-based external discriminator 25

INSERTION OF TIMING MARKS INTO THE WAVEFORM 26

Timing marks and long waveform readouts 27
Timing marks and down-sampling 27
Formation of timing marks 27

ENERGY SUMMATION 30

P2 and Energy Sums 30
Multiple Pre-Rise sums 31

DATA READOUT AT THE CHANNEL LEVEL 31

Readout general operation when pileup is rejected 31
Readout in "internal accept all" – pileup rejected 32
Effects of long waveform length settings; defining Readout Interference 32
Down-sampled readout 33
Readout of piled-up events 33

STOP READING HERE 36

STOP READING HERE 36

STOP READING HERE 36

Data Format 37
Data Header – leading-edge discriminator mode 37
Data Header – CFD mode 42

- Waveform Data Format Details 42
- Energy Summation Logic Operational Details 44
  - *Comparison of delays and sums versus the standard trapezoidal filter 44*
- Discriminator filter 45
- Constant-Fraction discriminator 48
- Leading-edge hold-off interaction with CFD operation 51
- Trigger Delay Buffer 51
  - *Effects of delayed discriminator bit formation 51*

WAVEFORM READOUT GENERALITIES 52

- Data Collection across the Digitizer and Readout over VME 52
  - *Waveform Down-Sampling 53*
  - *On-Off down-sampling 53*
- Pileup and Event Readout 54
- Readout Interference 54
  - *Readout Interference when Down-sampling is in use 55*
- Pileup Extension and Readout Modes 55
  - *Readout Modes when Pileup Extension is Disabled 56*
  - *Readout operation when Pileup Extension is Enabled 57*

INTERFACE TO THE EXTERNAL TRIGGER 60

- Timestamp Synchronization 60
- Event Veto 60
- Trigger Decision Latency 61
  - *Details of the trigger window calculation 61*
- Other Commands from the Trigger 63
- Information Sent to the Trigger System by the Digitizer 63
- Data Flow Control (Throttle) 63
- Special trigger modes 64

MASTER AND SLAVE DIGITIZERS 64

- Detailed Description of External Discriminator Modes 66
- Typical modes of Slave Digitizer Operation 67
  - *Gammasphere "Independent" Mode 67*
  - *Gammasphere "Clean", "Dirty" and "Module" Modes with Slave Digitizers 68*
  - *"Pseudo-GRETINA" Mode 69*
- Diagnostic capabilities using the on-board DAC 70

GLOSSARY 71

Table of Figures

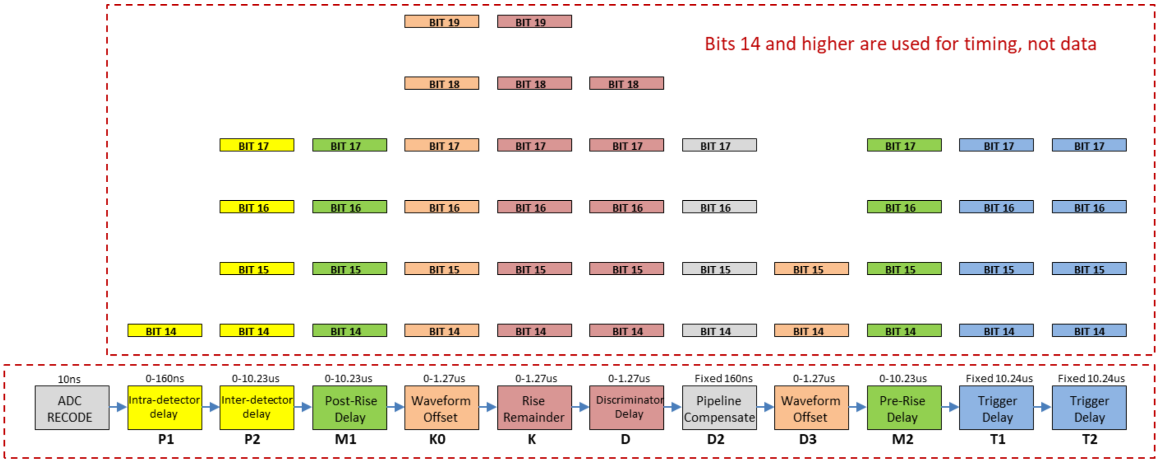
Figure 1 - Delay buffers per channel have differing widths for implementation of delays without counters. 5
Figure 2 - Timing of CHAIN_INVALID and sequenced initialization thereafter. 7
Figure 3 - Use of post-rise buffer to create delayed copy of CHAIN_INVALID 9
Figure 4 : pipeline filter calculation for discriminator noise reduction...10
Figure 5 – Channel architecture in the leading-edge discriminator mode. 10
Figure 6 - simulation of leading-edge discriminator operation. 11
Figure 7 - Channel architecture in the constant-fraction discriminator mode. 14
Figure 8 - CFD simulation showing CFD equation versus discriminator bit from leading-edge arming discriminator......14
Figure 9 - Pileup counter operation from simulation of firmware operation. 16
Figure 10 - Use of delay buffers M2, K and K0 to form pileup delay in the *leading-edge* mode. 17
Figure 11 - Use of delay buffers M2, K and K0 to form pileup delay in the constant-fraction mode. 17
Figure 12 - Pileup operation, with horizonal axis scaled to actual time. 18
Figure 13 - The PEHQ and Pending Event Queue (PEQ) as related to the channel pipeline. 19
Figure 14 - relationship between pipeline, PEHQ, PEQ and Event Header FIFO. 22
Figure 15 - timing of event expiration. 22
Figure 16 - External discriminator selection logic of one channel. 23
Figure 17 - Signal delays and logic of timing marks in threshold discriminator (LED) mode. 28
Figure 18 - Signal delays and logic of timing marks in constant-fraction (CFD) mode. 28
Figure 19 - Timing marks in waveforms as displayed in GammaWare. 29
Figure 20 - GammaWare display showing both timing marks and delay buffer sizes overlaid upon waveform 29
Figure 21 - approximate positions where early pre-rise sums are captured..31
Figure 24 - Event taken at downsample factor of 2, thus spanning 2^2 clock ticks (40ns) per sample ....**Error! Bookmark not defined.**
Figure 25 - Event taken at downsample factor of 2, in the on-off mode, with downsample holdoff of 255. ....**Error! Bookmark not defined.**
Figure 22 - Trigger windows within the digitizer 35
Figure 23 - Summary description of all flag bits within the header. 38
Figure 26 - Pre- and Post-rise accumulator logic 44
Figure 27 - Estimated frequency response plot of digital leading-edge discriminator filter. 46
Figure 28 - discriminator hold-off set far shorter (200ns) than rise time (~1us) - multiple hits per rise. 46
Figure 29 – Discriminator and Energy Summation portion of channel logic (CFD mode). 49
Figure 30 - Screen capture from GammaWare showing positions of buffers in CFD mode 50
Figure 31 - Event read out full speed (left) and down-sampled by 8x (right) 53
Figure 32 - Example waveform for describing pileup readout modes 55
Figure 33 - Visual representation of the readout modes with Pileup Extension DISABLED...57
Figure 34 - Visual representation of the readout modes with Pileup Extension ENABLED 59
Figure 35 - Trigger windows within the digitizer 62
Figure 36 - Connections between Master and Slave digitizers. 65

Introduction

This document is intended for users of the experiment setups at ATLAS who will be working with the firmware and adjusting parameters; a separate document, *Waveform Digitizers at the ATLAS experiments*, is available for those interested in a general overview of the digital data acquisition systems without reference to technical details. This document presupposes that the reader is already familiar with the overall design of the firmware and associated terminology.

Details of each channel

The 10 channels of the firmware are implemented as continuously running data pipelines. The structure of each channel is based upon the use of multiple delay buffers of varying width. Many of the timing functions of the firmware are based upon these delay settings. Xilinx FPGA logic cells implement small (16x1) RAM blocks that can also be configured as shift registers where the four input bits are then used as the length of the shift register. Similarly, Xilinx FPGA block RAM elements have parity bits that can be used for any general purpose. Both these features enable the designer to craft delay lines for single bits that are the same length as the delay buffers using far fewer FPGA resources than traditional counter-based delays. The delay buffer structure of each channel as implemented in shown in Figure 1.



The data path is 14 bits wide (bits 13:0) throughout

Figure 1 - Delay buffers per channel have differing widths for implementation of delays without counters.

Memory Structures within Xilinx FPGAs

The Xilinx FPGA implements two forms of memory resources, the ***Block RAM*** and ***Distributed RAM***. All RAM within a Xilinx FPGA is dual-port in nature. Block RAMs are 18kbit units that may be reconfigured into a few standard sizes (1Kx18, 512x36, etc.). The bit-width of all Block RAM implementations is based upon a parity memory so that there are always a multiple of nine, rather than eight, bits of width. Most of the time these extra bits are ignored but as they are inherent to the Block RAM architecture they may be used to carry data unrelated to the other bits so long as common clocking is acceptable. On the downside, Block RAM is scattered throughout the chip in fixed areas so as more and more of the Block RAM is used, it becomes increasingly difficult to route signals to those Block RAMs that remain. Many designs fail to route not because the chip is full, but because the design tries to use every Block RAM and the router software cannot find paths to every RAM at the speed required.

Distributed RAM memory uses the actual memory of the many logic slices of the FPGA itself. Every logic slice within the FPGA implements small 16x1 memories to perform the basic logic functions (AND, OR, etc.) as it is more efficient to simply use a small lookup table than to implement hard gates. This distributed RAM may be ganged together in small units to replace Block RAM and has the placement advantage of always being physically near the logic that wants to use it. Additionally, unlike block ram, the width of distributed structures is arbitrary. The downside of distributed RAM is that it requires some logic to gang together so sizes above 128x16 are usually impractical.

When implementing a delay structure as opposed to a general-purpose dual-port memory, the Distributed RAM may be further optimized by specific shift register macros named “SRL” in the Xilinx literature. These SRL macros use special internal connections available within the 16X1 distributed RAM buffer to “rewire” the RAM into a 16-element shift register. Data enters at position 0 and any one position may be brought out as the output. Serially ganging these SRLs together – assuming interconnect delays between them can meet timing constraints – allows arbitrarily long delay buffers to be implemented. As the SRL method does not require the address counters necessary in a generic dual-port RAM implementation, the SRL methodology provides very significant resource utilization reductions.

In the digitizer module, the actual ADC data is only 14 bits wide; thus, if a mix of Distributed RAMs (of any width desired) and 1Kx18 Block RAMs is used, there are four bits available for timing purposes across the entire delay chain. As most timing parameters are associated with the buffer lengths use of these extra bits to form state machine delays is highly efficient. The normal method by which this is done is to have a state machine generate a pulsed signal one clock period wide at the start of a delay state that is connected to the input side of a delay buffer. The state machine then waits for the delayed copy of the bit to fall out of the other side of the delay buffer before proceeding or branching. Resources needed to implement a delay counter are eliminated and comparison logic is reduced to a single bit. While this may not seem like much, multiplied over 10 channels with multiple delays per channel results in a significant fraction of the FPGA’s resources being conserved.

Important FPGA generalities

Within Xilinx FPGAs all logic must be formed from cells called CLBs (Configurable Logic Blocks). All CLBs consist of a small lookup memory with four inputs and one output (thus a 16x1 RAM) that implements all gate-level logic and a D flip-flop. There are no AND, NAND, OR, NOR, XOR or NOT gates inside; all asynchronous logic is formed from the 16x1 RAMs. Similarly, there is no such thing as a set-reset (SR) latch in an FPGA; there are only D flip-flops.

Collections of D flip-flops that work synchronously on a group of bits interpreted as a value, but that have no connection to software programs outside the FPGA, are called ***latches*** throughout this document. This is to differentiate them from ***registers***, that are constructs of software outside the FPGA used to load control values to or read status values from the FPGA. Obviously, ***registers*** are formed out of D flip-flops too, but to understand the timing of firmware operations there must be a distinct term for the internal objects that sample data values within state machines or pipelined signal processing chains. The word ***latch*** should imply "D flip-flop inside the FPGA" whereas ***register*** should imply "D flip-flop that connects outside the FPGA".

The design of the CLB must be understood to mean that every D flip-flop in the FPGA has a built-in multiplexer at the input. Additionally, the D flip-flop in every CLB has a *clock enable* term that allows it to only sample on some clocks, not all. Because of this, the word ***latch,*** when used in any drawing within this document, ***should not be misinterpreted to be an object that can only have one input.*** A drawing object identified by the word 'latch' with multiple inputs simply means that the built-in multiplexer is being used and that this particular set of D flip-flops can sample different input data at different times or under different conditions.

Channel Initialization

The various running sums used in the ANL firmware require proper initialization, as do all functions based upon delays. The firmware design allows the user to load all channel delay parameters into the registers of the module in any order desired without affecting channel pipeline operation. After all new parameters are loaded a single, final, write to a pulsed control register (a register that creates a one-clock-wide pulse in response to a '1' being written to a bit in the register irrespective of control interface speed) called ***LOAD_VALS*** causes a self-timed sequence of initialization throughout the length of the channel pipeline. This is shown in Figure 2.

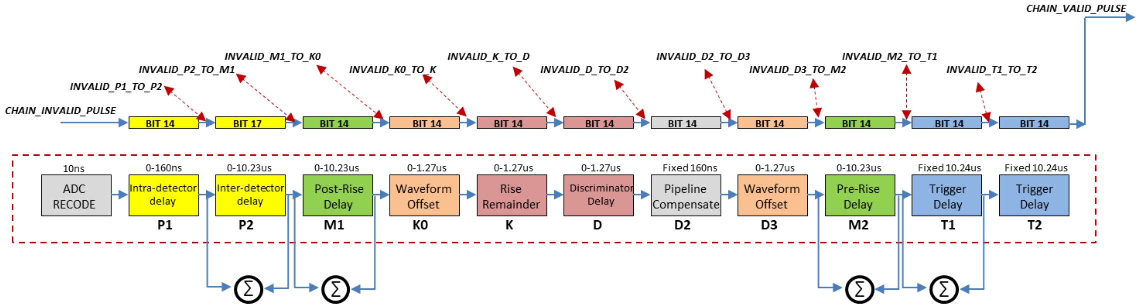


Figure 2 - Timing of CHAIN_INVALID and sequenced initialization thereafter.

In response to the ***LOAD_VALS*** signal another signal CHAIN_INVALID_PULSE is generated that enters the pipeline at the input to buffer P1 and proceeds throughout the pipeline until exiting at the other end. The immediate response to ***LOAD_VALS*** is to reset the length of all delay buffers to their new values, after which CHAIN_INVALID_PULSE enters the pipeline. The various tap delays of CHAIN_INVALID_PULSE initialize the energy sum logic at the appropriate times:

- The sums across P2, M1 and M2 are held at zero until INVALID_P1_TO_P2 goes high, at which point the sum across P2 starts adding.
- When INVALID_P2_TO_M1 goes high, the sum across P2 starts tracking and the sum across M1 starts adding.
- When INVALID_M1_TO_K0 goes high, the sum across M1 starts tracking.
- When INVALID_D3_TO_M2 goes high, the sum across M2 starts adding.
- When INVALID_M2_TO_T1 goes high, the sum across M2 starts tracking.

There is a distinction between similarly named signals CHAIN_INVALID, CHAIN_VALID_PULSE and CHAIN_INVALID_PULSE. When ***LOAD_VALS*** is asserted, CHAIN_INVALID and CHAIN_INVALID_PULSE both are asserted, but CHAIN_INVALID_PULSE lasts for only a few clock ticks. CHAIN_INVALID, once set, stays set for the entire time it takes CHAIN_INVALID_PULSE to propagate through all the delay buffers until CHAIN_INVALID is reset by CHAIN_VALID_PULSE. CHAIN_INVALID is therefore on for the entire duration of the initialization of the pipeline but the various tap delays of CHAIN_INVALID_PULSE are short pulses that occur in sequenced delays after ***LOAD_VALS***.

Best initialization practice

During the initialization sequence an internal bit vector SUBSECTION_RESETS(9:0) is set to “1111111111” when ***LOAD_VALS*** is asserted. The reset vector is held at “1111111111” until the signal ***CHAIN_VALID_PULSE*** falls out the end of the pipeline. At this point, zeroes are shifted into SUBSECTION_RESETS from bit 0 towards bit 9 every tick of the clock until the vector becomes “0000000000”. Different bits of the reset vector control different processes within the channel logic (e.g. discriminator, pileup, etc.) so that the order of release can be controlled.

So long as SUBSECTION_RESETS(9) is high, the **ADC_RECODE** block is held in a state where the input is held constant at a value the user stores in a register. When the reset bit is released normal ADC values are allowed in. This ensures that the energy sums all start with ‘n’ times this initialization value (where ‘n’ is the length of each buffer) and then track from that point to the sum of real data after the chain is released. Thus for best operation, the user should take some test waveforms to estimate the baseline ADC value of the channel in normal operation and set the initial value register for each channel to that estimated baseline value so that there’s not a large false step in the data that may be interpreted as a discriminator edge. As a secondary safeguard the threshold discriminator logic state machine is designed to ignore the first edge after reset, should the user fail to set the initial value register appropriately.

However, since the energy sums are tracking objects, they still require time to track to appropriate values after being released. This can take up to ~25us based upon user settings for the delay buffers. A secondary signal named ENERGY_INVALID, based upon the signal CHAIN_INVALID, is created using the post-rise delay buffer to further disable the threshold discriminator by the length of the post-rise sum buffer after ***CHAIN_VALID_PULSE*** is asserted, as shown in Figure 3. This, plus the ignorance of the first edge after release of the threshold discriminator, should guard against any erroneous energy sums in the first event after reset release under most circumstances.

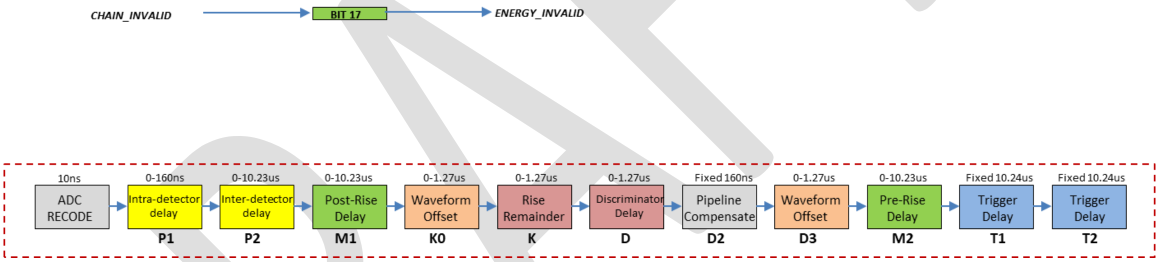


Figure 3 - Use of post-rise buffer to create delayed copy of CHAIN_INVALID

Specifics of discriminator operation

The firmware implements two forms of discriminator logic in every channel, ***leading-edge*** and ***constant-fraction.*** Each channel may be individually configured for mode and polarity of discriminator operation. There are *two* discriminator functions in every channel, an early, unfiltered one called the *coarse discriminator* and the *main* discriminator. Only the *main* discriminator can be operated in both modes; the *coarse* discriminator always operates in leading-edge mode. While the *coarse* discriminator has its own threshold setting, the depth of its delay buffer cannot be changed, and the polarity is constrained to be the same as that of the main discriminator.

The purpose of the *coarse* discriminator is to create a signal with minimum delay so that a fast multiplicity trigger may be generated by Digital Gammasphere for use with auxiliary detectors that do not have the ability to buffer events. As the coarse discriminator provides an "early warning" signal well before the edge reaches the main discriminator logic, the firmware as of August 2021 also uses the coarse discriminator to capture additional copies of the sum across the pre-rise buffer at times before the main discriminator can fire.

The main discriminator uses a digital filter to reduce noise. This filter is based upon the simple filter function Y(n) = X(n) + (2*X(n-1)) + X(n-2). If this filter is convoluted four times the resulting equation is

$$
\begin{aligned}
Y(n) = \quad & X(n) \\
+ \quad & 8 * X(n-1) \\
+ \quad & 28 * X(n-2) \\
+ \quad & 56 * X(n-3) \\
+ \quad & 70 * X(n-4) \\
+ \quad & 56 * X(n-5) \\
+ \quad & 28 * X(n-6) \\
+ \quad & 8 * X(n-7) \\
+ \quad & X(n-8)
\end{aligned}
$$

This is implemented in the firmware in a pipelined process that requires multiple clocks to solve, but comes up with a new answer every clock, as shown in Figure 4.

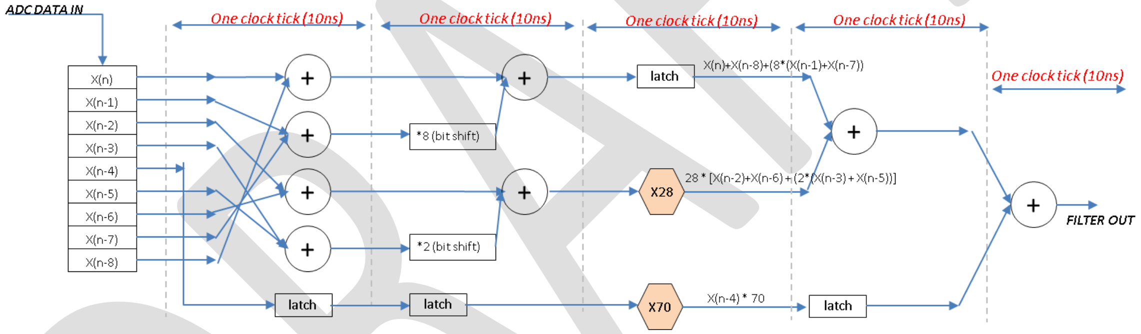


Figure 4 : pipeline filter calculation for discriminator noise reduction.

The ADC samples entering the K buffer, between the K buffer and the D buffer, and exiting the D buffer are sent through three copies of this filter. A multiplexer routes the appropriate two copies to the leading-edge discriminator block depending upon the discriminator operating mode. Since the sum of the coefficients of this filter equals 256, the lower 8 bits of the FILTER_OUT value are discarded so that FILTER_OUT, as used in discriminator logic, is still a 14-bit number to match the ADC raw data.

Operation in the leading-edge discriminator mode

An overall picture of the channel pipeline in the simpler threshold, or leading-edge, discriminator mode is shown in Figure 5.

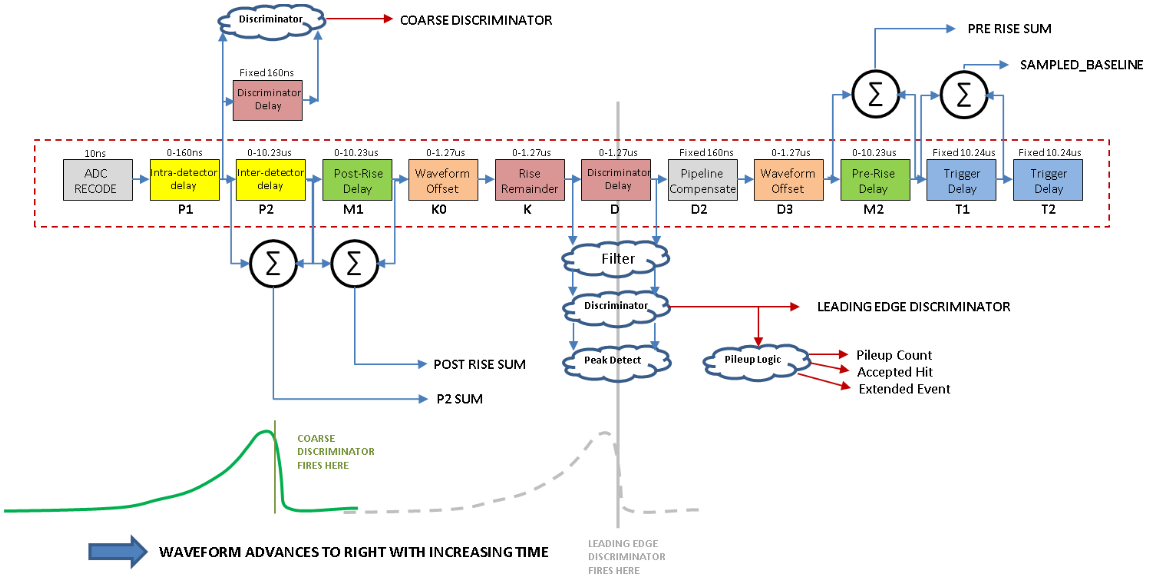


Figure 5 – Channel architecture in the leading-edge discriminator mode.

The two outputs of the two selected filter chains are processed by a discriminator state machine. This state machine subtracts the “delayed” input (the input on the right side, that has more delay from the ADC than the one on the left) from the “prompt” input (the input on the left side, that has less delay from the ADC than the one on the right) every 10ns. For a positive-going pulse this means that the resulting value will become larger as the signal rises. This difference is compared against a discriminator threshold every 10ns, and if the difference exceeds the threshold the discriminator fires. This means that the leading-edge discriminator is a discriminator of ***slope,*** not a discriminator of ***level.*** A calculated simulation of this running subtraction over time is presented as Figure 6.

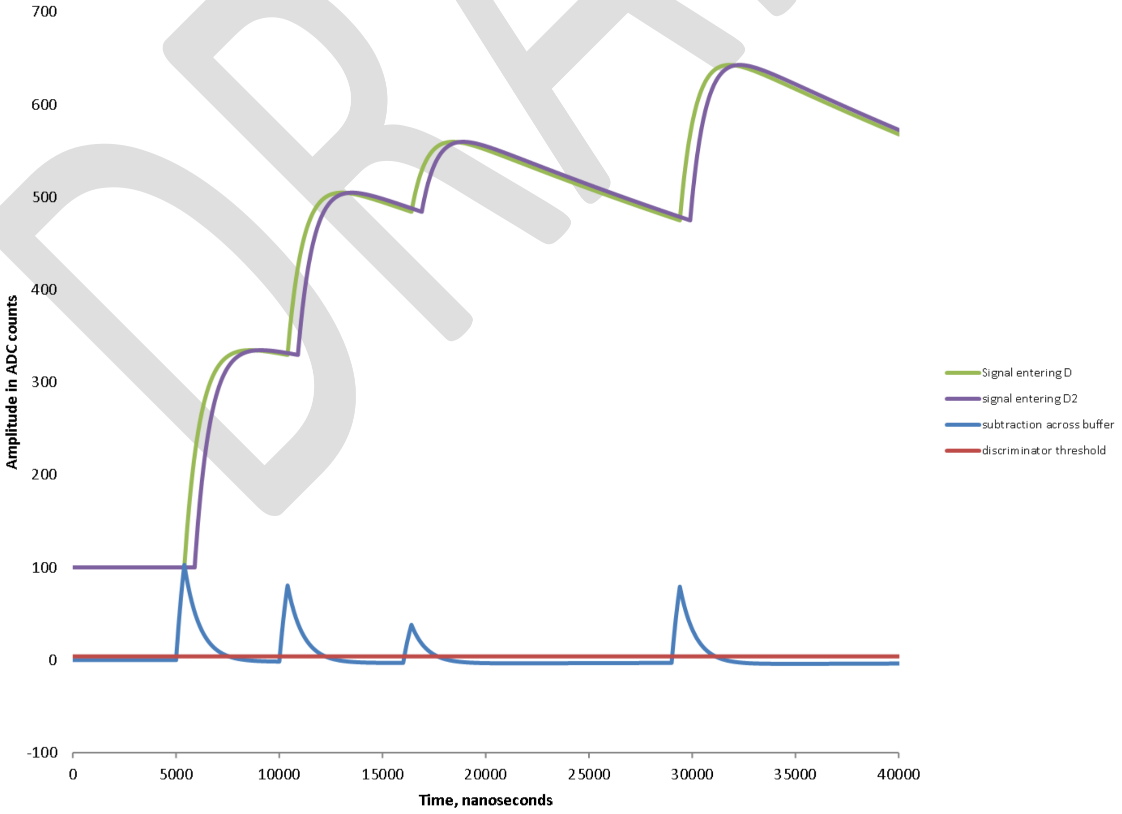


Figure 6 - simulation of leading-edge discriminator operation.

The amplitude of the subtraction is dependent upon the length of the ‘d’ buffer plus the slope of the edge of the signal. A larger ‘d’ allows the firmware to sense smaller amplitude and/or slower rise time signals. Conversely, a smaller ‘d’ exhibits less time walk. As a practical matter, a value of ‘d’ less than 10 is likely to result in non-optimal operation from numerical truncation error in the subtraction. For HPGe detectors with a rise time of less than 1usec, a ‘d’ value of 16 is recommended. For these reasons, the delay buffer of the coarse discriminator uses a fixed delay time of 160ns (16 clocks).

Discriminator Hold-off

Referencing Figure 6, the threshold level is exceeded for many clocks once crossed. The leading-edge discriminator avoids multiple firings by implementing a *discriminator hold-off* value, stored in a control register. On the first tick of the clock when the threshold is crossed, a counter is loaded with the hold-off count, and for that many ticks of the clock the leading-edge discriminator is disabled so that it only fires once per edge. The hold-off time is a function of the rise time of the signal and should be set by the user appropriately for the input signal characteristics. The discriminator output signal used by other portions of the logic is a pulse 10ns wide (one clock) that occurs on the first clock after the threshold is crossed. The hold-off logic ensures that this pulse only occurs once for each input edge.

Preamp Reset Kill

The firmware implements a function akin to hold-off named Preamp Reset Kill. The Gammasphere detectors use *transistor-reset* preamplifiers. This type of preamplifier continuously integrates charge from detector leakage current resulting in an output signal that looks like a slow ramp over many milliseconds. A comparator within the preamplifier detects when the ramp has reached the limit of the analog circuitry at which point a transistor turns on to bleed off all the accumulated charge resulting in a large amplitude, fast recovery. Over period of seconds these resets look like a sawtooth waveform. The detector signals of interest are much smaller steps overlaid upon this long term sawtooth.

The interface circuitry of Digital Gammasphere (the Slope Box Extension, or “SBX”) implements a differentiator circuit between the preamplifier and the digitizer such that the signals of interest become exponentially shaped signals. This has the effect of turning the preamplifier resets into very large exponentials of the opposite polarity of the signals of interest. The SBX implements a fast-acting clamp circuit that greatly limits the excursion and duration of these reset signals, but the clamp cannot eliminate them. This results in an opposing polarity signal of large amplitude, followed by a fast edge returning to baseline, followed by a period during which the clamp is on in which the exponential decay time constant of detector signals is far shorter than normal.

During the reset and the duration of the clamping time measurements of energy are distorted. The Preamp Reset Kill function, if enabled by setting a bit in the channel control register, implements another version of discriminator with large, preset threshold and opposite polarity to the main & coarse discriminators. When this circuit detects a Preamp Reset, it starts a delay count. While the delay is in progress the main discriminator is disabled preventing response to both the Preamp Reset recovery signal and the distorted detector signals.

In Gammasphere the Preamp Reset occurs at rates of once every few milliseconds to once every few hundred milliseconds depending upon how much neutron-induced damage the detector has. The SBX clamping time is normally on the order of 200-250 microseconds, so the dead time percentage introduced by Preamp Reset Kill is not significant until the detector is near the point at which it must be removed from the array and annealed.

Peak Detector

Inside the leading-edge discriminator machine, when the discriminator fires the filtered ADC value at the “prompt” input to the discriminator is saved. On every tick of the clock thereafter, the “prompt” value is compared to the saved value. If the “prompt” value is still climbing (or, if discriminator is set for negative edges, still falling), the saved value is updated with the “prompt” value. The saved value continues to rise (or fall) as the signals rises (or falls) towards its maximum.

There is noise in every physical system, so the peak detector implements a filter called the *peak sensitivity* value. The *peak sensitivity* value is the number of clocks in a row that the saved value is ***not*** updated before the peak is declared. A usual value for this parameter is 4, meaning that if the saved value hasn’t been updated for four clocks in a row the peak is found.

The peak detection process is also controlled by the holdoff time. If a peak is found before the holdoff time, the timestamp of the peak is saved for the header of the event and a bit in the header named PEAK_VALID is set. If the holdoff time elapses before a peak is found, the timestamp of the peak will be reported as zero and the PEAK_VALID flag will not be set.

Conversely, the user may set a bit in the channel control register that will terminate the hold-off time early once a peak is found. This can be useful if the rise time of the input signals varies; the hold-off time is set for the slowest pulses, but terminates early for the faster pulses, resulting in the discriminator always re-arming at the earliest possible time.

Difference in operation of the Constant-Fraction mode

The channel design is slightly changed in when constant-fraction mode is enabled, as shown in Figure 7. The leading-edge discriminator logic moves forward one delay block to span delay buffer “k”, and the center delay “d” is now monitored by the constant-fraction discriminator (CFD). The leading-edge discriminator is used to gate the operation of the CFD, to avoid false firings. The leading-edge discriminator fires as the leading edge of the waveform passes through the “k” buffer. The CFD logic starts calculating when the leading-edge discriminator fires, and the CFD itself fires when the CFD equation at the selection fraction is satisfied.

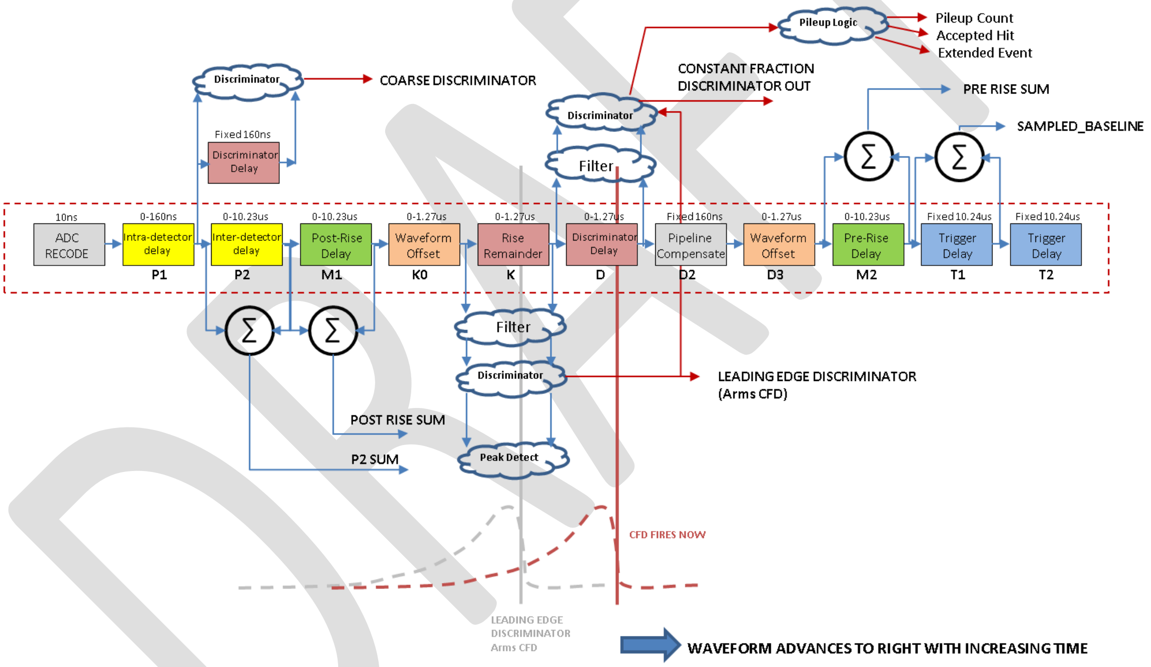


Figure 7 - Channel architecture in the constant-fraction discriminator mode.

All three filter elements are used in CFD mode, with the output of the middle filter connected to the leading-edge discriminator as the “delayed” input and also to the constant-fraction discriminator as its “prompt” input. The CFD state machine is normally in an idle state and waits for the leading-edge discriminator to fire before it begins calculating. Calculations continue until the CFD logic is satisfied or until the leading-edge holdoff time elapses, whichever comes first.

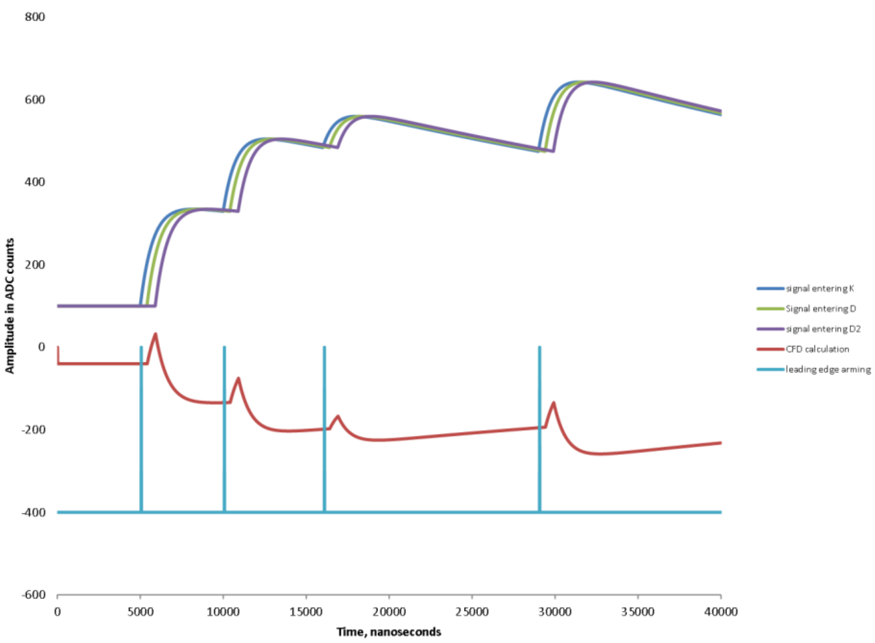


Figure 8 - CFD simulation showing CFD equation versus discriminator bit from leading-edge arming discriminator.

An important observation regarding Figure 8 is the varying level of the CFD equation for the different pulses. When the leading-edge discriminator fires, the level of the CFD equation *at that time* is saved as a value LOCAL_ZERO. The CFD state machine, once armed, calculates the CFD equation and subtracts from that the value of LOCAL_ZERO. Note that in Figure 8 the “leading edge arming” signal has no relationship to the Y axis. The “leading edge arming” is arbitrarily scaled to highlight the times when the CFD calculation is sampled to save LOCAL_ZERO. The sign of that subtraction is saved on the first clock that the CFD state machine runs, and the CFD firing is declared when the sign of (CFD_equation – LOCAL_ZERO) changes. This results in consistent CFD operation even as the DC level of the CFD equation wanders about due to the presence of other pulses – even during pileup.

Time interpolation using the CFD Sample values

Three CFD Sample values are provided in the CFD-mode header to allow interpolation of the timestamp. The CFD equation [$X_{(n)}$ * fraction] – $X_{(n-d)}$ for a positive-going pulse is expected to rise and then plummet through zero to a negative value as the input passes the fractional percentage value desired. The CFD Sample values are the values of the CFD equation at the clock tick when the timestamp is latched, the sample one clock tick earlier and the sample two clock ticks earlier. The expectation is that there will be two positive samples and one negative sample captured for a positive-going input pulse.

These three values are entered into a linear regression and the intercept (where the fit line equals zero) determined. The non-zero fractional ‘x’, or time, value at Y=0 is subtracted from the timestamp reported to obtain the interpolated timestamp. The accuracy of the interpolation method is highly dependent upon the rise time of the incoming signal, and secondarily dependent upon the amplitude of the incoming signal. This is simply because the CFD timing accuracy is proportionate to the slope of the effective differentiator, so faster rising inputs yield a greater slope and thus less time (x) error. At small signal/noise ratios the fit gets “fuzzy”. Test stand results obtained using signals with 800-1000ns rise times are a 1-sigma normal distribution of interpolated time results of about 1.7ns for relatively large signals, degrading to a 1-sigma of about 2.5ns for tiny signals at the limits of leading-edge discriminator detection.

Detecting improper CFD operation

For very closely spaced pulse pairs it is possible that the hold-off time set for the leading-edge discriminator may fall during the rise of the second pulse, such that the ‘local zero’ value of the CFD equation is latched late. In this case the CFD equation may fire but not at the desired fraction of the pulse. The risk of this can be minimized by using the early termination of the hold-off time when the peak is found. Events in which the ‘local zero’ has been sampled at a late time are normally identified by examining the sign of the three CFD interpolation samples. All correct CFD firings will have the sign of the first two samples the same with the third either being zero or of the opposite sign of the first two. Erroneous ‘local zero’ sampling typically results in all three CFD interpolation samples having the same sign.

Details of Pileup Logic

Pileup is based upon a counter that increments every time the discriminator fires and decrements every time that a delayed copy of the discriminator bit falls out of a delay chain formed by buffers M2 (pre-rise), K and K0, as shown in Figure 9.

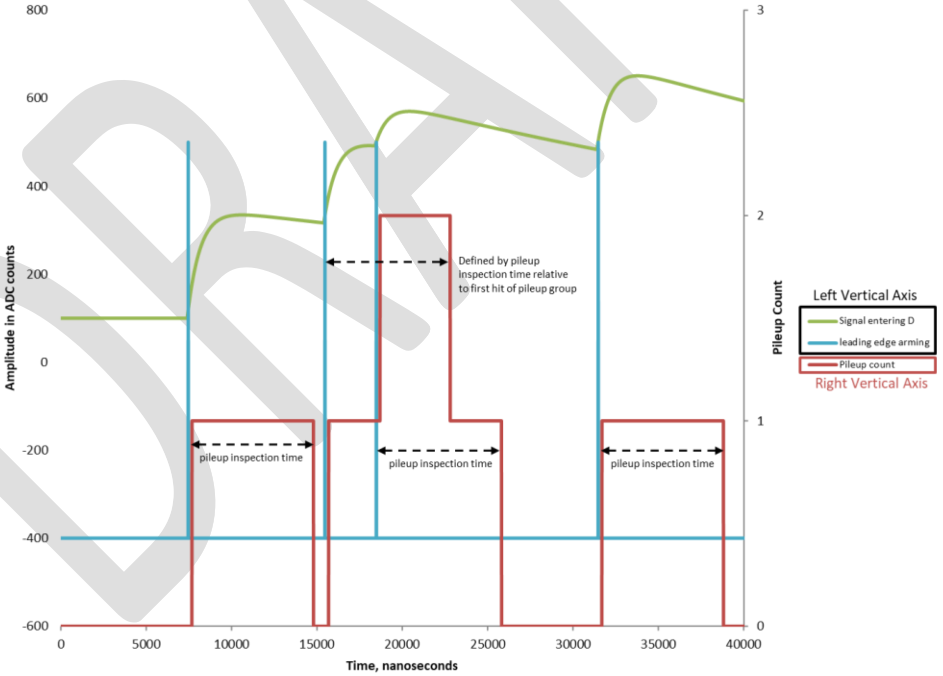


Figure 9 - Pileup counter operation from simulation of firmware operation.

The counter is incremented when the discriminator fires and decrements when a copy of the discriminator bit, delayed by the pileup inspection time 'm'+'k0'+'k', exits the delay chain. This is shown in Figure 10. Since the delay chain is a long shift register, this means that there can be an arbitrary number of 1s in the delay chain; the limiting factor is the discriminator holdoff time. A secondary limiting factor is that the pileup counter is only four bits, so there is another limit of no more than 15 discriminator hits during the pileup inspection time. In this exceptional case, since the pileup counter has overflowed, the channel pileup logic will seize up into a *pileup overflow* state. This state is detectable by reading status bits in the master status register, requiring software or user intervention to reset the channel logic and the pileup counter. With HPGe detectors using an 'm' of 6us, the counter limit sets the maximum hit rate at one hit every 400ns, twice as fast as the typical rise time of HPGe signals. A properly set holdoff time ending at or near the peak will ensure no pileup counter overflow.

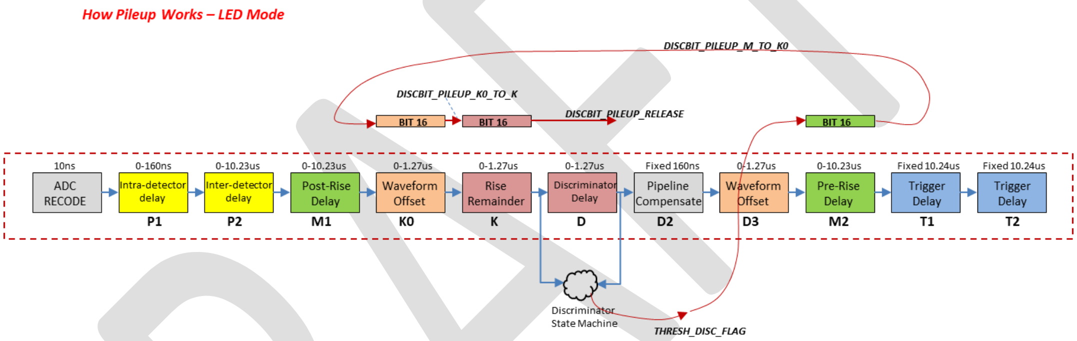


Figure 10 - Use of delay buffers M2, K and K0 to form pileup delay in the *leading-edge* mode.

The picture is the much the same in CFD mode. As shown in Figure 11, the pre-arming leading-edge discriminator moves to look at the signal across ‘k’ rather than ‘d’, but the pileup timing remains the same. The reason behind this is that the leading-edge discriminator ***must*** work for the constant-fraction discriminator to work, but if the user sets incorrect CFD parameters it is possible for the leading-edge to fire but the constant-fraction to sometimes fail. Thus, it is safer to always base pileup timing upon the leading-edge discriminator.

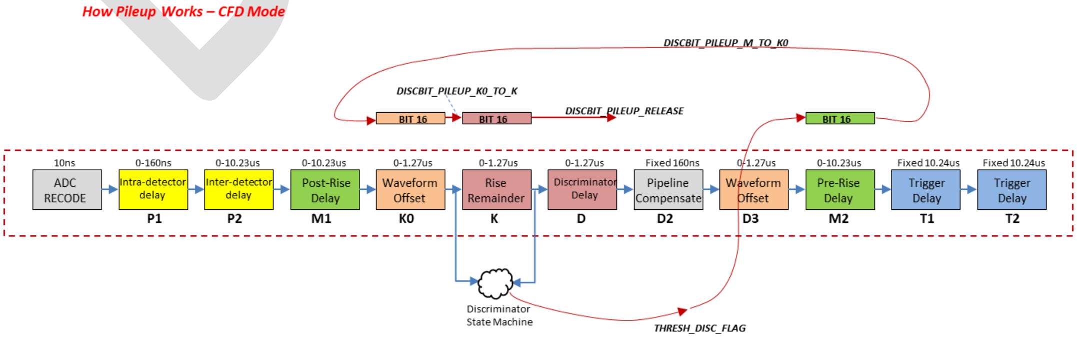


Figure 11 - Use of delay buffers M2, K and K0 to form pileup delay in the constant-fraction mode.

A properly scaled diagram of a waveform relative to these buffers relative to a hand-sketched waveform provides a bit more context. In Figure 12, the waveform and the buffers are shown in “oscilloscope view” to ease understanding; the order of the delay buffers is reversed left-right relative to previous pictures. The first edge of the waveform is shown aligned with the edge of the ‘d’ buffer as would be the case using the threshold discriminator. A minor adjustment relative to actual firmware construction, to simplify the drawing, shows the edge aligned with the left edge of ‘d’. In actuality, the filter and discriminator logic require a few clocks to function, so by the time the discriminator bit fires, the waveform would be at the left edge of ‘d2’; but for simplicity we will ignore that delay as it occurs on both sides of the pileup equation and cancels out for the purposes of Figure 12.

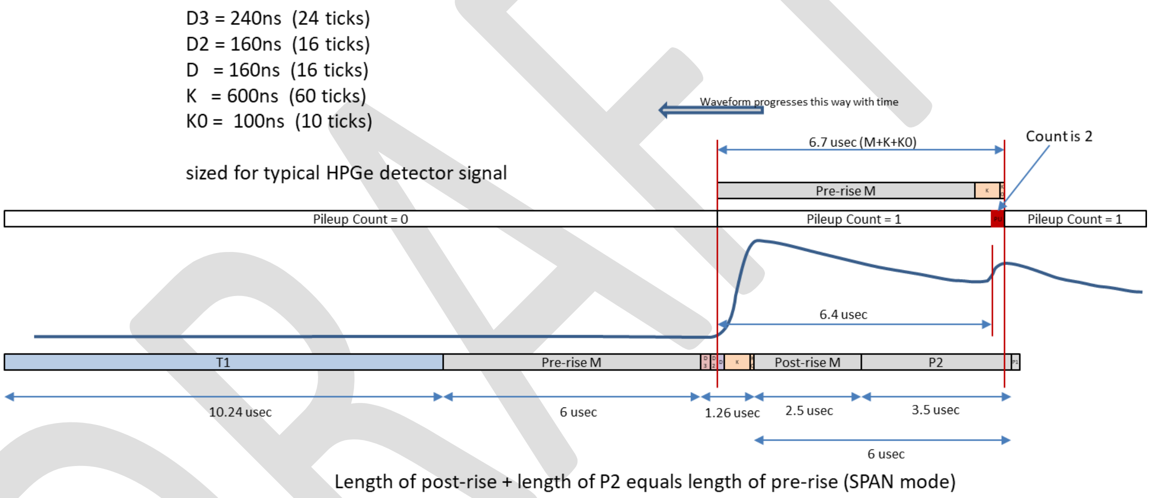


Figure 12 - Pileup operation, with horizonal axis scaled to actual time.

At the time of the discriminator firing for the first edge the pileup counter increments from 0 to 1. This enters the delay buffer chain shown in Figure 4 as “DISCBIT”. The discriminator fires a second time 6.4usec later, incrementing the pileup counter from 1 to 2. 6.7usec after the first edge the counter decrements from 2 to 1, and another 6.7usec after that (outside of picture) the pileup counter would again decrement from 1 to 0.

It is important to note here that *Discriminator Hits* become *Accepted Hits* based upon the value of the pileup counter *when the DISCBIT_PILEUP_RELEASE bit associated with the hit falls out of the delay buffer.* This means that the *first* hit in the “train” of piled-up hits *has been marked as piled upon before it is potentially accepted* **plus** *every piling-on hit is also marked as being in pileup.* The readout logic uses the **value** of the pileup counter to differentiate the *first* hit in a pileup train from all the other hits in the same train, marking the *second* and all following as *Extended Events* (if accepted), as opposed to the *first*, that if accepted becomes the *Accepted Hit*.

Handling of Pileup

The firmware may be set to either accept or reject pileup. If pileup is allowed, many different readout options are available that handle pileup in different ways; this will be discussed later in this document when the focus changes to the readout logic. If pileup rejection is enabled, then all *discriminator hits* that are in pileup are discarded. When pileup rejection is in play, *discriminator hits* not in pileup become *accepted hits* and there can be no *extended events.*

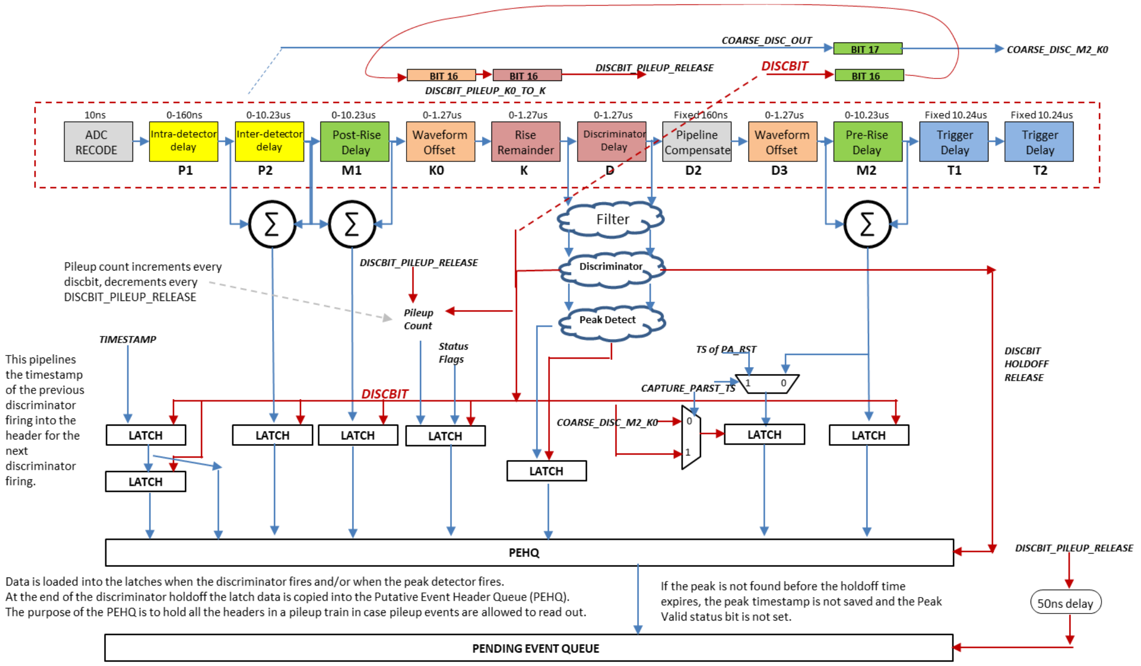
Relationship of pileup to event header formation

Event headers are captured by the digitizer firmware in stages. This timing is critically important to understanding the full operation, so please read carefully.

At the time of the discriminator firing, most of the data in the header is known, but not everything. A very wide latch captures most header information (timestamp, energy sums, etc.)

when the discriminator fires. During the time of the discriminator holdoff, this latch can accumulate additional information, so if a peak is found after the discriminator fires, the timestamp of the peak and/or its value can also be saved in this latch. At the time the discriminator holdoff expires, the data in this latch is copied to a short but wide FIFO called the Putative Event Header Queue, or PEHQ.

Extreme care with terminology is required here. The PEHQ data is named *putative* because at the time the PEHQ is written, the data in the PEHQ may or may not survive to become an *Accepted Hit* because at that moment the pileup status is not yet known. Typically, the discriminator holdoff time is set to something around 1.5usec. Referring to Figure 12, this time is much shorter than the pileup inspection time, calculated earlier to be 6.7us. The putative header is loaded into the PEHQ 1.5usec after the discriminator fires (holdoff time) but until the 6.7us pileup inspection time expires the pileup status will not be certain. The *putative* header resides in the PEHQ for that 5.2usec. A second header is loaded into the PEHQ 6.7usec after the first one, so in the example of Figure 5 there's never more than one event in the PEHQ. However, for longer values of 'm' combined with short holdoff times multiple putative headers could be in that buffer simultaneously. For example, with 'm' = 10.24usec combined with a faster input signal that only needs a 'k'+'k0' of 400ns (resulting in a likely holdoff time of ~1usec) there could be 10 events in the PEHQ if the hits come in a fast sequence just slower than the holdoff time would block. The PEHQ has a depth of 16, so should be capable of withstanding almost any normal experimental situation. See Figure 13.



Pileup count increments every discbit, decrements every DISCBIT_PILEUP_RELEASE

This pipelines the timestamp of the previous discriminator firing into the header for the next discriminator firing.

Data is loaded into the latches when the discriminator fires and/or when the peak detector fires.
At the end of the discriminator holdoff the latch data is copied into the Putative Event Header Queue (PEHQ).
The purpose of the PEHQ is to hold all the headers in a pileup train in case pileup events are allowed to read out.

If the peak is not found before the holdoff time expires, the peak timestamp is not saved and the Peak Valid status bit is not set.

50ns after the pileup inspection time expires, the putative header in the PEHQ is copied to the Pending Event Queue (PEQ)
If the event is accepted by the pileup processor (that is, it is an *Accepted Hit).*
Only the Accepted Hits enter the PEQ for possible selection by the trigger for readout. The other headers in the PEHQ (*Extended Events)* get copied from the PEHQ into the Event Header FIFO depending upon the readout mode selected. If the readout of any *Extended Event* does not occur due to *Readout Interference* or because the mode does not use them, every DISCBIT_PILEUP_RELEASE the *Extended Event* is popped off the PEHQ so that the PEHQ never fills.

Figure 13 - The PEHQ and Pending Event Queue (PEQ) as related to the channel pipeline.

PEHQ latch timing requirements

The PEHQ ***latch*** must collect information starting “sometime” before the discriminator fires (i.e. the “early” pre-rise energy sums), and must finish collecting before the end of the holdoff time (because that’s when the latch is written into the PEHQ). However, the different items within the PEHQ become valid at different times within the event. The PEHQ latch solves this by writing and re-writing certain latch values using different signals derived from the various state machines. The *clock* to all latches in the PEHQ is the 100MHz clock that runs continuously, but the *clock enable* to any given portion of the PEHQ latch may occur zero, one or multiple times during the formation of the event.

For example, one latch in the PEHQ holds the CFD_VALID bit that comes out in the header. That latch is loaded with ‘0’ every time the leading-edge discriminator fires, but a separate clock enable term allows a ‘1’ to be written if later the CFD fires.

The majority of PEHQ latch items are loaded when the leading-edge or constant-fraction discriminator fires, but some latches load at other times. The “early pre-rise energy” header value is sampled when the coarse discriminator fires, well before the main discriminator bit can occur. Similarly, a *delayed* copy of the coarse discriminator bit is used to capture a *third* copy of the pre-rise buffer sum ‘m’ + ‘k0’ *after* the coarse discriminator fires, a time that will be close to the timing of the main discriminator but still reliably before then. Because the timing of the coarse discriminator is not fixed relative to the main discriminator, there must also be a latch that captures some bits of the timestamp count when the coarse discriminator fires so that the delta-t between these early pre-rise sums and the normal pre-rise sum can be calculated for every event. Another example of this is the timestamp of the peak, that obviously must be sampled only when the peak is found.

Effectively, then, the PEHQ latch begins collection when the holdoff time of event(n) occurs, and collection ends when the holdoff time of event(n+1) occurs. With real signals, this is a variable number.

Holdoff time versus pileup time

Since the PEHQ latch gets written to the PEHQ at the end of the holdoff time, and the PEHQ is unloaded at the end of the pileup inspection time (‘m’+’k0’+’k’), there is a requirement that *the holdoff time must always be shorter than the pileup inspection time.* If this is violated, the PEHQ underflows and the digitizer firmware cannot function. To avoid this condition, the firmware implements a “pileup too short” process that will set an error bit PU_TIME_ERROR_FLAG if the pileup time is shorter than the holdoff time. If this bit is set the readout logic is prevented from operating, causing there to be no data from the digitizer. A bit in a status register is also set when this condition occurs.

The condition prevents digitizer operation, and the only way to resume operation is to adjust the holdoff and/or pileup time to meet the requirement and then re-initialize all channels. A complete power cycle reset is not required, just a soft reset of the logic in all channels will suffice.

Event Formation at the end of the pileup inspection time

When the pileup inspection time elapses after a discriminator hit, two things happen.

1. The *putative* header stored in the PEHQ *may* be transferred into the Pending Event Queue (PEQ), becoming an *Accepted Hit,* depending upon whether pileup events are allowed or not, and
2. The *Accepted Hit*, if formed, sets a discriminator bit that is transmitted to the trigger system for the formation of multiplicity triggers and for clean/dirty inspection.

In every channel the channel control register contains a bit that selects whether hits marked as pileup can become *Accepted Hits* or not. As noted previously only the *first* hit in a pileup train can become the *Accepted Hit.* The rest of the hits in the pileup train are accepted or rejected as a group based upon the selection applied to the first hit. As Figure 7 shows, there are two parallel paths of information after the PEHQ. The *timestamps* of the accepted hits enter the PEQ for possible selection by the trigger (thus changing from *accepted hits* to *accepted events)* whereas the header data from all hits (*accepted hits* and, if pileup is allowed, *extended events*) transfer from the PEHQ to the Event Header FIFO.

When pileup rejection is ON, no hit in a pileup train will become an *Accepted Hit*. The logic that transfers headers from PEHQ to Event Header FIFO uses *Accepted Hit* and/or *Extended Event* timing to both advance the PEHQ and write the Event Header FIFO.

- If pileup rejection is ON, the putative headers of the entire pileup train stored in the PEHQ are popped out of the PEHQ and never written to the Event Header FIFO at the end of the pileup inspection time.
- If pileup rejection is ON, *Accepted Hits* not marked as pileup *are* copied from PEHQ to the Event Header FIFO at the end of the pileup inspection time and simultaneously popped out of the PEHQ.
- If pileup rejection is OFF, all *Accepted hits* and *Extended Events* are copied from PEHQ to the Event Header FIFO, but still at the end of the pileup inspection time.

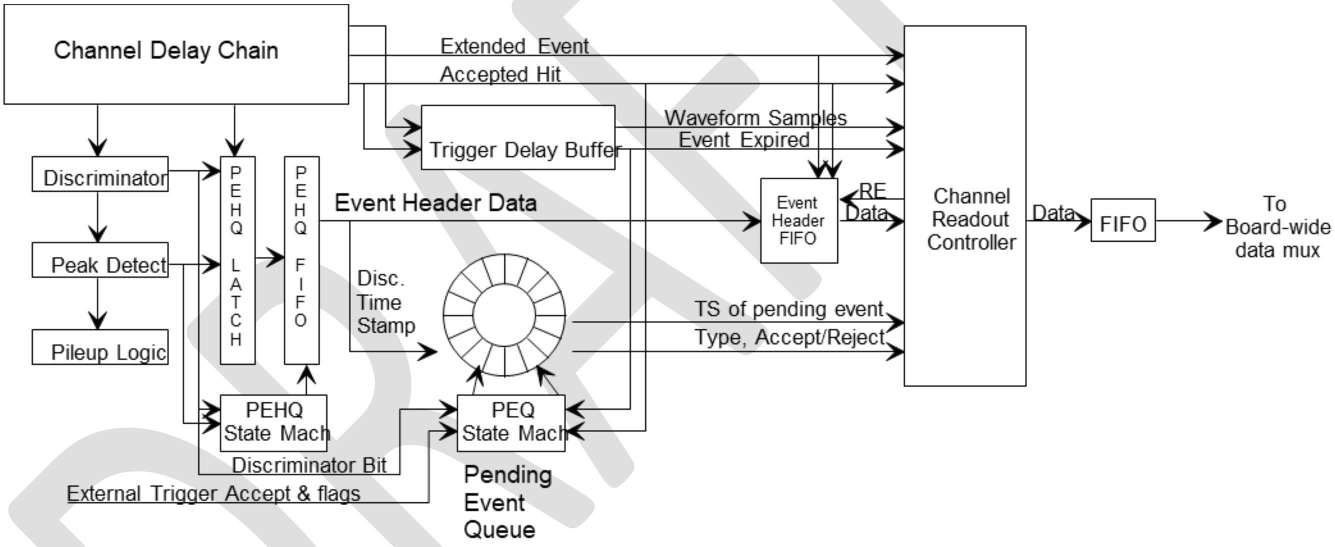


Figure 14 - relationship between pipeline, PEHQ, PEQ and Event Header FIFO.

Triggered modes vs. "Internal Accept All"

The Pending Event Queue is used only when the firmware is set to select events for readout based upon acceptance messages from the external trigger system. If the firmware is set to "internal accept all" the PEQ is bypassed, and all *accepted hits* immediately become *accepted events*. In the "TTCL" mode, a 2nd level of accept/reject of headers in the Event Header FIFO occurs at a time called **Event Expiry.**

Event expiry

An *Accepted Event* expires when the waveform data associated with the *Accepted Event* can no longer be read out. This timing is defined as the length of the Trigger Delay Buffer (T1 plus T2 viewed as a single delay element). See Figure 8. Every *Accepted Hit* inserts a 10ns wide pulse into the T1+T2 buffer set, that exits 20.48us later as ***EVENT_EXPIRED***. Since the hit is only consuming one of the 2048 cells of this digital delay buffer, *many hits can be simultaneously active*. Every tick of the clock, if ***EVENT_EXPIRED*** is asserted, the Event Header FIFO is popped and the eldest event in the PEQ marked as no longer valid for selection. These actions render the event no longer available for readout.

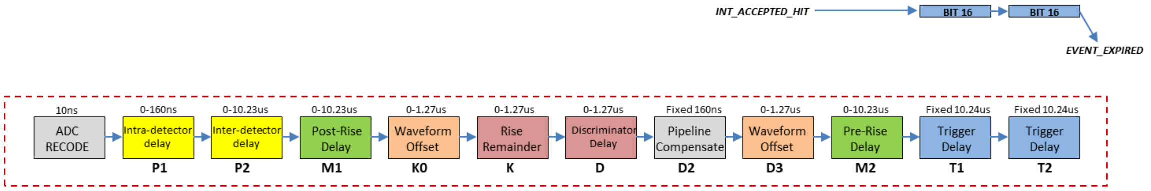


Figure 15 - timing of event expiration.

The operation of the PEQ, with events loaded by *Accepted_Hit* and events unloaded by ***EVENT_EXPIRED***, results in the PEQ always containing the list of timestamps of all events whose waveform could be read out because they are still within the T1+T2 buffers (collectively referred to in other documentation as the *Trigger Delay Buffer*).

Event status bits in the PEQ

Every entry in the PEQ has, in addition to the timestamp of the discriminator, status bits including

- a three-bit placeholder for the trigger *type* as broadcast in the acceptance message from the trigger;
- a *pending* bit that marks the PEQ entry as “active” but not “selected”.
- an *accepted* bit that marks the PEQ entry as “selected”. PEQ entries with the *accepted* bit set should read out
- a *ReadOutMachineSignaled*, or “ROMS”, bit that marks the PEQ entry as one that has been signaled to the readout logic.

An event in the PEQ whose *pending* bit was set but whose *accepted* bit was never set is simply one that dies because the trigger never selected it. No special action occurs in the PEQ at the time of ***EVENT_EXPIRY*** other than the removal of the pending event from the queue. The PEQ does not in any way mark, cause or count *dropped events*; this is a function solely of the readout logic.

External Discriminator Modes

The digitizer firmware provides a selection of different *external discriminator* signals, of which ***one*** may be chosen as the *external discriminator source* per channel. Then a 2nd set of selection logic in each channel allows the user to select whether the *external discriminator source* signal is

- ignored
- ANDed with main discriminator signal (i.e. external *gates* the main)
- ORed with main discriminator signal
- is used *in place of* the main discriminator

This selection logic is shown in Figure 16.

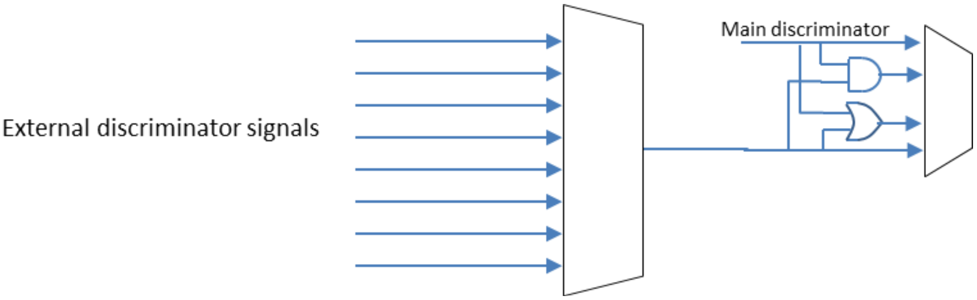


Figure 16 - External discriminator selection logic of one channel.

One of the external discriminator signals is derived from the timestamp and is itself multiplexed to provide different periodic signals.

External Discriminator Signals

The selection of external discriminator signals varies slightly between channels, and also upon whether the digitizer is a ”master” digitizer or a “slave” digitizer. The selection matrix is provided in Table x.

External Discriminator Select	***“Master” digitizer***			***“Slave” digitizer***		
	channel 9	channels 5-8	channels 0-4	channel 9	channels 5-8	channels 0-4
0	‘0’	channel 9	channel 9	‘0’	channel 9	channel 9
1	‘0’	‘0’	‘0’	ch. 9 of mstr dig	same ch. of mstr dig	same ch. of mstr dig
2	front panel	front panel	front panel	front panel	front panel	front panel
3	TS bit	TS bit	TS bit	TS bit	TS bit	TS bit
4	VME command	VME command	VME command	VME command	VME command	VME command
5	‘0’	‘0’	if channel (n+5) is killed by Preamp Reset	‘0’	‘0’	‘0’
6	command from trig. if selected	command from trig. if selected	command from trig. if selected	command from trig. if selected	command from trig. if selected	command from trig. if selected
7	any other channel	any other channel	any other channel	any other channel	any other channel	any other channel

Table 1- External discriminator source matrix.

A few of the entries in Table 1 require additional explanation. The “front panel” option is an RS-485 input that is part of the Auxiliary I/O of the digitizer module. There are 11 bits of Auxiliary I/O on the front panel. The MSbit (AUX_DIN[10]) is the bit used for external discriminator activity. The Auxiliary I/O connector has multiple modes of operation as described in a later section of this document; in some of them, bit 10 may be an output, not an input. In those modes the front panel bit cannot be used as part of external discriminator logic.

The ”command from trigger if selected” entry means that there is a command that the trigger system can send during Frame 15 of the 20 frame trigger command cycle. When the trigger sends a non-null command in Frame 15, the command can have various command *values*, of which one value means “assert external discriminator bit”. When this occurs, the command from the trigger will also contain a *selection mask* of which channels in the digitizer are to have their external discriminator bit asserted. This allows the trigger to selectively create discriminator firings in different channels of every digitizer in the system for the purpose of verifying the cabling plant and all sources of multiplicity.

The VME command mode is like the trigger command mode but does not have system-wide synchronicity like the trigger command. The source of the external discriminator bit is from a write-only “pulsed control” register; only those channels set to this mode will receive the external discriminator signal. This mode is intended for diagnostics in digitizers not connected to trigger systems.

Timestamp-based external discriminator

The timestamp bit option connects the external discriminator bit to a separate multiplexer that selects one of seven different timestamp bits as the external discriminator signal. The timestamp counter generates 10ns wide pulses at rates of

- 0.745Hz (TS bit 26)
- 6.0Hz (TS bit 23)
- 23.8Hz (TS bit 21)
- 95.4Hz (TS bit 19)
- 1.5kHz (TS bit 15)
- 48.8kHz (TS bit 10)
- 195.3kHz (TS bit 8)

When the timestamp bit is selected as the external discriminator source and the channel external discriminator mode is set to “External only”, the digitizer channel acts like an oscilloscope in auto-trig mode, regularly sampling the input irrespective of signal features. This is a convenient way to measure baseline levels. Alternatively, if the external discriminator mode is set to “OR” the timestamp bit may be used as a background “heartbeat” to force a discriminator hit every so often but also capture edges. This type of operation is useful for low-rate detectors (e.g. neutrino experiments) to ensure that data is regularly generated and to provide background monitoring. The higher speed timestamp bits can be used to test systemic bandwidth to exercise readout software or to test pileup response with known timing. When connected to the trigger system, timestamp-based external discriminator operation is also useful to measure variations in cable delay within the trigger cable plant that may affect triggering distinct from total delay measurements that would be done with test pulse circuits built into detector preamplifiers.

Insertion of timing marks into the waveform

As the ADC chips of the digitizer module provide 14 bit data but the VME readout is 32-bit words, the obvious utilization of a 32-bit word is two 16-bit halves. In each 16-bit half the ADC data value consumes bits 13:0 leaving bits 14 and 15 free for other use. Bit 14 is used to inject *timing marks* into the data so that waveform display software can show when, relative to the waveform itself, different things occur. The obvious things to highlight would be

- when the discriminator fires
- when the peak is found
- when the discriminator holdoff elapses
- when the pre-arming threshold discriminator fired to arm the constant-fraction discriminator
- when the coarse discriminator fired
- when the energy values were sampled

However, since there is only *one* bit in each 16-bit waveform value, a somewhat more creative solution must be found. The solution chosen is to *serialize* the timing mark over a fixed number of waveform data samples and to identify the feature by the n-bit serial data value. To minimize the risk of these timing marks interfering with each other, this value ‘n’ was set to three. Software must also be able to unambiguously identify when a timing mark starts, so the decision was made to set bit 14 to zero as default, such that the start of the timing mark is defined by bit 14 transitioning from zero to one. There are four unique three-bit codes that start with 1:

- 100
- 101
- 110
- 111

This means that there can only be four unique times marked per discriminator firing. A multiplexer is defined that changes which 3-bit code has what meaning based upon whether the discriminator is in threshold or constant-fraction mode, with a sub-selection in constant-fraction mode based upon the user’s selection of whether the pre- and post-rise sums are to be sampled at the time of the arming threshold discriminator or that of the constant-fraction discriminator. This gives us the following table.

		in constant-fraction mode	
timing mark code	in threshold mode	energy sampled at threshold discriminator firing	energy sampled at constant fraction firing
100	discriminator fired	CFD is armed	CFD is armed
101	delayed copy of coarse	delayed copy of coarse	delayed copy of coarse
110	holdoff time end	when energy sampled	when energy sampled
111	peak found	peak found	peak found

Timing mark “101” is a *delayed* copy of the coarse discriminator, because if the mark was placed against the waveform where the coarse discriminator actually fires (~P2 + M + K + K0 + D before the threshold discriminator fires, in threshold mode), the mark would be placed so far before the rest of the waveform that most of the time it would never be seen. To compensate, the coarse discriminator bit is delayed by M (CFD mode) or M+K0+K (threshold mode) so that the timing mark for the coarse discriminator will be issued before the rising edge, but by a small enough amount to usually be in the picture. The user may subtract the known delay to estimate where the coarse discriminator actually fired.

Timing marks and long waveform readouts

If long waveform readouts are used, it is quite possible that multiple discriminator hits may be in one waveform. Timing marks are issued for every discriminator firing, so in this case software has to interpret multiple sets of timing marks.

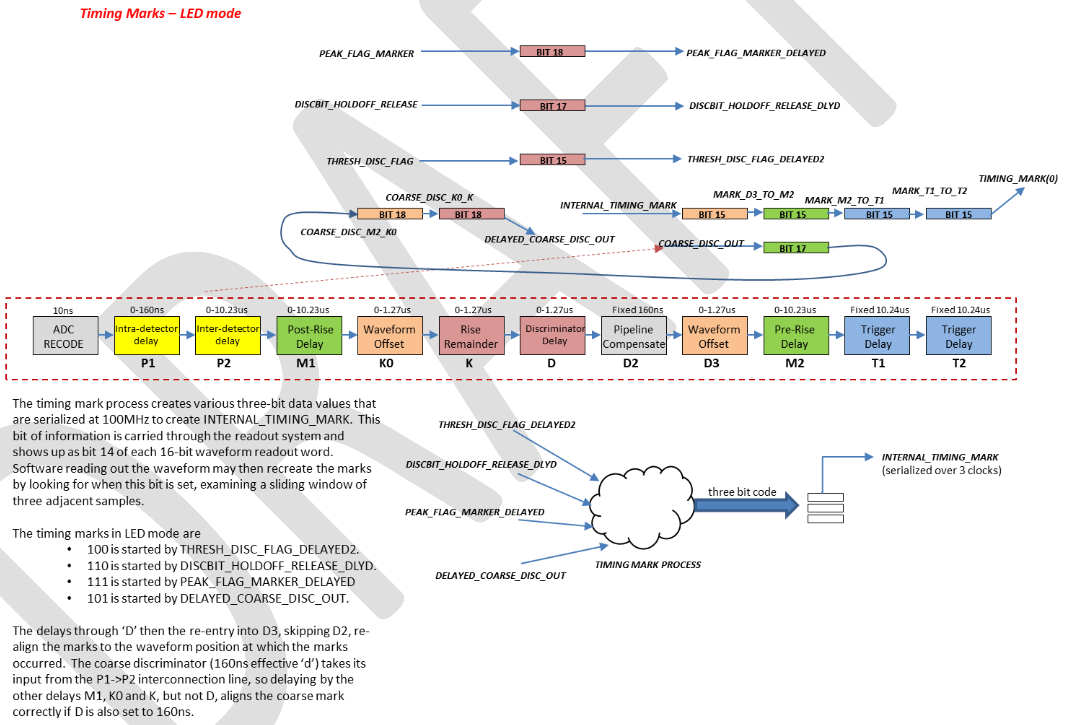
Timing marks and down-sampling

Timing marks are generated in the 100MHz clock domain and are subject to effects from down-sampled readout. When down-sampling at any factor is in play, the 1’s of the timing marks are buffered into a small FIFO that runs in parallel with the down-sampling function. This FIFO loads the timing marks in the 100MHz clock domain and unloads them at the down-sampled rate to match the waveform. This results in the timing mark data sequences with contiguous groups of 1s (100, 110, 111) being preserved for small down-sampling factors but means that at any down-sampling the “101” timing mark becomes indistinguishable from the “110” timing mark.

Additionally, with down-sampling in play, each timing mark data sequence stretches over more and more data samples as viewed in the 100MHz clock domain. As the down-sampling factor is increased, the timing marks overlap each other such that illegal timing mark sequences such as “11111111” might be seen in the down-sampled data. In general, timing marks can still be useful for down-sampling values up to 2 or maybe 3, but at any higher down-sampling factor the timing marks lose all value. Of course, if the down-sampling is set to the “on-off” mode where down-sampling is paused from the mark of the delayed coarse discriminator to the end of the down-sample holdoff time, timing marks in the full-speed “off” regions will still be rational.

Formation of timing marks

The logic for building the timing marks varies between threshold discriminator mode and constant-fraction discriminator mode, so two different diagrams are required.



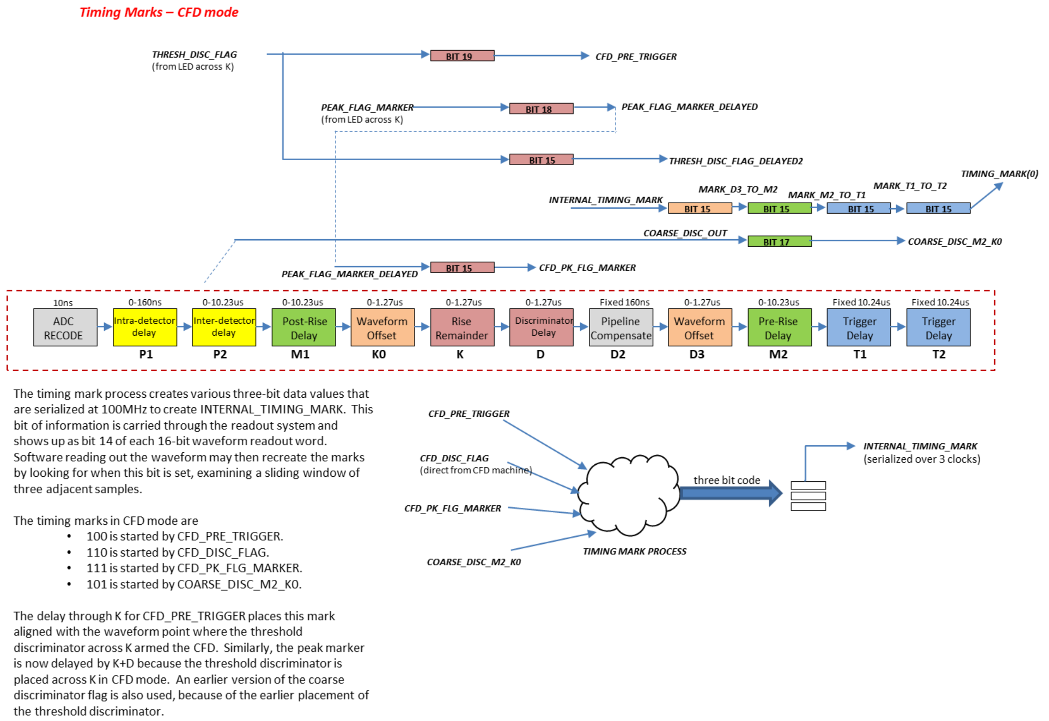
The timing mark process creates various three-bit data values that are serialized at 100MHz to create INTERNAL_TIMING_MARK. This bit of information is carried through the readout system and shows up as bit 14 of each 16-bit waveform readout word. Software reading out the waveform may then recreate the marks by looking for when this bit is set, examining a sliding window of three adjacent samples.

The timing marks in LED mode are

- 100 is started by THRESH_DISC_FLAG_DELAYED2.
- 110 is started by DISCBIT_HOLDOFF_RELEASE_DLYD.
- 111 is started by PEAK_FLAG_MARKER_DELAYED
- 101 is started by DELAYED_COARSE_DISC_OUT.

The delays through 'D' then the re-entry into D3, skipping D2, re-align the marks to the waveform position at which the marks occurred. The coarse discriminator (160ns effective 'd') takes its input from the P1->P2 interconnection line, so delaying by the other delays M1, K0 and K, but not D, aligns the coarse mark correctly if D is also set to 160ns.

Figure 17 - Signal delays and logic of timing marks in threshold discriminator (LED) mode.



The timing mark process creates various three-bit data values that are serialized at 100MHz to create INTERNAL_TIMING_MARK. This bit of information is carried through the readout system and shows up as bit 14 of each 16-bit waveform readout word. Software reading out the waveform may then recreate the marks by looking for when this bit is set, examining a sliding window of three adjacent samples.

The timing marks in CFD mode are

- 100 is started by CFD_PRE_TRIGGER.
- 110 is started by CFD_DISC_FLAG.
- 111 is started by CFD_PK_FLG_MARKER.
- 101 is started by COARSE_DISC_M2_K0.

The delay through K for CFD_PRE_TRIGGER places this mark aligned with the waveform point where the threshold discriminator across K armed the CFD. Similarly, the peak marker is now delayed by K+D because the threshold discriminator is placed across K in CFD mode. An earlier version of the coarse discriminator flag is also used, because of the earlier placement of the threshold discriminator.

Figure 18 - Signal delays and logic of timing marks in constant-fraction (CFD) mode.

"GammaWare", the test stand program developed at Argonne for testing all elements of the digital data acquisition systems, has been coded to process and identify timing marks in waveform data as shown in Figure 19.

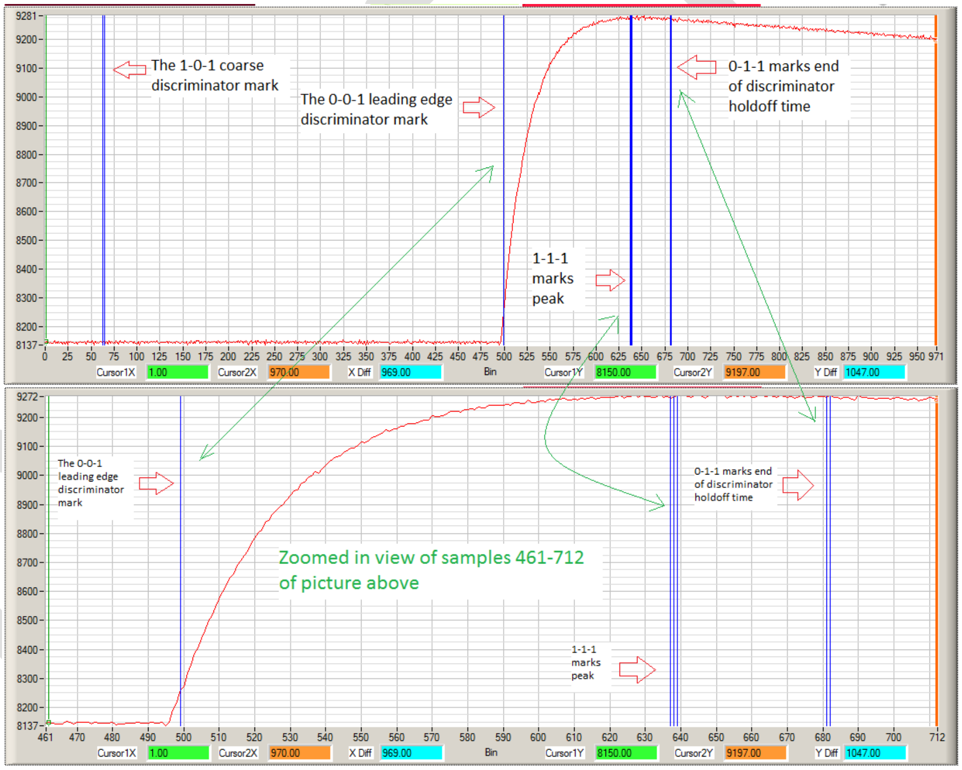

The 1-0-1 coarse discriminator mark
The 0-0-1 leading edge discriminator mark
0-1-1 marks end of discriminator holdoff time
1-1-1 marks peak
The 0-0-1 leading edge discriminator mark
0-1-1 marks end of discriminator holdoff time
Zoomed in view of samples 461-712 of picture above
1-1-1 marks peak
Cursor1X 1.00
Cursor2X 970.00
X Diff 969.00
Bin
Cursor1Y 8150.00
Cursor2Y 9197.00
Y Diff 1047.00

Figure 19 - Timing marks in waveforms as displayed in GammaWare.

Use of the timing marks to understand waveforms and firmware operation is highly recommended. A similar GammaWare display, Figure 20, overlays rectangles whose width is proportionate to their delay time along with the timing marks. This display helps clarify the interplay between timing marks and the delay buffers. The GammaWare program displays captured waveforms the way they would look on an oscilloscope, so the order of buffers left-to-right is reversed from the firmware-centric views shown previously (e.g. Figure 5, Figure 7, etc.).

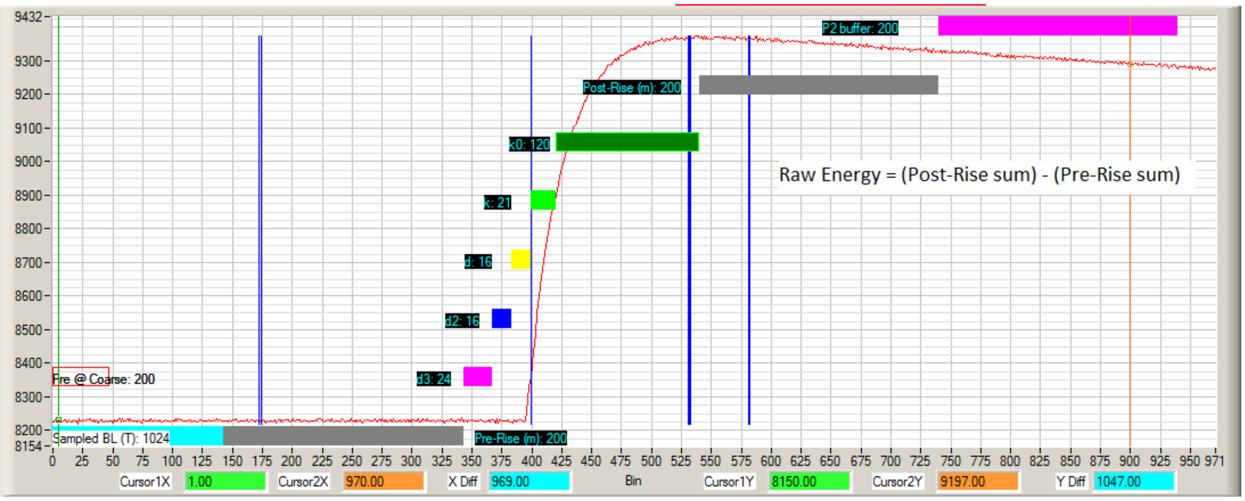


Figure 20 - GammaWare display showing both timing marks and delay buffer sizes overlaid upon waveform.

The primary observation is that the group of blocks ***D3, D2, D, K*** and ***K0*** are sized so that, collectively, they span the entire rise of the waveform. ***D2*** is a fixed delay required for timing that is not programmable by the user. A standard practice is to set these delays such that the Post-Rise buffer starts at or immediately after the peak detector timing mark. Once that is accomplished, the ***Post-Rise*** and ***Pre-Rise*** sums clearly yield a difference proportionate to the amplitude (energy) of the signal. The P2 buffer is also shown to highlight its relationship to the Post-Rise buffer. If no pileup is occurring, as in the case of this isolated signal, the sum across the P2 buffer may be used in addition to the sum across the Post-Rise buffer to provide a greater integration time.

Energy Summation

Obviously, the simple subtraction of ***Post-Rise – Pre-Rise*** is subject to error from pole-zero effects, baseline drift and offset of the pre-rise sum caused by the decay of previous pulses. Pole-zero and other corrections are better performed by software in data analysis rather than firmware. Studies at Gammasphere have shown that after appropriate pole-zero and baseline corrections derived from statistical analysis of captured waveforms are applied, waveform capture is no longer required, and the sum values saved in the header portion of the data for each event suffices to maintain resolution equivalent to traditional analog methods.

P2 and Energy Sums

With the release of the August 2021 firmware an accumulator function identical to that used for the Post-Rise and Pre-Rise buffers was added to the P2 buffer. This sum is reported in the data header and is logically an extension of the Post-Rise sum. As both the Post-Rise and the P2 sums are reported, analysis software may as desired use the P2 sum for a longer integration time so long as the varying number of samples in post-rise and pre-rise sums are appropriately considered. In systems where digital delay is not needed for timing alignment the firmware provides the sum across P2 for potential use in energy measurement. To further allow tuning of the firmware for the needs of any particular use case, a control bit is provided that modifies the length of the Post-Rise buffer to allow two different ways to sum up the post-rise energy.

- In SEPARATE mode, the length of the Post-Rise buffer is always identical to the length of the Pre-Rise buffer, and these are set by the “m” parameter. The length of the P2 buffer is set by the “P2” parameter.
    - SEPARATE mode is the likely choice for systems that want to use P2 as a delay and/or want the simplest energy calculations.
- In SPAN mode, the length of the Pre-Rise buffer is still defined by “m”, but the length of the Post-Rise buffer is modulated by the length set for P2 such that the length of the Post-Rise buffer *plus the length of the P2 buffer* is “m”, and the length of the P2 buffer is set by the “P2” parameter.
    - SPAN mode is the likely choice for systems that have no need to use P2 as a delay buffer and/or have high rates that may cause pileup. Here the sums across both P2 and Post-Rise may be added together for a “normal” energy sum in non-pileup situations but a lesser energy may be recoverable from using just the shorter Post-Rise in pileup or near-pileup situations.

- Pileup timing is always defined by “m” without reference to “P2”.

Multiple Pre-Rise sums

Careful examination of Figure 20 will show an unfilled red rectangle at far left labeled “Pre @ Coarse”. This shows the effect of using the coarse discriminator to capture an earlier copy of the Pre-Rise sum. Figure 21 shows approximate positions where the early pre-rise sums are collected. The data for the early pre-rise sums is always taken from the accumulator that spans the pre-rise buffer M, only the *timing* of when that sample is captured changes.

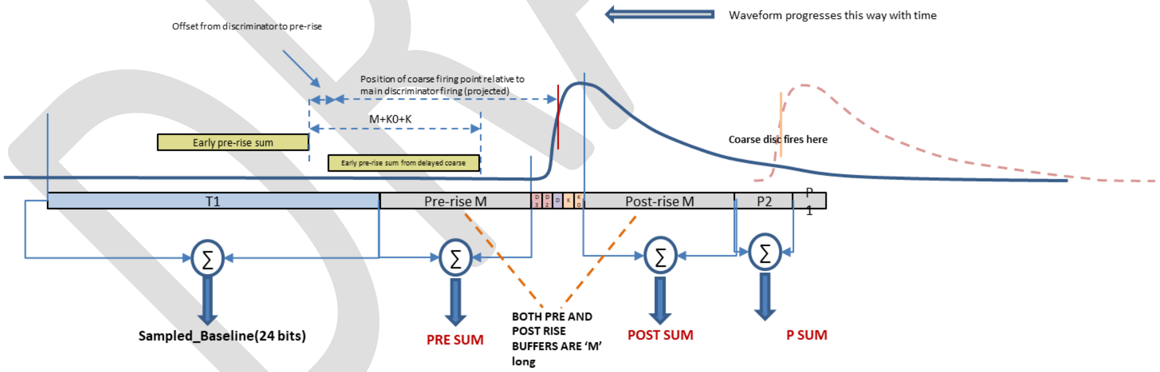

Waveform progresses this way with time
Offset from discriminator to pre-rise
Position of coarse firing point relative to main discriminator firing (projected)
M+K0+K
Early pre-rise sum
Early pre-rise sum from delayed coarse
Coarse disc fires here
T1
Pre-rise M
D3
D2
D
K
K0
Post-rise M
P2
P1
∑
∑
∑
∑
Sampled_Baseline(24 bits)
PRE SUM
BOTH PRE AND POST RISE BUFFERS ARE ‘M’ long
POST SUM
P SUM

Figure 21 - approximate positions where early pre-rise sums are captured.

Operation of the pre-rise sum collection is modified by the value of the CAPTURE_PARST_TS bit in the channel control register. When this bit is 0, the operation is exactly as pictured in Figure 21. When the bit is 1, the second early pre-rise sample (the sample closer to the normal pre-rise sum) in the header of the event is replaced by the timestamp of the last Preamp Reset signal.

Data Readout at the channel level

When pileup is rejected the readout of data is simple, but when pileup events are allowed through the system the readout can become quite complex. How *accepted hits* become *accepted events* has already been introduced as a concept in the section ***Triggered modes vs. “Internal Accept All”*** on Page 22. This section will go into greater detail regarding how the trigger selects specific events within the digitizer and will discuss issues related to readout of piled-up events. Discussion will start with operation when pileup is ***rejected,*** the more common usage.

Readout general operation when pileup is rejected

Data readout begins at the channel level where a ***Channel Readout Machine*** monitors the PEQ and the timestamp. As delay buffers T1 and T2 are defined as each being 1024 samples deep (thus spanning 20.48us of time at a clock of 100MHz), the sample falling out of the channel pipeline is, by definition, a fixed offset in time (based upon user settings of M, D3, D, K & K0) older than the sample currently being examined by the discriminator. By the same token a fixed numerical relationship exists between the data values in the T buffers relative to the timestamps stored within the PEQ.

In “internal accept all” mode, the readout logic operates by performing an arithmetic offset of the current timestamp based upon the delay parameters, and when the adjusted timestamp is equal to the timestamp in the PEQ, readout begins. The reader should note that since the timestamp in the PEQ is the timestamp of the discriminator, *the readout is always aligned to the discriminator firing*. The user has two control values that define the position and amount of waveform data.

- The *waveform offset* value is included in the arithmetic offset calculation that monitors the timestamp. The *waveform offset* sets how many timestamp ticks (waveform samples) *prior* to the discriminator firing the readout begins.
- The *waveform length* value defines *how many* waveform samples are read out. Because the data is packed two ADC samples per 32-bit word, the least significant bit of this value is ignored to force an even number.

Readout in “internal accept all” – pileup rejected

In “internal accept all” mode the PEQ is bypassed; every *accepted hit* is immediately promoted to an *accepted event* and headers are immediately transferred into the channel’s ***Event Header FIFO.*** The readout logic simply waits for every *accepted event* to propagate far enough down the T buffers and then transfers data from the channel’s pipeline into the channel’s ***Accepted Event FIFO*** as the date range selected by the *waveform offset* and *waveform length* falls out of the pipeline. All other ADC samples not within the offset/range of the discriminator timestamps falls out of the channel pipeline and is lost. Referring to ***Figure 14 - relationship between pipeline, PEHQ, PEQ and Event Header FIFO.*** on page 22 is recommended.

Effects of long waveform length settings; defining Readout Interference

The maximum readout length the user may select is 1024 samples and the maximum readout offset is also 1024 samples. Combinations of these two parameters can easily exceed the pileup inspection time (M+K0+K), either before or after the discriminator firing. If the waveform parameters selected by the user is longer than the pileup inspection time, event “n” may still be reading out when event “n+1” becomes ready for readout. This phenomenon is called *Readout Interference*. The readout logic is generally required to finish the readout of the current event before starting the next one, so *Readout Interference* may result in data loss and should be avoided. What will happen is that the readout length demanded will be read, and whatever portion of event “n+1” happens to fall in that length will be in the waveform data, but the *header* for event “n+1” will be lost.

Avoidance of *Readout Interference* when pileup is being rejected is simple. As the pileup inspection time is defined by (M+K0+K), ensuring that the number of waveform samples demanded for readout (the *Readout Length*), plus the number of samples prior to the edge the readout should start (the *Readout Offset*) is less than (M+K0+K+30) allows for insertion of the header and eliminates any possibility of *Readout Interference*.

Down-sampled readout

The user may select down-sampling factors from 0 to 7, meaning that the waveform data will be presented as samples that average together 2^N samples per waveform point (1 to 128 ADC samples per data point for $0 \le N \le 7$). The down-sampling logic acts by using a running accumulator that spans 2^N samples, adjusted every 10ns, but only read out once every 2^N samples. The sum is then shifted right N bits (integer division) and this is the datum provided each down-sampled waveform sample.

The waveform sample data read out by the digitizer consists of two 16-bit halves per 32-bit word, but the ADC data is only 14 bits wide. One of the two extra bits in each 16-bit half identifies transitions between down-sampled and normal speed waveform samples. Down-sampling may be applied to the waveform readout the same for all values or may be set to operate in an “on-off” mode. In the “on-off” mode, data starts out down-sampled, but the logic looks for the *Timing Mark* indicative of the coarse discriminator firing and pauses down-sampling when this is seen. The pause in down-sampling lasts for the number of words defined by a control register, after which down-sampling starts up again.

Readout Interference and down-sampling

When down-sampling is in effect the *time* required to read out a down-sampled waveform increases with the down-sampling factor. This is because the internal processing of down-sampling and packing words into the output data still must use every full-speed data value at the same 100MHz processing rate that is contained in the time span of the down-sampled readout. The system itself does not change speed. Because of this, the risk of *Readout Interference* when down-sampling is enabled is much greater.

Readout of piled-up events

If pileup rejection is turned off, the readout logic must now process all *Extended Events* in addition to *Accepted Hits.* An obvious consequence of this is that the time between discriminator firings will often be quite short, limited now only by the discriminator hold-off time. With typical register settings this means that edges could occur separated by ~1us rather than many microseconds.

Control of how the firmware handles this readout is implemented in two levels. The first level of control is named ***Pileup Extension Enable.***

- When Pileup Extension is ***disabled*** the readout logic is constrained to process only *Accepted Hits* that have become *Accepted Events*. This means that isolated events (no pileup) and the **first** event of a pileup train may potentially be read out, but that the 2nd and later events within a pileup train (the *Extended Events*) will all be discarded.
- When Pileup Extension is ***enabled*** the readout logic will process all *Accepted Hits* that have become *Accepted Events*, ***and*** the logic will also process all *Extended Events* associated with the *Accepted Events.*

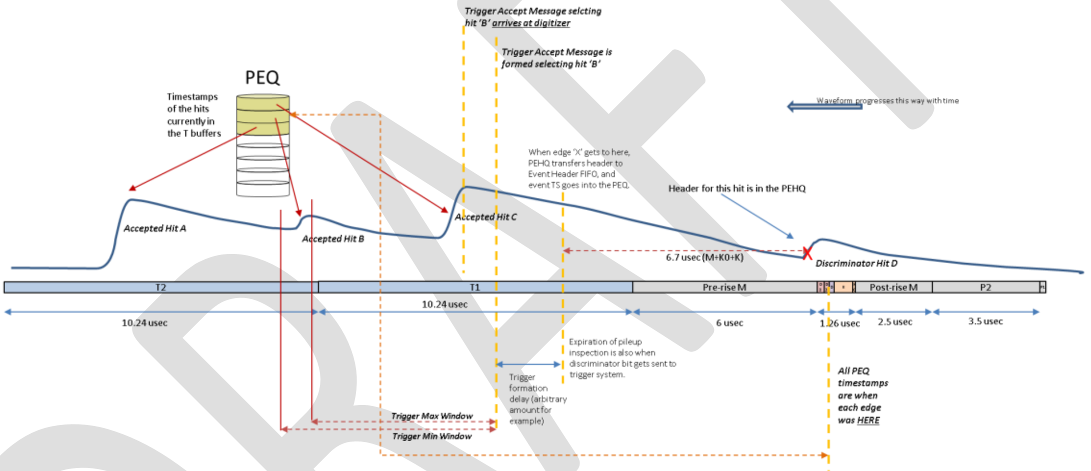
Separately, the user may select one of four possible event readout modes, named ***No Offset, Offset, Offset with Truncation*** and ***Header Only.*** These four modes operate in slightly different ways depending upon whether Pileup Extension is enabled or disabled so the combination of the Pileup Extension and Readout Mode controls effectively generate ***eight*** different readouts. The two sections below describe how each of the four readout modes will operate in each of the two Pileup Extension states.

Details of the trigger window calculation

When a trigger acceptance message is received, all event timestamps in the PEQ are compared against the timestamp value contained within the trigger message. A bit of math is required. The timestamp reported by the trigger is the time at which the trigger module determined that a trigger algorithm was satisfied. If an event occurs at time T_0, the trigger will respond at a variable time T_{tf} later. When processing the events in the pending event queue the timestamps in that queue are the time of the leading edge (T_0).

Upon receipt of the trigger message, the timestamp of the *trigger message* is subtracted from the timestamp of the *firing within the pending event queue* for every firing within the queue. Under normal conditions where the trigger is formed from multiplicity, this will always yield a negative number. However, if the trigger is caused by a different detector that may be faster than the one connected to the digitizers, a positive answer may result. The answer is compared against both upper and lower time window registers (< upper, > lower) and if both comparisons are valid, the event is accepted for readout.

The method of subtraction used means that the window is defined ***with respect to the time the Master trigger detected the appropriate condition*** and not the time of the discriminator firing. That is, if there were an oscilloscope connected to the discriminator signal and to the trigger accept, and the oscilloscope were triggering on the signal from the trigger system, discriminator firings would be seen in a range of times prior to the trigger signal. Thus, the correct settings for the two window registers would also be *negative*, with the '***min*** window' register set to the relative time of the earliest (farthest back from the trigger) discriminator firings and the '***max*** window' set to the relative time of the latest (closest to the trigger) discriminator firings. This apparent reversal of "min" and "max" is because the logic always requires that the "max" window edge be ***later*** in absolute time than the "min" position. See Figure 22.


Trigger Accept Message selcting hit 'B' arrives at digitizer
Trigger Accept Message is formed selecting hit 'B'
PEQ
Timestamps of the hits currently in the T buffers
Waveform progresses this way with time
When edge 'X' gets to here, PEHQ transfers header to Event Header FIFO, and event TS goes into the PEQ.
Header for this hit is in the PEHQ
Accepted Hit A
Accepted Hit B
Accepted Hit C
6.7 usec (M+K0+K)
Discriminator Hit D
T2
T1
Pre-rise M
Post-rise M
P2
10.24 usec
10.24 usec
6 usec
1.26 usec
2.5 usec
3.5 usec
Expiration of pileup inspection is also when discriminator bit gets sent to trigger system.
Trigger formation delay (arbitrary amount for example)
Trigger Max Window
Trigger Min Window
All PEQ timestamps are when each edge was HERE

When the trigger accept message arrives, the timestamp value *contained within the message* is subtracted from the timestamp values in the PEQ. Any subtraction result that falls between the MAX and MIN values chosen by the user is selected for readout.

Figure 22 - Trigger windows within the digitizer

The discriminator bit from any hit is sent to the trigger at the end of the pileup time (M+K0+K). This is shown in Figure 22 as the red 'X' showing the discriminator firing on "Hit D", but the red dashed arrow and note show that this point in the waveform would have proceeded through the buffers into T1 when the discriminator bit is sent to the trigger. For purposes of illustration the yellow dashed lines are shown with reference to when **Hit B** is sent to the trigger system, although it should be clear that *every* discriminator hit is reported when its edge passes the right-most yellow line.

The correct way to view Figure 22 is that the yellow lines and the buffer positions are all fixed, but as time progresses *only the waveform* slides to the left. Figure 22 is showing a snapshot of the waveform position near where **Hit D** is being noted by the discriminator with **Hit A, Hit B** and **Hit C** all having occurred in the past.

Somewhat later, after the *trigger formation time,* the trigger system determines that a trigger-able condition exists, and the timestamp is captured. This is the 2nd yellow dashed line to the left of the 'X'. Finally, there is some transmission latency for the trigger to send the acceptance message to the digitizer. Because of the way the trigger command stream is constructed, the transmission latency is variable; the trigger accept message is queued within a FIFO and is sent when the next available message slot becomes available.

For the cable plant of Gammasphere, using only local multiplicity triggers, the *trigger formation delay* is approximately 0.7usec, including the transmission latency from digitizer to router, router logic, transmission latency to master trigger and master trigger logic. Triggers that require coincidence with signals from other detectors may take longer. The *transmission latency* ranges from ~0.8us to ~2.8us, but because the logic uses *the timestamp in the message* and ***not*** the timestamp when the message arrives, *the transmission latency can be ignored.*

For Gammasphere locally-generated triggers (e.g. multiplicity), the timestamp in the trigger acceptance message will be approximately "m"+"k"+ 0.7us after the timestamp of the actual discriminator firing. The acceptance windows should then be set to ***–(m+k)*** and ***–(m+k+1.4us)*** to insure collection of the appropriate events. If test events completely synchronous to values of the timestamp are generated using the Digitizer Tester module, the difference between the ***min*** and ***max*** windows can be set as small as 50-100ns with 100% acceptance depending upon the amplitude and rise time of the signal; this is consistent with a +/-10ns sampling error plus discriminator time-walk. In tests of this system in the array at Gammasphere using external pulse generators driving multiple channels, a difference of 120ns is sufficient to obtain the same 100% acceptance.

STOP READING HERE

STOP READING HERE

STOP READING HERE

Channel interactions

When an event is selected for readout, the *header* from the *Event Header FIFO* is combined with the matching set of ADC data samples from the *Channel FIFO*, and this package of data is sent to the board-wide *Accepted Event FIFO* that services all 10 channels. The *Accepted Event FIFO* is drained by a state machine that copies the data to the FIFO memory external to the FPGA unless the external FIFO cannot be written to because it is too full; in that case the data is lost and counted as a *dropped event.* Because there are 10 *Channel FIFOs* that fill simultaneously from all 10 channels but only one *Accepted Event FIFO*, large event sizes combined with sufficient rate may result in *dropped events* because the channels block each other from access to the *Accepted Event FIFO.*

Data Format

The channel pipelines have two basic modes of operation, ***Leading-Edge Discriminator*** and ***Constant-Fraction Discriminator.*** The latter is often abbreviated "CFD", but the similar contraction "LED" is easily confused with the identical abbreviation for *light emitting diode*. Because of this the abbreviations shall not be used. Data in both modes reads out as a packet consisting of *header* followed by *waveform*. VME data readout is always 32-bit words. A *header type* value in the *header* indicates which mode the channel was in when the data was taken. A *header length* field indicates how many 32-bit words are in the *header*, and a separate *packet length* field states the total length of *header* plus *waveform*.

Data Header – leading-edge discriminator mode

The header format for the ANL digitizer, for the leading-edge discriminator mode, is shown in Table 2. This table reflects the format for the August 2021 release. All data read from the digitizer is 32 bits wide, and the header is 14 words long.

	LEADING EDGE DISCRIMINATOR DATA FORMAT - HEADER TYPE IS 7																															
	31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23*	*22*	*21*	*20*	*19*	*18*	*17*	*16*	*15*	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*
0	FIXED 0xAAAAAAAA																															
1	Geo Addr					PACKET LENGTH											USER PACKET DATA												CHANNEL ID			
2	LEADING EDGE DISCRIMINATOR TIMESTAMP[31:0]																															
3	HEADER LENGTH						EVENT TYPE			CEM	TTS	PBYP	HEADER TYPE				LEADING EDGE DISCRIMINATOR TIMESTAMP[47:32]															
4	TIMESTAMP OF PREVIOUS LEADING EDGE DISCRIMINATOR[15:00]																PF	PO	GE	SE	CV	OF	PV	ED	TSM	VF	WF	PTE	0	0	0	0
5	TIMESTAMP OF PREVIOUS LEADING EDGE DISCRIMINATOR[47:16]																															
6	0	0	0	0	PILEUP COUNT				SAMPLED BASELINE[23:00]																							
7	DETECTOR DATA FROM TRIGGER (Target Wheel)																EXTRA DATA FROM TRIGGER (Multiplicities)															
8	POST RISE SUM[07:00]								PRE RISE SUM[23:0]																							
9	TIMESTAMP OF PEAK[15:00]																POST RISE SUM[23:08]															
10	TRIGGER TIMESTAMP DATA																CPTS	P2M	P2_SUM[13:00]													
11	PREVIOUS_HIT_POST_RISE[23:16]								MULTIPLEX_DATA_FIELD[23:00]																							
12	PREVIOUS_HIT_POST_RISE[15:08]								EARLY_PRE_RISE_ENERGY[23:00]																							
13	PREVIOUS_HIT_POST_RISE[07:00]								TIMESTAMP_OF_COARSE[09:00]										CF	PCV	PTSM	2DF	P2_SUM[23:14]									

Table 2 - format of channel header, leading-edge discriminator mode

Many items are clear from the names, but a few entities should be explained.

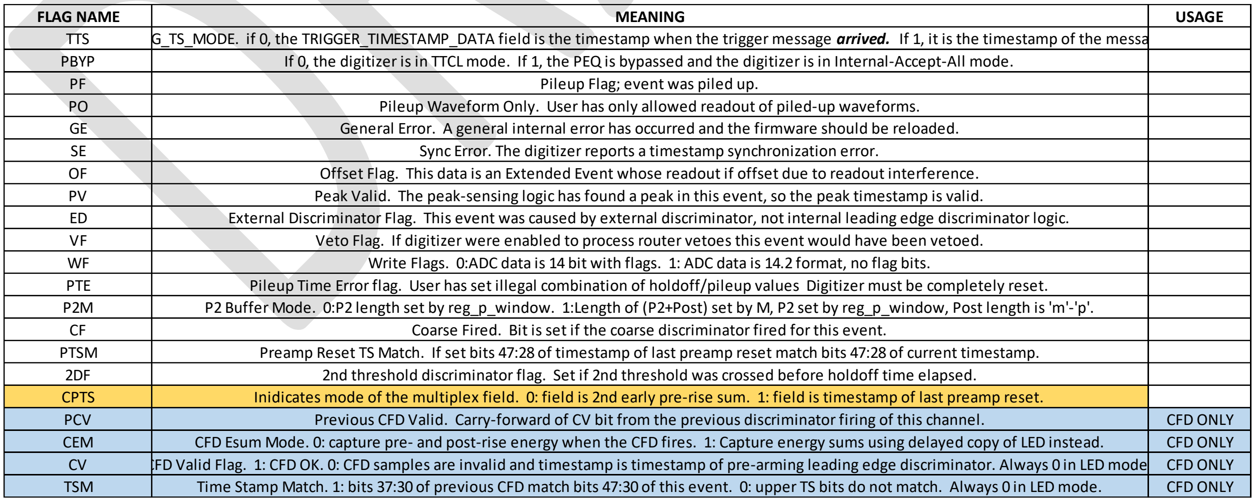
- The ***Header Type*** field is a four-bit code indicative of which format of header this is. By fiat, a ***header type*** of 0000 is the GRETINA format.
    - ANL digitizer firmware versions released prior to May 2015 have been assigned type 1 for leading-edge discriminator data, and type 2 for CFD discriminator

data. Other ANL digitizer firmware versions released later have been assigned header types 3&4, and then yet later 5&6.

  - The firmware of August 2021 uses header type 7 for LED information and header type 8 for CFD information.

- The ***Event Type*** field is a three-bit field that indicates which trigger system algorithm selected this event if the digitizer is being run in TTCL mode. If the digitizer is being run in Internal mode, the ***Event Type*** is “000”.

Many flag bits are found within the header. A summary description of the flag bits is provided in Figure 23.



FLAG NAME	MEANING	USAGE
TTS	G_TS_MODE. if 0, the TRIGGER_TIMESTAMP_DATA field is the timestamp when the trigger message ***arrived.*** If 1, it is the timestamp of the messa	
PBYP	If 0, the digitizer is in TTCL mode. If 1, the PEQ is bypassed and the digitizer is in Internal-Accept-All mode.	
PF	Pileup Flag; event was piled up.	
PO	Pileup Waveform Only. User has only allowed readout of piled-up waveforms.	
GE	General Error. A general internal error has occurred and the firmware should be reloaded.	
SE	Sync Error. The digitizer reports a timestamp synchronization error.	
OF	Offset Flag. This data is an Extended Event whose readout if offset due to readout interference.	
PV	Peak Valid. The peak-sensing logic has found a peak in this event, so the peak timestamp is valid.	
ED	External Discriminator Flag. This event was caused by external discriminator, not internal leading edge discriminator logic.	
VF	Veto Flag. If digitizer were enabled to process router vetoes this event would have been vetoed.	
WF	Write Flags. 0:ADC data is 14 bit with flags. 1: ADC data is 14.2 format, no flag bits.	
PTE	Pileup Time Error flag. User has set illegal combination of holdoff/pileup values Digitizer must be completely reset.	
P2M	P2 Buffer Mode. 0:P2 length set by reg_p_window. 1:Length of (P2+Post) set by M, P2 set by reg_p_window, Post length is 'm'-'p'.	
CF	Coarse Fired. Bit is set if the coarse discriminator fired for this event.	
PTSM	Preamp Reset TS Match. If set bits 47:28 of timestamp of last preamp reset match bits 47:28 of current timestamp.	
2DF	2nd threshold discriminator flag. Set if 2nd threshold was crossed before holdoff time elapsed.	
CPTS	Inidicates mode of the multiplex field. 0: field is 2nd early pre-rise sum. 1: field is timestamp of last preamp reset.	
PCV	Previous CFD Valid. Carry-forward of CV bit from the previous discriminator firing of this channel.	CFD ONLY
CEM	CFD Esum Mode. 0: capture pre- and post-rise energy when the CFD fires. 1: Capture energy sums using delayed copy of LED instead.	CFD ONLY
CV	FD Valid Flag. 1: CFD OK. 0: CFD samples are invalid and timestamp is timestamp of pre-arming leading edge discriminator. Always 0 in LED mode	CFD ONLY
TSM	Time Stamp Match. 1: bits 37:30 of previous CFD match bits 47:30 of this event. 0: upper TS bits do not match. Always 0 in LED mode.	CFD ONLY

Figure 23 - Summary description of all flag bits within the header.

In addition to the ***header type, event type*** and flags, many different data fields provide sufficient summary data for each event to perform timing, energy analysis, pole-zero correction and many other functions without need to read any waveform data whatsoever. These can be grouped into categories:

- **Event/device identifier** fields such as the User Packet Data, the channel ID and the Geo Addr; these are used for data sorting.
- Multiple ***sums*** captured from the accumulators that span the P2, Post-rise and Pre-rise buffers are provided for energy measurement.
- ***Timestamps*** allow for correlation of this event to other events, both with respect to this channel and all other channels in a system of multiple digitizers synchronized by the trigger. The timestamp of the peak detector is also recorded to simplify analysis of events with variable rise time.
- ***Trigger Information*** tags events with system information known only to the trigger at the time of the event such as target wheel rotational position or system-wide multiplicity sums.

Event/device identifier data

- The **Geo Addr** field provides the slot number of the VME crate the digitizer is in.
- The **Packet Length** field lists the size of the total event, inclusive of the header.
- The **User Packet Data** field is free for the user to fill in with whatever data is desired, such as a detector identifier.
- The **Channel ID** field identifies which channel the data came from.
- The **Header Length** field states how many words of the packet are "header", with the remainder of the **Packet Length** defined as "waveform".
- The **Event Type** field identifies which trigger system algorithm selected this event for readout, if the digitizer is set to use trigger accepts. If the digitizer is set to accept all hits as events ("Internal Accept All" mode), the **Event Type** is always 0.
- The **Header Type** field is the version number of the header. For August 2021 firmware this will always be 7 (LED mode data) or 8 (CFD mode data).

Sum data

The various sum fields contain the sum of a contiguous set of ADC samples captured at specific timing relative to the discriminator hit and are used for energy spectra. Please refer to Figure 5 for the names of the buffers when reading this list.

- The **Sampled Baseline** is the sum of the data across the 10.24us long T1 buffer, captured at the time the discriminator fires.
- The **Post-rise sum** is the sum of the data across the post-rise "M" buffer, captured at the time the discriminator fires.
- The **Pre-rise sum** is the sum of the data across the pre-rise "M" buffer, captured at the time the discriminator fires.
- The **P2 sum** is the sum of the data across the "P2" buffer, captured at the time the discriminator fires.
- The **Early Pre-rise sum** is the sum of the data across the pre-rise "M" buffer, captured at the time the *coarse discriminator* fires.
  - The **Multiplex Data Field** <u>*may*</u> contain a 2nd copy of the **Early Pre-rise sum**, captured at a time *between* the coarse discriminator and the normal discriminator.
  - The **Multiplex Data Field** should be interpreted as the 2nd early pre-rise sum ***only if the CPTS flag bit is 0.***

Timestamp data

The various timestamp fields contain timestamp counts at which important items occurred. Please refer to Figure 5 for the names of the fields and flag bits when reading this list.

- The **Timestamp of Discriminator** is the 48 bit timestamp count when the main discriminator fires. This is the timestamp of the event.
- The **Timestamp of Previous Discriminator** is the timestamp of the last time the main discriminator of the channel fired previous to the current event. This may be used along with Sum information for baseline extrapolation and/or pole-zero correction. Note that this timestamp is the timestamp of the most recent *discriminator* operation; in a system

using triggers to select events that previous discriminator firing may not have been selected for readout.

- The **Timestamp of Peak** is the lower 16 bits of the 48-bit timestamp when the peak detector logic found a peak. The difference between the discriminator timestamp and the peak timestamp may be used by sorting software to separate events by rise time.
    - If the **PV** (Peak Valid) flag bit is not set in the header, the **Timestamp of Peak** is invalid as the peak detector did not find a peak before the discriminator holdoff time elapsed.
- The **Timestamp of Coarse** is the lower 10 bits of the 48-bit timestamp latched when the coarse discriminator fired. This is provided to allow correct interpretation of the Early Pre-Rise sum(s).
- The **Multiplex Data Field** *may* contain bits 27:4 of the system timestamp, captured when the preamp reset logic last detected a preamp reset condition.
    - The 24-bit value spans 2^{28} clock ticks at 10ns per tick, or 2.68 seconds, with a resolution of 160ns.
        - The **PTSM** bit, if set, indicates that bits 47:28 of the preamp reset timestamp also match bits 47:28 of the discriminator timestamp. This ensures that the full 48-bit timestamp span is used.
    - The **Multiplex Data Field** should be interpreted as the timestamp of last preamp reset ***only if the CPTS flag bit is 1.***
- The **Trigger Timestamp Data** field provides the lower 16 bits of a timestamp associated with a trigger acceptance message that selected the event for readout.
    - If the **TTS** flag bit is 0, the Trigger Timestamp is the timestamp within the digitizer at the time the acceptance message *arrived.*
        - **TTS** mode 0 is typically only used for diagnostic purposes. **TTS** mode 1 is the normal mode of operation.
    - If the **TTS** flag bit is 1, the Trigger Timestamp is the lower 16 bits of the 48-bit timestamp that *was contained within the trigger accept message*, not the timestamp at which the message arrived.
        - The timestamp within the message is always the timestamp used for event acceptance.

Trigger information

The trigger information fields contain information from the trigger system that provide additional event characterization. Please refer to Figure 5 for the names of the fields and flag bits when reading this list.

- The **Detector Data From Trigger** contains a data value indicating the address source for each of the three lookup tables (*Sweep_RAM, Trig_RAM and Veto_RAM*) within the trigger, plus the current address of the *Sweep_RAM*. This is usually indicative of the rotational position of the rotating target wheel.
- The **Extra Data From Trigger** contains the 8-bit multiplicity totals from trigger virtual planes X and Y at the time the trigger message was sent. Typically in Gammasphere these would be the “clean” and “dirty” multiplicity sums, providing an indication of how

many detectors participated in this event. This value can be used to sort events by multiplicity at the time of the event.

Data Header – CFD mode

The header format for the ANL digitizer, for the CFD build, is as shown in Table 3.

<table>
<tr><td></td><td colspan="32">CONSTANT FRACTION DISCRIMINATOR DATA FORMAT - HEADER TYPE IS 8</td></tr>
<tr><td></td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td colspan="32">FIXED 0xAAAAAAAA</td></tr>
<tr><td>1</td><td colspan="5">Geo Addr</td><td colspan="11">PACKET LENGTH</td><td colspan="12">USER PACKET DATA</td><td colspan="4">CHANNEL ID</td></tr>
<tr><td>2</td><td colspan="32">CONSTANT FRACTION DISCRIMINATOR TIMESTAMP[31:0]</td></tr>
<tr><td>3</td><td colspan="6">HEADER LENGTH</td><td colspan="3">EVENT TYPE</td><td>CEM</td><td>TTS</td><td>PBYP</td><td colspan="4">HEADER TYPE</td><td colspan="16">CONSTANT FRACTION DISCRIMINATOR TIMESTAMP[47:32]</td></tr>
<tr><td>4</td><td colspan="16">TIMESTAMP OF PREVIOUS CONSTANT FRACTION DISCRIMINATOR[15:0]</td><td>PF</td><td>PO</td><td>GE</td><td>SE</td><td>CV</td><td>OF</td><td>PV</td><td>ED</td><td>TSM</td><td>VF</td><td>WF</td><td>PTE</td><td colspan="4">TRIG_DET_DATA[15:12]</td></tr>
<tr><td>5</td><td colspan="2">TDD[11:10]</td><td colspan="14">CFD SAMPLE 0</td><td colspan="2">TDD[9:8]</td><td colspan="14">TIMESTAMP OF PREVIOUS CONSTANT FRACTION DISCRIMINATOR[29:16]</td></tr>
<tr><td>6</td><td colspan="8">TRIG_DET_DATA[07:00]</td><td colspan="24">SAMPLED BASELINE[23:00]</td></tr>
<tr><td>7</td><td colspan="2">PILE_CNT[3:2]</td><td colspan="14">CFD SAMPLE 2</td><td colspan="2">PILE_CNT[3:2]</td><td colspan="14">CFD SAMPLE 1</td></tr>
<tr><td>8</td><td colspan="8">POST RISE SUM[07:00]</td><td colspan="24">PRE RISE SUM[23:00]</td></tr>
<tr><td>9</td><td colspan="16">TIMESTAMP OF PEAK[15:00]</td><td colspan="16">POST RISE SUM[23:08]</td></tr>
<tr><td>10</td><td colspan="16">TRIGGER TIMESTAMP DATA</td><td>CPTS</td><td>P2M</td><td colspan="14">P2_SUM[13:00]</td></tr>
<tr><td>11</td><td colspan="8">PREVIOUS_HIT_POST_RISE[23:16]</td><td colspan="24">MULTIPLEX_DATA_FIELD[23:00]</td></tr>
<tr><td>12</td><td colspan="8">PREVIOUS_HIT_POST_RISE[15:08]</td><td colspan="24">EARLY_PRE_RISE_ENERGY[23:00]</td></tr>
<tr><td>13</td><td colspan="8">PREVIOUS_HIT_POST_RISE[07:00]</td><td colspan="10">TIMESTAMP_OF_COARSE[09:00]</td><td>CF</td><td>PCV</td><td>PTSM</td><td>2DF</td><td colspan="10">P2_SUM[23:14]</td></tr>
</table>

Table 3 - format of channel header, CFD build

The CFD header has a few differences with respect to the leading-edge header:

1. The CFD header provides three **CFD Sample** values. These are ***signed*** objects that are the result of the subtraction of the digital CFD equation calculated by the firmware. The CFD equation is essentially a differentiation of the input pulse, so the logic looks for the crossover point from one sign to the other. CFD Sample 0 is the sample associated with the sign crossing, whereas CFD Samples 1 and 2 are the two previous samples of this subtraction taken 10ns and 20ns before CFD Sample 0.
    a. These new fields take up space, requiring sacrifice of other information. The **Extra Data From Trigger** field of the LED header is sacrificed to make room.
    b. Similarly the size of the **Timestamp of Previous Discriminator** field is cut back from a 48-bit number to a 30-bit number.
2. The **PCV, CEM, CV** and **TSM** bits, always zero in the LED header, have meaning in the CFD header.

Waveform Data Format Details

In the ANL firmware, the 14-bit data samples are reported as 14-bit *unsigned offset binary* rather than *signed binary*, such that a value of "00000000000000" indicates a sample at the *most negative voltage* the ADC can measure. A value of "11111111111111" indicates a sample at the *most positive voltage* the ADC can measure. The ANL firmware reserves the two most significant bits of each 16-bit value, as shown in Table 4. One is used as the timing mark and the other is associated with down-sampling.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>09</td><td>08</td><td>07</td><td>06</td><td>05</td><td>04</td><td>03</td><td>02</td><td>01</td><td>00</td></tr>
<tr><td>M</td><td>D</td><td colspan="14">ADC DATA SAMPLE</td><td>M</td><td>D</td><td colspan="14">ADC DATA SAMPLE</td></tr>
</table>

Table 4 - Format of waveform data

When reading waveform data, the data sample in bits 13:0 is the earlier sample, with the data sample found in bits 29:16 having been captured one 10ns clock tick later.

later insert comparison CFD picture here

------------------- review paused here ====================

Energy Summation Logic Operational Details

Upon channel initialization the accumulators are filled with a singular value from the ***Assumed Baseline*** register such that all accumulators initialize to the value of their length multiplied by ***Assumed Baseline***. On every tick of the clock after initialization the eldest value is subtracted and the newest value is added from both accumulators, so that after “X” samples – where “X” is the depth of any given buffers – the accumulator sum is valid and continuously tracks the incoming data, as shown in Figure 24. When the discriminator fires, the running sum is latched.

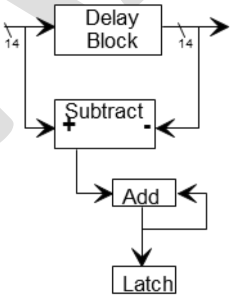


Figure 24 - Pre- and Post-rise accumulator logic

All ADC values are unsigned 14-bit objects after the ADC recode logic. In the VHDL, the two 14-bit data values are promoted to 24-bit values to support any data value that can occur from the maximum buffer length setting of 1024 samples, and both add and subtract operations are full 24-bit numbers.

Comparison of delays and sums versus the standard trapezoidal filter

An unfortunate naming convention has reversed the names of the delay buffer in the firmware relative to the naming convention used in seminal documents such as *Digital synthesis of pulse shapes in real time for high resolution radiation spectroscopy (*Jordanov and Knoll, Nuclear Instruments and Methods in Physics Research A 345 (1994) 337-345). The integrating buffers in the firmware have been inadvertently named as M1 and M2 rather than K1 and K2. The delay amount between the integrating buffers of the firmware is the ***sum*** of multiple delay buffers named “k0”, “k”, ”d”, “d2” and “d3’’.

For comparison purposes <u>*in this section alone*</u>, the naming convention used in Jordanov and Knoll will be preserved. That is, the integration time (“m” in the firmware) will be called “k” for easier comparison with the language of Jordanov & Knoll, and the sum of all the intervening delays between integration buffers (“k0” + “k” + “d” + “d2” + “d3” in the firmware) will be called “m”.

Referencing figure 5 of Jordanov and Knoll, the accumulator output is given as

$$\sum_{0}^{k}[(X(n) - X(n-k))] - [(X(n-m-k) - X(n-m-2k))]$$

The ANL digitizer firmware continuously calculates both

$$\sum_{0}^{k}[(X(n) - X(n-k))] \qquad \text{and} \qquad \sum_{0}^{k}[(X(n-m-k) - X(n-m-2k))]$$

Both sums are sampled at the time the discriminator fires. Thus, instead of continuously calculating the Jordanov/Knoll equation and attempting to determine a peak or otherwise seemingly optimal value, instead the firmware samples the two halves simultaneously at the time of the discriminator firing with the “m” term of Jordanov and Knoll’s equations defined by the sum of the firmware parameters “k0”, “’k”, “d”, “d2” and “d3”. Pole-zero compensation and calculation of the net energy is left to analysis software, using the parameters that are reported by the firmware in the event header in addition to these two partial sums. If waveform data is saved in addition to header data, the timing marks and full knowledge of all delay parameters allows the user to perform any other analysis desired in software after the event is collected.

Reporting the two separate partial sums allows for higher accuracy corrections in those cases where the pulse of interest has occurred very quickly (i.e. within the first time constant) after a previous pulse. The multiple sums taken prior to the rise of the signal along with the potential of using different sums (e.g. Post-Rise and/or P2) and the carry-forward of the Post-Rise from the previous discriminator firing provides the opportunity to calculate the different slope of the exponential decay each of the two parts are riding upon and perform both pole-zero and baseline corrections.

Discriminator filter

A symmetric digital filter is used prior to performing the discrimination function. Based upon the filter function Y(n)=[X(n)+ 2*X(n-1) + X(n-2)], the discriminator filter is set to a four-pass implementation of the base function, or

Y(n)=[X(n)+ 8*X(n-1) + 28*X(n-2) + 56*X(n-3) + 70*X(n-4) + 56*X(n-5) + 28*X(n-6) + 8*X(n-7)+ X(n-8)]

The filter is implemented using a set of adders plus hardware multipliers, resulting in the following frequency response (graph courtesy of Xilinx Coregen). With a 100MHz ADC sampling clock, frequency components above 20MHz are measurably attenuated, as shown in Figure 25. This is the functionality of the Filter block shown in Figure 5 and Figure 7.

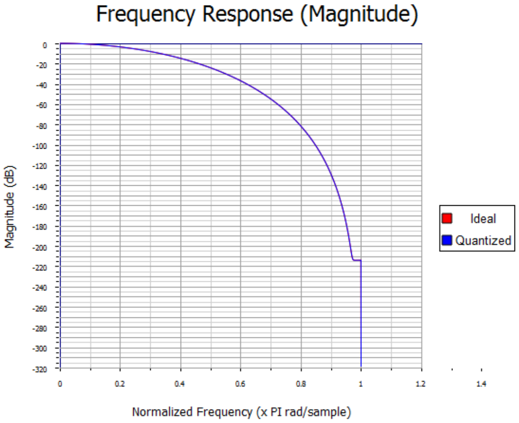
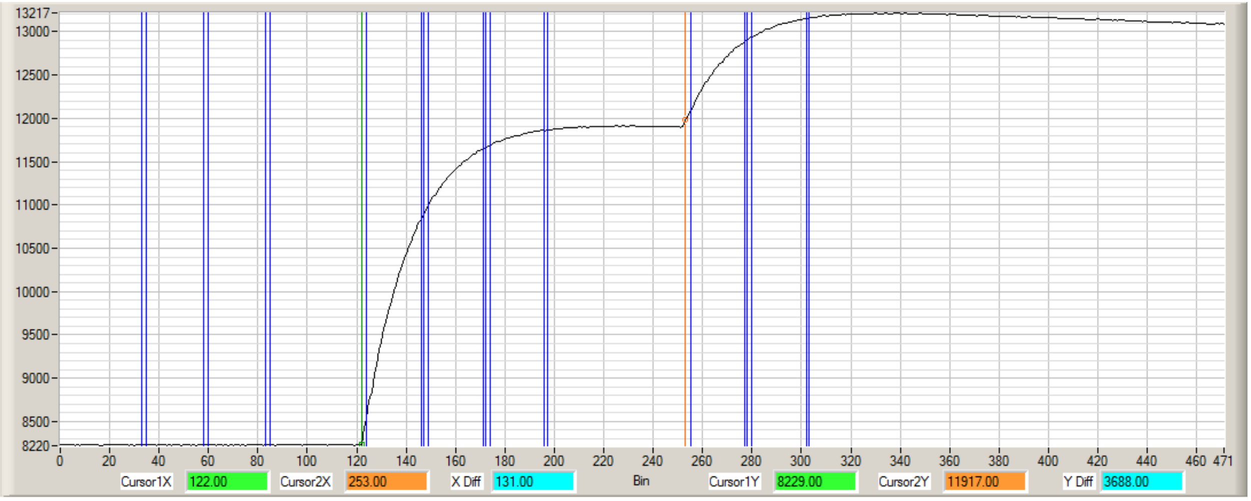


Figure 25 - Estimated frequency response plot of digital leading-edge discriminator filter.

In most applications, the correct setting for the discriminator hold-off time is a value 10% - 20% longer than the average rise time of the detector in use, with early termination on peak detection enabled. Conversely, setting the discriminator hold-off time *too small* can generate confusing results as the discriminator may fire multiple times during a single rise, as shown in Figure 26. Much as this shows how fast the discriminator can work, the holdoff value should be tuned by the user to match the typical rise time of the signals being measured.



Cursor1X 122.00 Cursor2X 253.00 X Diff 131.00 Cursor1Y 8229.00 Cursor2Y 11917.00 Y Diff 3688.00

Figure 26 - discriminator hold-off set far shorter (200ns) than rise time (~1us) - multiple hits per rise.

Preamplifier Resets in Digital Gammasphere

In the Gammasphere system a transistor-reset preamplifier is used. A monitor circuit within the preamp sets an integrated charge limit and when that limit is exceeded the preamp is reset. This results in an occasional, very large, negative-going input signal that occurs in each detector at different rates defined by the leakage current of that particular detector. The slow positive ramp caused by integration of leakage current in the detector is blocked by the AC coupling of the interface card that is juxtaposed between the preamp output and the input to the digitizer. A cutoff circuit in the interface card limits the excursion and time of the large negative signal, but some leaks through, followed by some oscillation as the interface card stabilizes. This large signal and following oscillation will generate some number of false events even if the discriminator is set to accept only positive signals.

A 'preamp reset kill' option is provided in the firmware for such situations. The default power-up state is disabled. If enabled by software, any signal creating a difference greater than 512 ADC counts across the length of the leading edge discriminator delay buffer (most normal signals are only a few counts), *and in the opposite polarity to that selected*, will disable the discriminator for a user-programmable time. The time is controlled by a register and is equal to from 1 to 255 times the register-based maximum discriminator hold-off time defined immediately above. Because of the opposing-polarity rule, this "preamp reset dead time" can only be enforced when the discriminator is set to positive-only or negative-only mode.

External Discriminator Signal

The user may set registers to enable various signal combinations external to the digitizer channel to create discriminator firings even if the actual discriminator itself has not fired. These 'external' events are indistinguishable from the leading-edge discriminator logic and can cause pileup response. There is one global control allowing the user to select the *source* of the external stimulus that is available to all channels. A second, *per-channel* control allows selection of the *response* to the external stimulus. These selections allow implementation of master-slave relationships between digitizer modules, use of the digitizer as a simple waveform recorder, timestamp-based forced baseline sampling and implementation of gated acquisition.

External Discriminator Source Selection	Channel 0-4	Channels 5-8	Channel 9
0	if channel 9 fires	if channel 9 fires	disabled
1	discbit from master in slave build	discbit from master in slave build	discbit from master in slave build
2	Front panel input pin	Front panel input pin	Front panel input pin
3	Selectable timestamp bit rollover	Selectable timestamp bit rollover	Selectable timestamp bit rollover
4	Upon write to control register	Upon write to control register	Upon write to control register
5	In master, if paired GeCenter channel is in preamp-reset-kill state	disabled	disabled
6	Upon message from trigger system, with mask	Upon message from trigger system, with mask	Upon message from trigger system, with mask

7	If any other channel fires	If any other channel fires	If any other channel fires

Table 5 - External discriminator source selection by channel.

The external discriminator *mode* is always one of

- Internal only; the external discriminator signal is ignored.
- OR; the internal discriminator is logically ORed with the selected external, an edge from the OR fires the channel
- AND; the internal discriminator is logically ANDed with the selected external, meaning that the external can act as a gate (1:yes, 0:no) upon the internal
- External only; the internal discriminator logic is ignored and only the external signal causes a hit.

Examples of common external discriminator use are

1. Setting a channel to source select 3, mode of External Only causes the channel to fire at one of a number of selectable rates irrespective of input signal. This can be used to look for noise or to measure baseline.
2. Setting a channel to source select 2, mode of OR, allows the user to connect a pulser to introduce “heartbeat” events at some rate of their choosing that intermingle with normal events.
3. Setting a channel to source select 2, mode of AND, allows the user to connect a gating signal to disable discriminator activity at certain times (e.g. during target changes or to cut rate).

Mode 1 is expected to be used in Digital Gammasphere to correlate events captured in the Side/Pattern (slave) digitizer to GeCenter events.

Constant-Fraction discriminator

A constant-fraction discriminator utilizes a delayed copy of a signal to generate the discriminator output at a time when the delayed copy is crossing a level equal to some percentage of the full amplitude of the pulse. If pulses are viewed in isolation with no concept of pileup or baseline shift, this is relatively simple. Unfortunately, a simple implementation creates many false triggers when noise is introduced. To avoid these problems, a hybrid implementation has been developed using a standard leading edge discriminator across the "k" buffer to pre-trigger a digital CFD placed across the "d" buffer as shown in Figure 27.

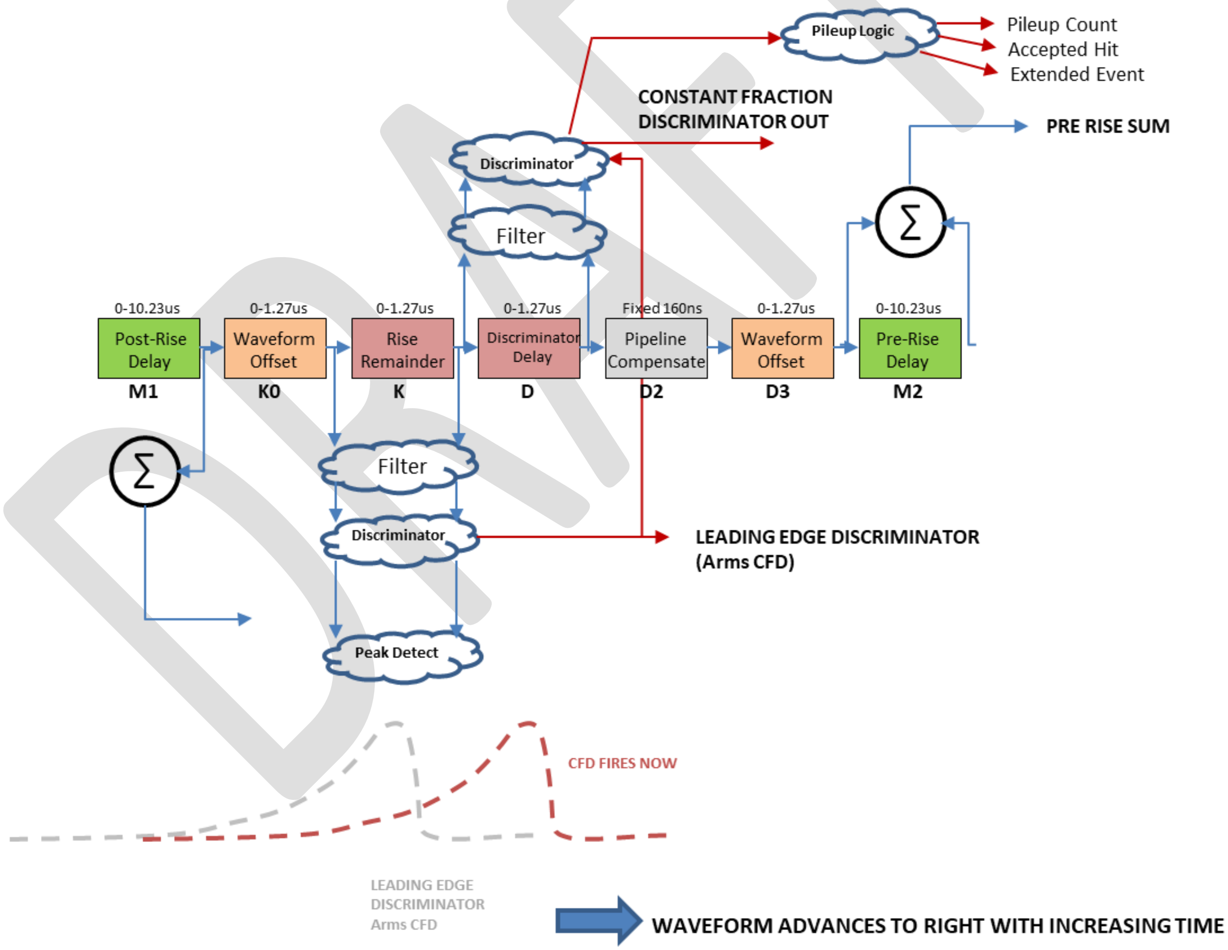


Figure 27 – Discriminator and Energy Summation portion of channel logic (CFD mode).

When the leading-edge discriminator, now placed across “k”, rather than “d”, fires, the current value of the digital CFD calculation is latched as a 'local zero' value to minimize effects from pileup or baseline drift. The CFD logic then looks for the time at which the digital CFD calculation re-crosses that 'local zero' and captures the timestamp and energy sums at that moment. A small pipeline of the digital CFD values is continuously kept so that the value of the digital CFD calculation at the moment of crossing and the two values from the previous two clock cycles may be reported in the event header.

This provides three (amplitude, time) data points allowing a linear regression to interpolate the time at which the digital CFD calculation was exactly zero. This simple interpolation – expected to be done by the DAQ system software from the values in the event header - provides a higher resolution timestamp. The increase in timing resolution is, of course, dependent upon both the rise time of the signal and the amount of noise in the system; however, tests on Digital Gammasphere signals with rise times of 750-1000ns and their equivalent indicate that resolution to approximately 2ns can be achieved, a factor of 5 better than the 10ns sampling rate.

Appropriate settings of delay parameters in CFD Mode

As the user may specify an arbitrary fraction value for the CFD, the placement of the discriminator along the rise of the signal will vary. In order to obtain reasonable energy values the delay parameters “k0”,"k", "d" and "d3" must be set to insure that, for the percentage of maximum value specified for the CFD, the entire rise of the signal is contained across the span “k0”, "k"+"d"+"d2"+"d3". Unlike the leading-edge discriminator mode, where parameter “d3” is typically set near zero, now “d2” + “d3” must be set so that it spans the amount of the rise of the waveform that occurs *before* the CFD triggers, and similarly “k0” + “k” is set to the amount of waveform *after* the triggering point, as shown in Figure 28. The usual algorithm followed is

1. Set the desired CFD fraction. 50% is common; fractions from 10% to 90% should work in most systems, but practically there is no value exceeding the range of 40% to 70%.
2. Determine a reasonable value for “d”. A typical starting point is to set “d” to a delay equal to 10 samples.
3. Set “k” to a value sufficient to use the leading-edge discriminator reliably, typically a minimum of 16 samples.
4. Take a few sample events and adjust “d3” to align the pre-rise sum appropriately.
    a. GammaWare’s buffer overlay mode or its equivalent is ideal for this purpose.
5. Adjust “k0” as required to align the post-rise sum appropriately.
6. An iterative approach to optimize “k” and “d” may be required.
7. As a general rule, setting “d” smaller will improve time-walk, ***but under no circumstances make “d” less than 10***. The usable range of CFD fraction that will work reliably decreases as “d” is decreased due to effects of rise time and noise.
8. Faster rise times will yield more consistent CFD timing with less walk.

Energy Measurement in the CFD build

The pre-rise and post-rise sums are both sampled when the CFD fires, not when the pre-arming leading-edge discriminator fires. Thus it is critically important to tune “k0” and “d3” to insure that the accumulators are properly aligned with respect to the CFD firing. Examination of sampled waveforms and the position of the timing marks embedded in the waveform data are usually necessary to obtain the correct settings.

INSERT NEW PICTURE HERE

Figure 28 - Screen capture from GammaWare showing positions of buffers in CFD mode

The careful reader will note that comparing **Error! Reference source not found.** with Figure 28 shows that the position of the discriminator firing relative to the “d” buffer is different in the two discriminator modes. In leading-edge mode the position in time of the discriminator firing is at the *right* edge of the “d” buffer whereas in constant-fraction mode the position in time of the discriminator firing is at the *left* edge of “d”. This is a function of the underlying arithmetic differences between the two modes.

Leading-edge hold-off interaction with CFD operation

Trigger Delay Buffer

The detection of pileup imposes a delay of "m"+”k” upon the discriminator bit as reported to the trigger system because the firmware can be set to accept or reject piled-up pulses. Thus, even though the discriminator fires when then rise crosses the "d" buffer, the *event* can only be marked as valid or invalid (due to pileup) "m"+”k” clocks later. At this time signals are sent to the external trigger system indicating which channels have fired. By the time the trigger system has been alerted of discriminator activity the edge of the pulse has progressed to be within the Pre-rise delay buffer. To allow for the use of an external trigger system that can make accept/reject decisions based upon a collection of pileup-modulated discriminator bits from multiple channels, a delay is required.

The Trigger Delay Buffer is a fixed-length 20.48us delay buffer that defines the amount of time after the pileup-modulated discriminator bit is asserted before an external trigger system must make a decision. If no external trigger is used (“internal” mode), the delay still exists but all events are pre-marked as selected for readout immediately upon assertion of the pileup-modulated discriminator bit.

Effects of delayed discriminator bit formation

Combined with the other delays, the pileup-modulated discriminator bit often does not arrive at the trigger system until 10us or more after the edge actually enters the digitizer. There is a delay of “m” + “k” before the edge reaches the discriminator, then the second delay of “m”+”k” after the discriminator fires before the discriminator bit is reported to the trigger. This can cause confusion when attempting to correlate events with other detectors that may not have buffered data acquisition systems and requires careful consideration. While it is easy to implement a delay to use the other detector to gate decisions within Digital Gammasphere, it is not so easy to trigger non-buffered detectors with Digital Gammasphere unless the fast, coarse discriminator is used.

This long delay may be seen by some as an architectural disadvantage, but the architecture has the advantage of being able to run at very high event rates. The only dead time imposed by the architecture of this firmware is the discriminator hold-off time; event rates in excess of 100kHz have been demonstrated in the field (400kHz in the lab) and event rates approaching 1MHz should be achievable.

Waveform Readout Generalities

When an event is accepted for readout the timestamp of the discriminator firing is available in the Event Header FIFO and the current timestamp is also known. A programmable waveform readout offset is applied to the timestamp in the Event Header FIFO, adjusted for the length of the Trigger Delay Buffer and then compared to the current timestamp. When a match is made readout commences for a programmable length. As there is no enforced correlation between the pileup time and the readout length it is possible that an event becomes available for readout before the readout of the previous event is complete. This can result in a ***dropped event***, one that could not be read out. Each channel of the digitizer has four counters that monitor channel activity; one of them counts ***dropped events*** and should be monitored at all times.

Data Collection across the Digitizer and Readout over VME

Readout of each channel fills a channel-specific FIFO. A second readout state machine continuously scans the holding FIFOs filled by the channel readout machines. When a channel FIFO indicates that a full event is available the board-wide machine transfers that event from the memory inside the FPGA to the large external FIFO memory read out by VME. No partial transfers ever occur; only whole events are transferred. Only one event is transferred from a given channel at a time; when one event from a channel is completely transferred the machine searches for the next channel with an available event even if additional events may be available in the current channel.

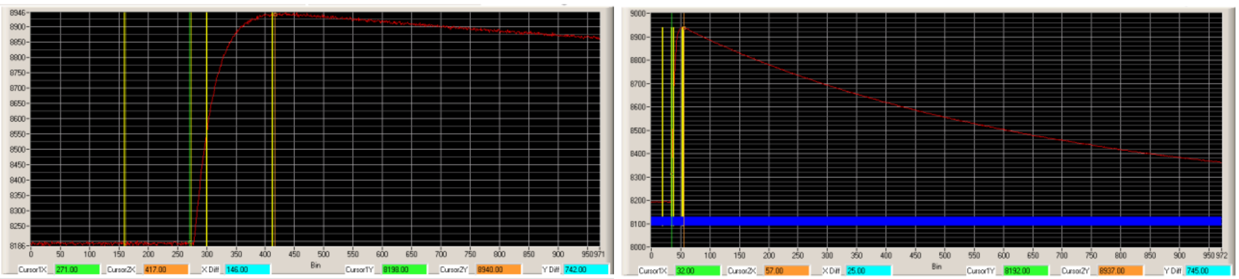
This round-robin method insures that one busy channel can't block data from the other channels, but may result in ***dropped events*** if the one busy channel fills its channel-specific FIFO and then has *another* event before the readout selector is able to circulate back to the busy channel again. In the end this is a bandwidth management issue controlled most strongly by the amount of waveform data the user has selected to read out per channel. Firmware built after May 20, 2014 partially addresses this issue by doubling the size of the Accepted Event FIFO so that it can store two full-sized (1024 sample) events before it goes full. In a VME system read out by a typical embedded processor board the total backplane bandwidth will be on the order of 10Mbytes/sec after accounting for monitoring overhead plus the inefficiencies of polling & reading out multiple digitizer boards. The total bandwidth should be divided by the total number of channels to estimate the bandwidth available per channel. Assume a crate with four digitizers within it; 10MByte/sec divided by 4 allows only 2.5MByte/sec of bandwidth per digitizer on average.

Worst-case event rates without dropped events may then be estimated, using the scenario in which all channels within a digitizer are hit simultaneously and the user has selected maximum waveform readout length in all channels. An event with full waveform readout is approximately 2100 bytes, including the header. At 2.5MByte/sec, an event of that size takes 840usec to read out from each channel. With every channel hit, then, the round-robin can’t get back to a channel once read for 8.4 milliseconds. With single buffering the Accepted Event FIFO can handle one event in 8.4msec (119Hz) but with double-buffering twice that rate can be supported in the worst-case scenario. Such worst-case situations are unlikely in any practical experiment. It is more likely that in a practical experimental situation only a few percent of channels will be simultaneously hit. Assuming a channel occupancy 10%, but with every digitizer module in the crate still being hit every time, with double-buffering the full-length readout will support event rates of approximately 2.4kHz. With a more practical waveform readout length of 100 samples per event rather than the maximum of 1024, a much more useful event rate of 24kHz is achieved.

Waveform Down-Sampling

Often experimenters desire to have waveform readouts spanning a greater amount of time, but the full 100MSamp/sec sampling rate is excessive. To accommodate this, "down-sampling" logic is provided in the channel-level readout machine. A programmable data reduction factor from 0 to 7 is available. The data reduction factor selects a power-of-two data reduction (0=all, 1=reduce by factor of two, 2=reduce by factor of four, etc.). Data reduction is accomplished by averaging together the selected number of full speed data samples and then writing the average-of-n every nth clock. The result is that the length in time covered by the waveform buffer doubles for each successive data reduction factor. At the maximum setting, a full length (1024 sample) readout spans 10.24us * 128, or 1.31 milliseconds. An example of the effect of down-sampling is shown in Figure 29.

The blue bar in the down-sampled readout is added by the GammaWare program to indicate to the user that the data is down-sampled. It is important to note that when down-sampling is in effect the down-sample factor also applies to the waveform readout offset; the number of down-sampled waveform data points that will be provided is equal to the time offset factor demanded (in units of full-speed data points) divided by the down-sample factor.



Cursor1X 271.00 Cursor2X 417.00 X Diff 146.00 Cursor1Y 8198.00 Cursor2Y 8940.00 Y Diff 742.00

Cursor1X 32.00 Cursor2X 57.00 X Diff 25.00 Cursor1Y 8192.00 Cursor2Y 8937.00 Y Diff 745.00

Figure 29 - Event read out full speed (left) and down-sampled by 8x (right)

On-Off down-sampling

The user may, in addition to setting a down-sample factor, set the digitizer to implement ***on-off down-sampling.*** In this mode the data starts out down-sampled at the factor prescribed but the data then *switches* from down-sampled to full speed data in the region of a signal. This is triggered by the leading-edge discriminator timing mark (Timing marks are discussed later in this document, in the section **Waveform Data Format Details** on page 42). In this mode a 2nd hold-off timer is used to determine how many samples are read out at full speed before the readout switches back to the down-sampled format. Transitions between full speed and down-sampled occur at the point in the full-speed waveform at which the next down-sampled data point would occur so that the actual number of samples provided at full speed line up with the down-sampled data, neither skipping a full speed sample nor displaying a full-speed sample that is also contained within the average value of a down-sampled waveform point.

Pileup and Event Readout

The ANL digitizer handles pulse pileup at three levels. The first level of logic is at the channel level where events are accepted or rejected. If a channel is set to reject piled-up events neither the readout logic nor the trigger system knows that these pileup events occurred. However, the coarse discriminator will always report found edges irrespective of pileup settings. A second level of pileup control occurs at the readout machine level. This logic necessitates a bit of terminology. We define events leaving the channel logic and entering the readout logic as being either ***accepted*** or ***extended***.

- When pileup rejection is on, all events proceeding to the readout logic are, by definition, not piled up. Such events are all flagged as "***accepted"*** events.
- When pileup rejection is off, only the *first* event in a string of piled-up events, or any isolated and non-piled event, is ***accepted;*** all subsequent events in a piled-up string are marked as ***extended*** events.
- By definition ***extended*** events can only occur following an ***accepted*** event. The ACCEPTED_HIT signal is the source of the delayed signal EVENT_EXPIRED that removes events from being accepted for readout by an external trigger system. Thus the PEQ only contains the timestamps of ***accepted*** hits.

The third level of pileup control occurs based upon a pileup flag stored in the header of all ***accepted*** events. The readout machine may be set to only read out those ***accepted*** events that have the pileup bit set, resulting in *only piled up events showing up in the readout and all other events suppressed.*

Readout Interference

When the readout length of one event *overlaps* the data from the next event there is risk of *readout interference*. This situation usually arises when the selected readout length is longer than the accumulation buffer depth. Figure 30 shows an example waveform highlighting the effects of pileup and selected waveform readout length. In Figure 30, four pulses cause four discriminator firings and, thus, four events. The timing between the firings, when compared to the depth of the summation buffer "m", causes events #2 and #3 to be piled upon one another. However, the waveform readout depth selected by the user causes the ***readout*** of event #1 to interfere with that of event #2. The lower portion of Figure 30 shows the events on a timeline showing when each event would ideally read out. This timeline will be repeated in the following sections to explain how the various modes affect the readout.

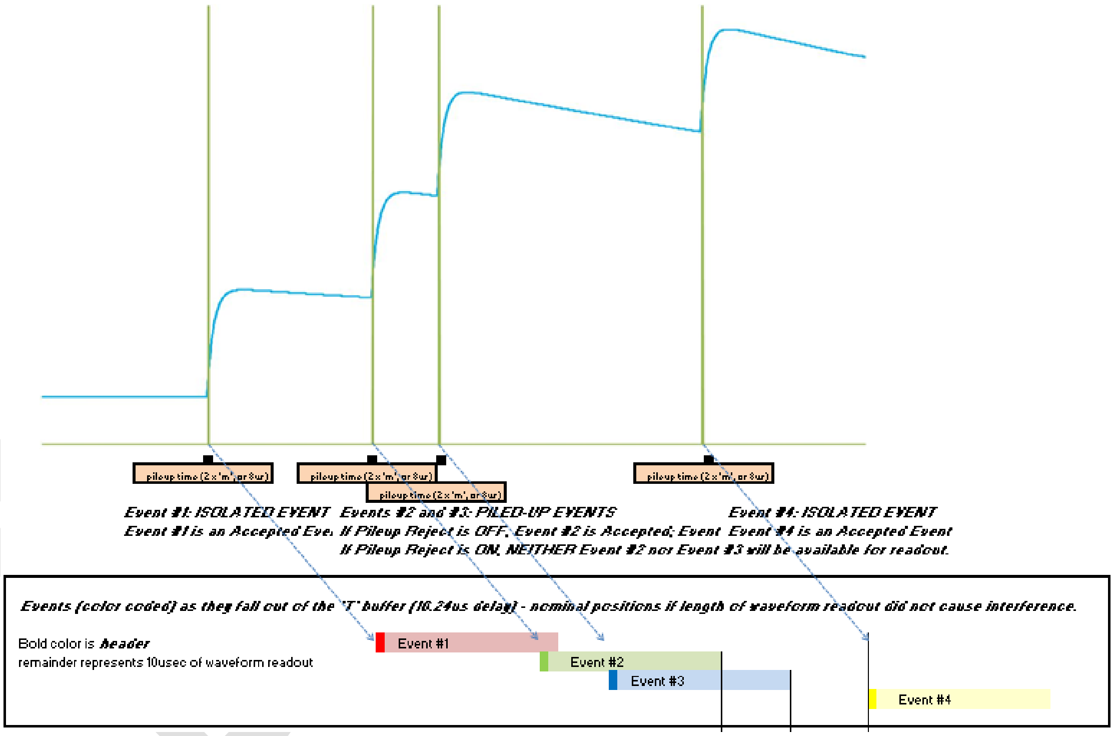

pileup time (2 x 'm', or 8ur)
pileup time (2 x 'm', or 8ur)
pileup time (2 x 'm', or 8ur)
pileup time (2 x 'm', or 8ur)
Event #1: ISOLATED EVENT
Event #1 is an Accepted Eve
Events #2 and #3: PILED-UP EVENTS
If Pileup Reject is OFF, Event #2 is Accepted; Event
If Pileup Reject is ON, NEITHER Event #2 nor Event #3 will be available for readout.
Event #4: ISOLATED EVENT
Event #4 is an Accepted Event
Events (color coded) as they fall out of the 'T' buffer (10.24us delay) - nominal positions if length of waveform readout did not cause interference.
Bold color is header
remainder represents 10usec of waveform readout
Event #1
Event #2
Event #3
Event #4

Figure 30 - Example waveform for describing pileup readout modes

Readout Interference when Down-sampling is in use

It is critically important to understand that the use of waveform down-sampling has a direct impact upon readout interference. With waveform down-sampling enabled, once readout of an ***accepted*** event occurs, the longer physical time spanned by the decimated readout will inevitably result in a higher number of following events being contained in the waveform readout from the first event. In the default readout mode of ***No Offset,*** use of significant down-sampling factors will typically result in many events contained in waveforms that have no headers. Generally, if the DAQ system can handle variable event sizes, use of the ***Offset with Truncation*** readout mode is considered best when down-sampling is activated.

Pileup Extension and Readout Modes

When Pileup Extension is ***disabled*** the readout logic is constrained to process only ***accepted*** events. This means that isolated events (no pileup) and the ***first*** event of a pileup train may potentially be read out, but that the 2nd and later events within a pileup train (the ***extended*** events) will all be discarded. When Pileup Extension is ***enabled*** the readout logic is may process all events, ***accepted*** or ***extended***. The default state is ***disabled.*** Separately, the user may select one of four possible event readout modes, named ***No Offset, Offset, Offset with Truncation*** and ***Header Only.*** These four modes operate in slightly different ways depending upon whether Pileup Extension is enabled or disabled so the combination of the Pileup Extension and Readout Mode controls effectively generate ***eight*** different readouts. The two sections below describe how each of the four readout modes will operate in each of the two Pileup Extension states.

Readout Modes when Pileup Extension is Disabled

In the event example shown as Figure 30, Event #2's start occurs before the entire readout of Event #1 is complete, setting up a situation where *readout interference* occurs. Because Event #3 is an ***extended*** event – being the 2nd pulse in the pileup group – it will never be read out in any of these cases because Pileup Extension is *disabled*. If the pileup train were longer (additional events between #3 and #4), these would also be ***extended*** and thus excluded from the readout. Each of the four readout modes will react differently to the *readout interference* condition, as summarized in Table 6.

Extension Mode	**Name**	**Operational Description**
00	No Offset	If the readout length of any event overlaps with the data of a later event, the later event is lost.
01	Offset	If the length of the readout of an event causes that readout to extend past the start of the next, a second header will be issued and a 2nd full buffer of samples will be read out for the 2nd event, even though the leading edge of the 2nd event may be found in the 1st event's waveform data or may span both buffers. Offset continues to accumulate until the amount of offset exceeds the length of the readout machine's accepted event FIFO. Should this occur the next event is dropped.
10	Offset with Truncation	If the length of the readout of one event causes that readout to extend past the start of the next, readout of the later event starts immediately after the first one. The total length of waveform data in the 2nd event is shortened so that only *additional* waveform data past the overlap point is read out. This results in a total readout length of the following event that is variable.
11	Header Only	If the length of the readout of one event causes that readout to extend past the start of the next, only the header of the overlapping event is read out and no waveform data in the 2nd event is provided.

Table 6 - Modes of readout with Pileup Extension DISABLED

Figure 31 provides a visual demonstration of these effects. The ***No Offset*** mode prevents readout of Event #2 because #2 *interferes* with the readout of Event #1 already in progress. The other three modes allow some or all of Event #2 to be read out by offsetting the beginning of the readout until the interference is cleared. The ***Header Only*** mode provides an alternate solution to *readout interference* by collapsing interfering events to only headers.

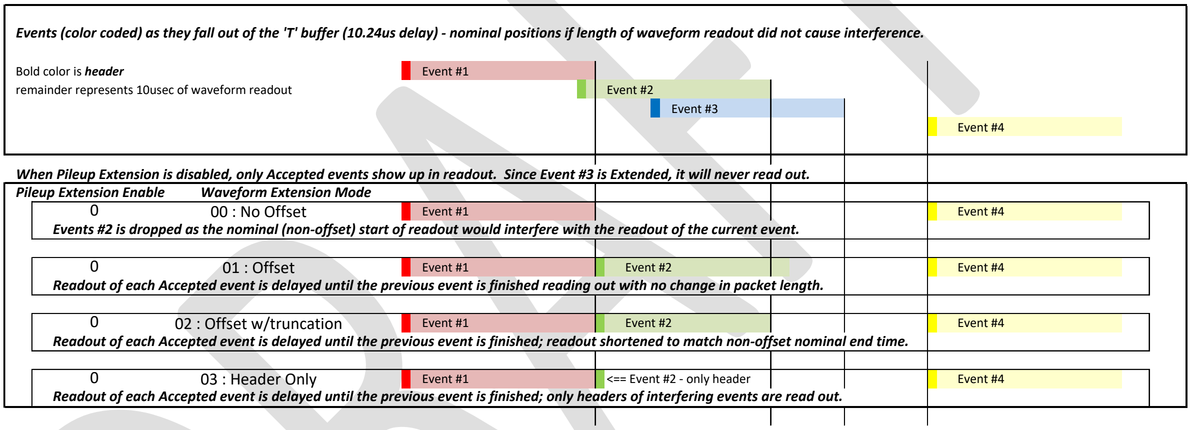

Events (color coded) as they fall out of the 'T' buffer (10.24us delay) - nominal positions if length of waveform readout did not cause interference.
Bold color is header
remainder represents 10usec of waveform readout
Event #1
Event #2
Event #3
Event #4
When Pileup Extension is disabled, only Accepted events show up in readout. Since Event #3 is Extended, it will never read out.
Pileup Extension Enable
Waveform Extension Mode
0
00 : No Offset
Event #1
Event #4
Events #2 is dropped as the nominal (non-offset) start of readout would interfere with the readout of the current event.
0
01 : Offset
Event #1
Event #2
Event #4
Readout of each Accepted event is delayed until the previous event is finished reading out with no change in packet length.
0
02 : Offset w/truncation
Event #1
Event #2
Event #4
Readout of each Accepted event is delayed until the previous event is finished; readout shortened to match non-offset nominal end time.
0
03 : Header Only
Event #1
<== Event #2 - only header
Event #4
Readout of each Accepted event is delayed until the previous event is finished; only headers of interfering events are read out.

Figure 31 - Visual representation of the readout modes with Pileup Extension DISABLED

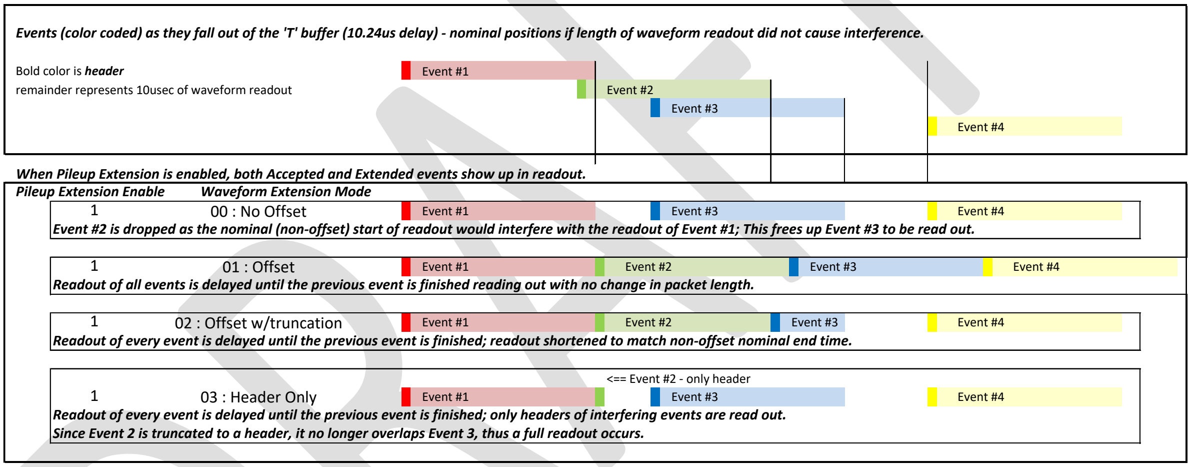
Readout operation when Pileup Extension is Enabled

Again using the event example shown as Figure 30, Event #2's start occurs before the entire readout of Event #1 is complete, setting up a situation where *readout interference* occurs. However, because Pileup Extension is now ***enabled***, Event #3 (an ***extended*** event) can now be read out. If the pileup train were longer (additional events between #3 and #4), these would also be ***extended*** and thus also available for readout. Each of the four readout modes will react differently to the *readout interference* condition, as summarized in Table 7.

Extension Mode	Name	Operational Description
00	No Offset	In this mode, both ***accepted*** and ***extended*** events may be read out but *only* if the entire event can be read out to the programmed event length before the next event has to be read out. In this mode either ***accepted*** or ***extended*** events may be discarded because the full readout wouldn't fit; thus use of this mode is discouraged when Pileup Extension is enabled.
01	Offset	If the length of the readout of an event causes that readout to extend past the start of the next, a second header will be issued and a 2nd full buffer of samples will be read out for the 2nd event, even though the leading edge of the 2nd event may be found in the 1st event's waveform data or may span both buffers. Offset continues to accumulate until the amount of offset exceeds the length of the readout machine's accepted event FIFO. Should this occur the next event is dropped.
10	Offset with Truncation	If the length of the readout of one event causes that readout to extend past the start of the next, readout of the later event starts immediately after the first one. But, the total length of waveform data in the next event is shortened so that only additional waveform data past the overlap point is read out. This results in a total readout length that is variable; however, this is the most efficient format for reading out piled-up waveforms.
11	Header Only	If the length of the readout of one event causes that readout to extend past the start of the next, only the header of the overlapping event is read out. If any event can be read out in full, it will be.

Table 7 - Readout modes with Pileup Extension ENABLED

As in the previous section, Figure 32 provides a visual demonstration of these effects. The ***No Offset*** mode prevents readout of Event #2 because #2 *interferes* with the readout of Event #1 already in progress; however, Event #3 can be read out is it does not interfere. This potential of skipping the ***accepted*** event while allowing through some or all of the ***extended*** events associated with the missing ***accepted*** event is why we discourage the use of the ***No Offset*** mode when Pileup Extension is enabled. Please note that the ***Header Only*** mode only collapses *interfering* events into headers, and that such collapse may then free other events to be read in full.



Events (color coded) as they fall out of the 'T' buffer (10.24us delay) - nominal positions if length of waveform readout did not cause interference.

Bold color is ***header***

remainder represents 10usec of waveform readout

When Pileup Extension is enabled, both Accepted and Extended events show up in readout.

Pileup Extension Enable	***Waveform Extension Mode***
1	00 : No Offset
1	01 : Offset
1	02 : Offset w/truncation
1	03 : Header Only

Event #2 is dropped as the nominal (non-offset) start of readout would interfere with the readout of Event #1; This frees up Event #3 to be read out.

Readout of all events is delayed until the previous event is finished reading out with no change in packet length.

Readout of every event is delayed until the previous event is finished; readout shortened to match non-offset nominal end time.

Readout of every event is delayed until the previous event is finished; only headers of interfering events are read out.

Since Event 2 is truncated to a header, it no longer overlaps Event 3, thus a full readout occurs.

Figure 32 - Visual representation of the readout modes with Pileup Extension ENABLED

Interface to the external trigger

The ANL firmware implements a SERDES link between digitizer and trigger system in a manner similar to that used in GRETINA, but data issued by the digitizer has a significantly different format. On the receive side, the digitizer receives the control stream from the trigger and synchronizes itself to the clock from the trigger and to the timestamp as broadcast by the trigger. The firmware responds to the command frames defined in the Trigger Timing and Control Link specification as originally written for the GRETINA experiment, plus many additions made for multiple detectors and cross-triggering unique to the Digital Gammasphere version of the trigger firmware. Specific features of the digitizer firmware related to the triggering system are discussed in this section.

Timestamp Synchronization

The ANL digitizer uses a reception state machine similar to that of the Router trigger implementing the 'stringent lock' feature. All fixed values within the command stream may be monitored for more stringent checking of data validity. In addition, enhanced timestamp synchronization logic is implemented. The digitizer resets its timestamp upon receipt of Imperative Sync commands but also uses the non-imperative Sync commands to compare against the local timestamp counter. Mismatches between the timestamp value in the Sync command are assumed to be data transmission errors rather than counter errors. For a singular mismatch, the local timestamp is maintained and the trigger's timestamp is ignored. If three mismatches in a row occur, it is assumed that there is a serious communications error and a status bit is set in a register so that slow controls may alert the user to the problem.

Event Veto

Provision is made for an Event Veto input from the trigger for each channel. The Event Veto is enabled by register bits within the digitizer. If this function is enabled the *most recently entered value* in the Pending Event Queue is erased should the trigger assert the Event Veto bit for a given channel. This feature requires relatively tight timing; any such veto must be issued before the next pulse places an entry into the Pending Event Queue. Fortunately, only ***Accepted*** events add data to the Pending Event Queue, so a delay of a couple microseconds should always be available with normal pileup settings.

The Event Veto is intended for use in concert with the external triggering system. In the specific case of Digital Gammasphere, two channels of each digitizer are associated with the Ge Center and BGO Sum signals of a given detector. Digital Gammasphere detectors are intended to be run in "clean" (Ge hit without corresponding BGO), "dirty" (Ge and BGO) or "module" modes (Ge or BGO), but the hits are not coincident in time. By use of the Event Veto function, the Router modules of the external trigger may monitor the channel pairs, implement the time window and issue commands to Veto the events that are "dirty" or "module" if only "clean" events are desired. There is no provision to go back and Veto an older event deeper in the PEQ. As the PEQ only contains *Accepted* events, a Veto by definition suppresses any and all *Extended* events associated with the vetoed event and thus suppresses an entire train of piled-up pulses.

In the Digital Gammasphere version of trigger firmware, the Router trigger modules independently calculate the Event Veto bits for the digitizers each serves to minimize latency. The Veto bits themselves are inserted by the Routers into the 5th word of each Trigger Timing and Control frame on the fly, without any intervention by the Master Trigger. As channel pairs are vetoed by the Router boards, internal “Clean multiplicity”, “Dirty multiplicity” and “Module multiplicity” sums are continuously updated. Any two of these per-Router partial sums may be selected for transmission to the Master Trigger. The Master Trigger forms the system-wide multiplicity sums and may then issue trigger acceptance commands based upon these sums.

The Veto process is optionally enabled by the user, so that raw multiplicities may always be used in place of the Clean, Dirty or Module sums. In non-Veto modes mapping registers within the Router trigger boards allow each discriminator bit to be optionally mapped into the “X-plane” or “Y-plane”, allowing the same firmware to service the dual-sided silicon strip detector of DFMA.

Trigger Decision Latency

The trigger will not respond to all events with the same time delay. This is caused by two factors:

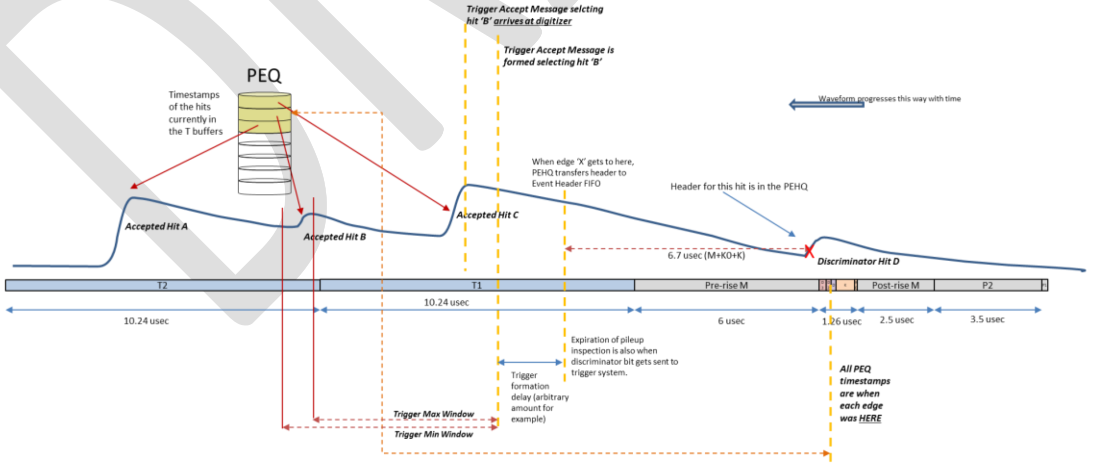
1. Each different trigger algorithm that may be coded will have a different ***trigger formation time*** that we will name T_{tf}.
2. The 2us system cycle of the Trigger Timing and Control link means that there is an additional delay of from zero to 2us after T_{tf} before the trigger accept message can be transmitted; then of course there is the risk that in a high rate experiment triggers could back up in the FIFOs of the trigger system causing occasional delays of 0-4us before transmission.

Details of the trigger window calculation

When a trigger acceptance message is received, all event timestamps in the PEQ are compared against the timestamp value contained within the trigger message. A bit of math is required. The timestamp reported by the trigger is the time at which the trigger module determined that a trigger algorithm was satisfied. If an event occurs at time T_0, the trigger will respond at a variable time T_{tf} later. When processing the events in the pending event queue the timestamps in that queue are the time of the leading edge (T_0).

Upon receipt of the trigger message, the timestamp of the *trigger message* is subtracted from the timestamp of the *firing within the pending event queue* for every firing within the queue. Under normal conditions where the trigger is formed from multiplicity, this will always yield a negative number. However, if the trigger is caused by a different detector that may be faster than the one connected to the digitizers, a positive answer may result. The answer is compared against the upper and lower time window registers (< upper, > lower) and if both comparisons are valid, the event is accepted for readout.

The method of subtraction used means that the window is defined ***with respect to the time the Master trigger detected the appropriate condition*** and not the time of the discriminator firing. That is, if there were an oscilloscope connected to the discriminator signal and also to the trigger signal[1], and the oscilloscope were triggering on the signal from the trigger system, discriminator firings would be seen in a range of times prior to the trigger signal. Thus, the correct settings for the two window registers would also be *negative*, with the '***min*** window' register set to the relative time of the earliest (farthest back from the trigger) discriminator firings and the '***max*** window' set to the relative time of the latest (closest to the trigger) discriminator firings. This apparent reversal of "min" and "max" is because the logic always requires that the "max" window edge be ***later*** in absolute time than the "min" position. See Figure 33.


Trigger Accept Message selcting hit 'B' arrives at digitizer
Trigger Accept Message is formed selecting hit 'B'
PEQ
Timestamps of the hits currently in the T buffers
Waveform progresses this way with time
When edge 'X' gets to here, PEHQ transfers header to Event Header FIFO
Header for this hit is in the PEHQ
Accepted Hit A
Accepted Hit B
Accepted Hit C
6.7 usec (M+K0+K)
Discriminator Hit D
T2
T1
Pre-rise M
Post-rise M
P2
10.24 usec
10.24 usec
6 usec
1.26 usec
2.5 usec
3.5 usec
Expiration of pileup inspection is also when discriminator bit gets sent to trigger system.
Trigger formation delay (arbitrary amount for example)
All PEQ timestamps are when each edge was HERE
Trigger Max Window
Trigger Min Window

When the trigger accept message arrives, the timestamp value *contained within the message* is subtracted from the timestamp values in the PEQ. Any subtraction result that falls between the MAX and MIN values chosen by the user is selected for readout.

Figure 33 - Trigger windows within the digitizer

The discriminator bit from any hit is sent to the trigger at the end of the pileup time (M+K0+K). For the cable plant of Gammasphere, using only local multiplicity triggers, approximately 0.75usec is required for the message to propagate through the SERDES link to the Router trigger board and then again for the Router data to propagate to the Master Trigger. After the message is received, a variable *trigger formation delay* occurs dependent upon the trigger conditions. A Gammasphere multiplicity trigger has a short *formation delay* of ~0.2usec, but triggers that require coincidence with signals from other detectors may take longer. For Gammasphere locally-generated triggers (e.g. multiplicity), the timestamp in the trigger acceptance message will be approximately "m"+"k"+ 0.95us after the timestamp of the actual discriminator firing. The acceptance windows should then be set to ***–(m+k)*** and ***–(m+k+1.8us)*** to insure collection of the appropriate events.

[1] Usually available by setting the NIM output of the Master Trigger to the ANY_TRIG setting.

Other Commands from the Trigger

Trigger decision commands are processed by the digitizer if the “TTCL” mode is enabled; otherwise they have no effect. The Demand Slow Data command (frame #13) of the trigger’s command stream – used in the LBNL version of the digitizer – has no effect in the ANL version as there is no ‘slow’ data to send; in Gammasphere and DSSD all data is ‘fast’. The ANL digitizer is, however, sensitive to commands distributed in the 16th frame. This frame is interpreted as a Synchronous System Capture command that causes diagnostic counters within the digitizer to first reset and then collect counts of various activities within each channel for a period of time defined within the data values of the command.

Information Sent to the Trigger System by the Digitizer

By the same token, the lack of the ‘fast’ vs. ‘slow’ data concept as used in the GRETINA version written by LBNL allows for a simpler data format to the trigger system. In the ANL firmware each word every 20ns is a unique message. The data sent every 20ns is simply a snapshot of the state of every channel’s discriminator bit plus the fast, or coarse, discriminator bits from channels 5-9. The selection of channels 5-9 for the coarse bits is driven by the specific cable plant of Digital Gammasphere, where channels 5-9 are connected to the germanium center contacts and channels 0-4 are connected to the BGO sum signals. By having the fast, coarse, Ge discriminator bits sent to the trigger system prompt multiplicity-based pre-triggers may be formed system-wide. As all channels are reported the Ge/BGO nomenclature is only a convenience for Digital Gammasphere and any other detector system still has all the information from all channels.

The amount of time each discriminator bit is asserted is programmable to insure multiplicity sums are properly calculated. Within the Digital Gammasphere system, where multiple channels of a digitizer are tied to the same detector module, cross-channel marking or triggering is performed entirely within the trigger.

The choice to send only discriminator bits and not energy values is driven by the specific architecture of the Digital Gammasphere and Digital FMA systems where all triggers are based upon multiplicities. No energy information is provided to the trigger by the digitizers in these systems as the system designers have concluded that energy-based triggers are not practical for this detector. There is no provision in the all-fast data format to add any information other than discriminator data to the data sent to the trigger.

Data Flow Control (Throttle)

The digitizer hardware supports two non-serialized LVDS control signals that run from the digitizer to the router trigger board. One of these is defined in both GRETINA and DGS firmware as the Throttle bit. This bit is asserted by the firmware whenever the board-wide FIFO reaches the half full point. The trigger routers collect the Throttle bits from each digitizer and the Master Trigger may suspend the issuance of trigger accept messages while any digitizer is asserting the Throttle bit.

This hardware- and firmware-based rate throttling has been tested and is in common usage in Digital Gammasphere. As of this writing, however, GRETINA digitizers do not use the Throttle bit and instead implement trigger rate control by a software mechanism.

Special trigger modes

The digitizer may be set to two special trigger modes that have rare but occasional utility. By use of the trigger configuration register the digitizer may be set to *External Trigger* or *Diagnostic Trigger* mode.

- In *External Trigger* mode the digitizer operates like TTCL mode, but the trigger itself is not a message from the trigger system but instead the application of a signal to a test point of the digitizer module. The timestamp of the trigger is the timestamp value within the digitizer at the time the trigger signal is applied to the test point.
- In *Diagnostic Trigger* mode the digitizer operates like TTCL mode, but the trigger is automatically and internally generated every time the lower 16 bits of the digitizer timestamp roll over from 0xFFFF to 0x0000.

Master and Slave Digitizers

A compile-time build option named SLAVE_MODE allows use of the front bus ribbon cable to connect one digitizer to another in a subservient relationship. The front bus ribbon cable contains a mix of signals with a certain amount of data direction control. The mix of signal types and directions is the same as in GRETINA digitizers, but the definition and usage has been changed:

- A differential Clock signal whose direction is controlled, not by firmware, but by hardware jumpers. The Master and Slave boards must have their jumpers set correctly to work as desired. From the firmware standpoint, FBUS_CLK is an input that either is used (Slave) or ignored (Master). The clock sent from the Master to the Slave is the switched 50MHz clock, that when a trigger system is used is the clock received from the trigger via the SERDES link.
- Eighteen bits that are used to carry data from the Master to the Slave(s). The lower 17 bits are a direct and immediate copy of the SERDES data that comes from the trigger system. In the Slave digitizer these bits are fed to the SERDES receive state machine in lieu of the data from the Slave's SERDES chip. The data is DC-balanced but with the "clock guard" bit removed.
    - Bit 17 is a reset bit sent from the Master to the Slave, the same as the Master's main reset signal.
    - Bit 16 of the Master-to-Slave data is, instead of the clock guard, a copy of the discriminator bit from channel 0 of the Master digitizer. This bit is one of the four "external discriminator" options. Certain combinations of these modes allow any or all of the channels of a Slave digitizer to be directly driven by the bit from the Master.

        - Due to delay in crossing the cable, the P1 and/or P2 delay buffers of all channels in the Slave must be adjusted to values a few clocks larger than that of the Master digitizer to synchronize operation.
- Three bits are transmitted from the Slave to the Master.
    - A status flag from the Slave's external FIFO is sent back to the Master so that the Master may assert its Throttle Request bit to the trigger if either the Master or the Slave is in danger of FIFO overflow.
    - A status bit is sent from the Slave to the Master to indicate that the Slave's SERDES receiver state machine is locked onto the trigger command stream.
    - The third bit is available as a spare.
- A "Wire-Or" bit is also available on the cable that may be used for the Throttle Request feature in cases where multiple Slaves are connected to the Master. This, of course, requires that jumpers and terminations be appropriately set to insure clean data transfer.

These connections are summarized in Figure 34. An important item to note is that the Slave digitizer receives and may respond to any commands sent by the trigger system, but the Slave digitizer has no way to communicate any discriminator information ***back*** to the trigger. One signal is available to provide rate control via the trigger throttle request, but that is the total extent of the Slave's communication to the trigger system. As such, the trigger is blind to the operation of the Slave and has no ability to consider the Slave's state in any trigger decision. This differs from the GRETINA model in which there is a slow collection mechanism to collect Slave digitizer discriminator bits when the central contact fires. However, that mechanism does not preserve timing of the Slave bits and cannot detect pileup.

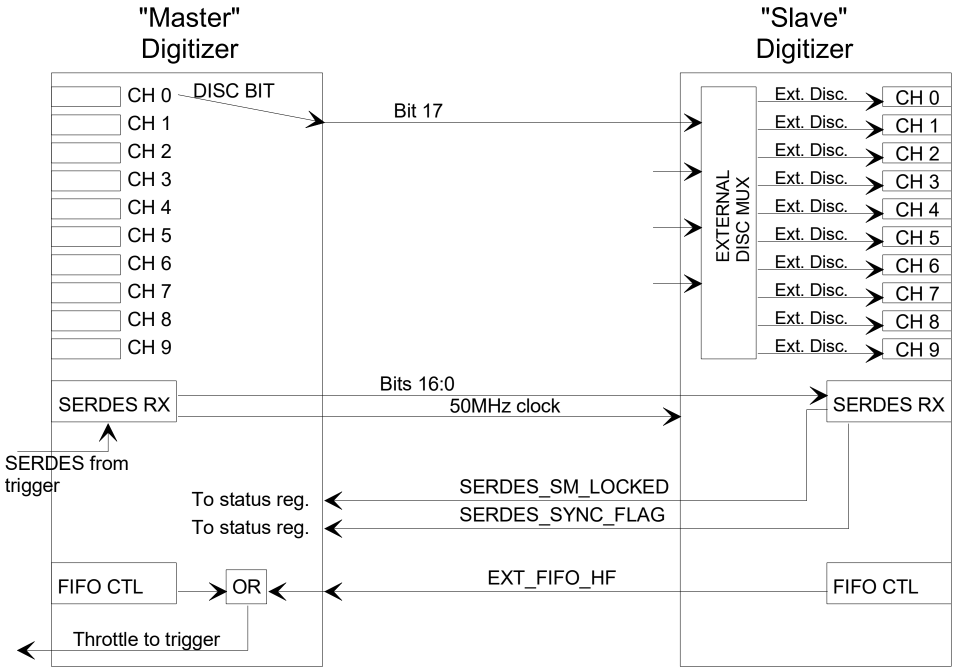


Figure 34 - Connections between Master and Slave digitizers.

Detailed Description of External Discriminator Modes

Figure 34 shows that the cable is but one potential source for the External Discriminator signal in the channels of the digitizer. Note that the External Discriminator operation is available in ***both*** Master and Slave builds of the ANL digitizer. The *source* control for the external discriminator allows the user to select from one of four options to define the external discriminator signal:

1. The discriminator bit from Channel 0 of the board.
2. If the digitizer has the Slave version of firmware, a second option is bit 17 of the data from the front bus cable. In Master digitizer builds, this option is disabled.
3. Auxiliary I/O bit 10 on the front panel.
4. One of six selectable timestamp-based clocks.

At the *channel* level, each channel has four options defining that channel's response to the external discriminator signal:

1. The channel fires based upon its own discriminator logic (default); the external discriminator signal is ignored.
2. The channel fires when its own discriminator fires, OR when the external discriminator signal has an edge.
3. The channel fires when its own discriminator fires, but only if the external discriminator signal is also asserted (gated internal mode).
4. The channel ignores its own discriminator and fires only when the external discriminator signal has an edge.

Table 8 summarizes the combinations available using the external discriminator logic, suggesting when various modes may be useful.

Response Mode	***External Source***	***Description***	***Suggested Usage***
Internal Only	Any	Standard operation. All channels operate independently.	The most common usage mode.
OR with internal	Other channel, channel via cable, or front panel	The channel takes a sample either when it has a signal or when its supervisory channel has a signal. External pulse stretcher normally disabled.	Usable for a shield or side channel to obtain the most accurate timestamp when a large enough signal arrives, but also to insure something is always recorded.
OR with internal	Timestamp bit	Works as normal channel but timestamp takes fixed rate samples to measure background/baseline.	Usable to monitor baseline shifts in resistive preamp detectors over time as the TS-based events are tagged as 'external' in the header.
AND with internal	Other channel, channel via	The subordinate channel is enabled for a time window after the master channel fires.	Enforced coincidence between two detector strips.

	cable		
AND with internal	Front panel bit	The channel is only enabled when the front panel signal is on (global veto)	Useful when a single signal from an upstream other detector defines the time window during which your detector's hits are valid.
AND with internal	Timestamp bit	The channel is regularly deadened for a period of time.	Only useful in cases where there is some timestamp-correlated noise in the system you wish to ignore (e.g. beam scraping, LN2 purges, etc.)
External Only	Any	The channel becomes a simple waveform recorder.	General purpose "oscilloscope" modes, ADC testing, etc. Generally not useful for physics.

Table 8 - Summary of External Discriminator Modes

Typical modes of Slave Digitizer Operation

Combining the ability of the Slave digitizer to receive trigger system commands along with information from the Master digitizer, plus the ability to define various external discriminator and external trigger modes, yields a wide variety of operational modes for various detectors. This section will explore a few of the options available to describe how they would work. The reader is reminded that in Digital Gammasphere, the Master digitizer monitors the Ge Center and BGO Sum signals from each of five detectors, whereas the Slave digitizer monitors the Ge Side and BGO Pattern signal from the same five detectors. In GRETINA, the Master digitizer monitors the central contact channel plus 9 of the segments; three Slave digitizers monitor the other 27 segments plus backup copies of the central contact. In DFMA, no Master-Slave digitizer relationships are used.

Gammasphere "Independent" Mode

In "Independent" mode, all channels of the Slave digitizer are set to use their own internal discriminators and ignore any external options. The Slave digitizer runs alongside the Master digitizer but not under the control of the Master digitizer. The clocks of the boards are synchronized and the Slave receives trigger commands. Events occur in the channels of the Slave digitizer only when signals applied exceed the thresholds set, and selection of events for readout works as normal in both internal and TTCL modes. Ge side channel signals show up in the readout only if the Ge Side contact was hit, but BGO Pattern channel signals are expected to always occur when the Ge Center contacts hit because the BGO Pattern signal definition (a pulse whose amplitude is proportionate to the binary pattern value optionally followed by individual charge values, that only is sent if the BGO Pattern logic senses a Ge Center hit in its own discriminator that runs parallel to digitizer operation) is certain to fire a BGO Pattern channel discriminator.

As the actual energy in the Ge Side contact is unlikely to be of interest in many "Independent" setups, the user is free to set the waveform length of the Ge Side (and BGO Pattern) readouts small to reduce bandwidth needs. Timing of the Ge Side signal relative to the Ge Center signal will be usable as the latency between boards is expected to be constant.

However, some timing offset over the front bus cable is inevitable and should be measured separately for calibration purposes.

Gammasphere "Clean", "Dirty" and "Module" Modes with Slave Digitizers

The Router trigger can provide Event Veto information to the Master digitizer via the SERDES link. When the Clean/Dirty rejection logic is enabled in the Router, a state machine processes each pair of discriminator bits sent by the Master digitizer every time either the Ge Center or BGO Sum bit is set. The rejection logic generates 5 "Veto" bits (one per channel pair) that are asserted if the most recent event in a pair of channels meets programmable time criteria. Assertion of a Veto bit causes the most recent event to be removed from the Pending Event Queue for the Ge Center and BGO Sum channels associated with that bit. The Veto bits are encoded by the Router into the normally unused 5th word of each trigger system command frame. By definition, the Veto bits are only issued in cases where both the Ge Center and the BGO Sum channels of a given Master digitizer have both been hit, so a common Veto is safe. The net result of this logic is that, when enabled, all "dirty" Ge/BGO channel event pairs are vetoed, so that only the "clean" Ge Center and "lonely" BGO Sum (BGO hit without associated Ge Center hit) channels will survive to be read out.

A critical point to understand is that the pileup logic feeds directly into the utility of this clean/dirty scheme. The discriminator bits are not available to the Router until ***after*** the pileup time ("m"+"k") has elapsed. Thus it is imperative that "m" and "k" be set to identically the same values in both the Ge Center and BGO Sum channels for a given detector or the clean/dirty logic cannot function. The cogent reader may also query whether pileup will affect whether the correct event can be selected to be vetoed. The answer to this is that the Pending Event Queue only contains ***Accepted*** events, not ***Extended*** events. This has two ramifications:

1) Since the PEQ only holds the timestamps of ***Accepted*** hits, a successful veto of one entry in the PEQ will by definition also veto any arbitrary number of following ***Extended*** events caused by pileup.
2) The only way to absolutely guarantee that accidental overlap of discriminator hits in pileup conditions does not create false excess vetoes is to run the discriminator in pileup-reject mode.

This veto logic works irrespective of whether the digitizer is in "internal" or "TTCL" mode. If using internal trigger the digitizers and routers will in the background strip out the dirty channel pairs by themselves. Additional event filtering may be implemented using the master trigger to sub-select events based upon multiplicity and/or coincidence with other detectors. When using clean/dirty vetoes in the Router trigger, the Router may be set to report raw multiplicity, clean-only multiplicity or lone-BGO multiplicity to the Master trigger in either the "X" or "Y" sums. This allows for highly filtered readouts such as only the clean Ge Center channels that participated in a clean multiplicity greater than some threshold.

Effects of Vetoes within the Slave Digitizer

By virtue of the front bus cable the Slave digitizer receives the same Veto signals that the Master digitizer does. The Ge Side channels of the Slave digitizer are enabled to process vetoes but the BGO Pattern signals are not. This means that when the Veto is issued for a particular detector by the Router trigger, the Ge Center, BGO Sum and Ge Side channels will be vetoed but the BGO Pattern signals are not. By implementing the Slave logic this way none of the BGO Pattern data of triggered events is ever lost, allowing offline software to reconstruct the scatter pattern and later reject those events where neighboring BGO panels are both hit (the “electric honeycomb”).

Use of External Discriminator modes with “clean/dirty” vetoes in a Slave digitizer

In many cases a “dirty” event will fire the Ge Center discriminator but not the Ge Side channel due to drift direction. This leaves the *previous* event in the Ge Side channel, if not yet read out, at risk of being incorrectly vetoed. Thus, in high rate events, it is important to insure that the Ge Side channel *always* has a discriminator firing each time the Ge Center does. This may be accomplished by setting the External Discriminator mode of the Ge Side channels to “OR” mode, if the Master digitizer is also capable of sending all five Ge Center bits to the Slave. This can be done by overloading the 5th word of each command frame as only five bits of the sixteen available are used by the Veto signals. If the Ge Side has already fired of its own accord, the copied Ge Center will come soon enough that it will be swallowed by the discriminator holdoff logic. If the Ge Side hasn’t fired, the copied Ge Center forces an event to protect the integrity of the Pending Event Queue.

“Pseudo-GRETINA” Mode

In “Pseudo-GRETINA” mode, the Slave digitizer is in complete thrall to the operation of one channel of the Master digitizer. The discriminator bit from the one channel of the Master is sent on bit 17 of the front bus cable and all channels of the Slave digitizer are set to fire only when the external signal is asserted, using the External Discriminator option built into all ANL digitizer firmware.

Thus, if the Master digitizer is set so that channels 1-9 fire only from the external signal and the external signal is set to channel 0, channels 1-9 are slaved to channel 0 and always sample whenever channel 0 fires. Similarly, in the Slave digitzer, all channels are set to use the external signal only and the external signal is set to be the copy of the discriminator bit from the Master digitizer’s channel 0. This then forces all channels in both boards to capture an event every time channel 0 of the Master digitizer fires, just like GRETINA.

The exact same trigger acceptance messages received by the Master digitizer are received by the Slave digitizer, so all channels in the slave will read out the same events that come from the Master whether in internal or TTCL mode. Propagation delay across the front bus cable will be small so the same time windows are used in both digitizers for TTCL mode. Obviously, delay chain parameters ‘m’, ‘k’, ‘d’ and ‘d3’ must be set the same in all channels of both Master and Slave digitizers.

Diagnostic capabilities using the on-board DAC

The global DAC multiplexer implemented for board-wide diagnostics allows a selected waveform from any of the 10 channels to be driven out the front panel DAC output. Each channel has a separate diagnostic waveform selection multiplexer that allows one of four diagnostic waveforms to be selected.

- Selection “00” copies the raw ADC data at the output of the P2 buffer of the channel to the DAC, for direct monitoring of input signals.
- Selection “01” generates a digital-like signal that shows, in time, when the baseline tracking logic is active.
- Selection “10” monitors the ADC data as it is entering the buffer used by the leading-edge discriminator.
- Selection “11” monitors the baseline tracker by driving the DAC with the slowly tracking baseline value itself (BASE_SAMPLE), so that the slew rate and filtering characteristics of the baseline logic may be viewed directly.

A suggested procedure at system startup is to monitor the baseline of a channel using an oscilloscope with no beam present but all electronics on. Reset the baseline logic of the channel. The signal should ‘home’ to the user-supplied estimated baseline and then begin tracking. Tracking will continue until the leading edge discriminator fires. For low event rates one should see the baseline hold steady during events but re-track between events.
EXTRA JUNK PASTE REGION

Digital Gammasphere detectors implement the concept of *clean* and *dirty* events. A *clean* event is one in which the Ge Center detector is hit but none of the surrounding BGO shields are struck within a given *overlap time.* If, however, any of the BGO are struck within the *overlap time* (plus or minus) the event is considered *dirty.* The overlap timer starts when either the Ge Center or the BGO Sum discriminator fires; if the other fires before the overlap time elapses the event is dirty. There is also a third category of event, a *BGO-only.*

The trigger system and the digitizer firmware work in concert to implement event veto mechanisms that may be enabled by the user. If the user enables the digitizer to accept clean/dirty vetoes, the firmware removes the top *accepted hit* in the queue from the pair of channels (Ge Center and BGO Sum) if the event is “dirty”, so that the “dirty” event is no longer available to be chosen as an *accepted event*. Similarly, *BGO-only* events may also be removed from the BGO channel individually. Obviously, the overlap time for clean/dirty rejection is limited and must be less than the pileup time and/or the event-to-event time at expected rates for proper operation. Normally for Gammasphere, the overlap time is ~600-700ns, less than the full 0%-100% rise time of ~1usec of the detector signals.

The firmware implements a flow control mechanism (“throttle”) that signals the trigger system if the on-board FIFO becomes too full. When asserted the throttle vetoes all trigger decisions so that the readout may catch up with the events already stored. To accommodate detectors with transistor-reset preamps, a “preamp reset kill” timer may be enabled that temporarily vetoes the discriminator when large pulses of polarity opposite to that desired occur to ensure no false firings during the reset interval.

Glossary

- *Digitizer* – a device that measures analog signal waveforms at a fixed sampling frequency, identifying ‘events’ based upon specific features in the waveforms.
- *Trigger* – a device that collects data from multiple digitizer modules plus the detectors and any other systems (e.g. target wheels, accelerator, etc.) and uses this information to, in real time, select which ‘events’ within the digitizer modules are read out.
- *Discriminator* – a device that inspects an analog waveform, developing a digital signal when change in the signal amplitude exceeds a threshold. There are two forms of *discriminator* used in the digitizer firmware, *leading-edge* and *constant-fraction*.
    - The *leading-edge* discriminator looks for the amplitude differential between two filtered data samples a programmable time apart exceeds a threshold. In short, this looks for the slope (dV/dt).
    - The *constant-fraction* discriminator uses a delay plus a fractional multiplier with intent to mark edges at the same relative fraction of total amplitude irrespective of the total amplitude. A *constant-fraction* discriminator, properly tuned, exhibits better timing than a *leading-edge*.
- *Hits* – for purposes of this document, a *hit* is the digital output of a discriminator.
- *Event* – for purposes of this document, an *event* is a packet of data initiated by a *hit*, that has subsequently been processed for things such as pileup or trigger to determine its fitness for readout.
- *Rise Time* - the time, or # of samples, from the beginning of a change in signal amplitude sufficient to cause discriminator response to the end of that (presumptively monotonic) rise or fall in amplitude.
    - The *rise time* is not usually expressed as an exponential time constant but as the amount of time the signal requires to traverse from start to end, either from 0% to 100% of final amplitude, or in some cases 10%-90% of amplitude.
- *Decay Time* – Waveforms associated with the detectors are assumed to be dual exponentials in shape with a relatively fast rise time (order of 1 microsecond) but much longer fall times. The decay time is an exponential decay time constant (e.g. 50 microseconds). To first order all signals processed are expected to follow the formula X(t) = A*[1-exp((t0-t)/tr)]*exp((t0-t/tf)) where
    - A is the amplitude of the signal
    - t0 is the time at which the signal occurs; X(t) is defined as 0 for all t<t0
    - tr is the rise time constant
    - tf is the decay time constant

- *Pipeline* – a form of logic implementation in which data values are continuously processed every tick of the clock. A pipeline never stops to perform any calculation; calculated values also continuously update every tick of the clock.
    - A common practice in pipeline design of functions such as accumulators is to initialize the accumulation by first zeroing the accumulation, then adding new input data for as many pipelined data values as the accumulator spans, then switch into a ‘tracking’ mode in which every clock tick one new data value is added while the eldest data value is subtracted. In this way the accumulator always contains the sum of all the samples it spans.
- *Down-sampling* – Compression of the number of data samples needed to span a given amount of time by replacing ‘n’ individual samples by one sample that is the sum of the ‘n’ samples, divided by ‘n’. As a practical matter, for firmware systems ‘n’ must always be an exact power of two. When implemented as part of a data pipeline, each sum of ‘n’ samples, divided by ‘n’, is issued every ‘n’ clocks. Thus down-sampling allows a fixed data volume to span a longer time, but at the expense of taking that same longer time to read out which may incur rate limitations.