Instruments That Advance The Art

Pixie Link User's Manual

Version 1.00

1 Contact Information

Address	XIA LLC 2744 East 11th St, Suite H2 Oakland, CA 94601-1443 U.S.A.
Phone	+1-510-401-5760
Support	support@xia.com
Web	xia.com

2 Disclaimer

Information furnished by XIA LLC is believed to be accurate and reliable. However, no responsibility is assumed by XIA LLC for its use, or for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of XIA LLC. XIA LLC reserves the right to change hardware or software specifications at any time without notice.

3 Warranty Statement

See XIA's Warranty and Support Policy

4 Safety

Please take a moment to review these safety precautions. They are provided both for your protection and to prevent damage to the Pixie Link. This safety information applies to all operators and service personnel.

4.1 Power Source

The Pixie-Net XL module is powered through an AC/DC wall adapter. Note that “plug-in” modules require additional power, notably any SFP and PMOD modules. **SFP modules above 1W are not supported.**

4.2 Servicing and Cleaning

These units contain no user serviceable parts. To avoid personal injury, and/or damage to the Pixie Link or connected equipment, do not attempt to repair or clean these units.

4.3 Voltage Ratings

Signals on the analog inputs must not exceed ± 5 V. Please review the pinout in the appendix before making any connections.

Signals on the gold MMCX connectors must not exceed 3.3 V.

5 System Overview

5.1 Architecture

The Pixie Link system architecture separates high-throughput data from standard control and monitoring data. The system routes these data streams through an engineered network built with Commercial Off-The-Shelf (COTS) gear. This central network bridges the hardware and the hosts, routing the specific data protocols (WebSocket, TCP, UDP) to the appropriate destinations. Host connections can be a modern web browser with java script enabled or a custom application using the provided C/C++ SDK.

5.2 Technical Specifications

Digitization	
Sampling Frequency	250 MSPS
Bit Resolution	14

Front Panel I/O	
Analog signal input	2x 8 analog inputs, 0.1" header with input impedance of 50 Ohm to GND. After termination and attenuation, up to ±2.5V DC can be added to compensate for signal DC offsets. After DC offset compensation, the ADC accepts signals in the range from 0V to (2V/analog gain). Designed for fast-rising, exponentially decaying signals. Step pulses and short non-exponential pulses can be accommodated with specific parameter settings. Staircase type signals from reset preamplifiers generally need to be AC coupled.
Veto Input	1 MMCX coaxial connector labeled "VETO" that accepts a 3.3 V logic signal. The signal should not exceed 5 or go below 0 V.
Pulser Output	1 MMCX coaxial connector, with 50 ohm output impedance, labeled "PULSE" that produces an analog periodic, exponentially decaying reference signal.
SFP	1x SFP cage labeled SFP Z that is unused at this time.
Rear Panel I/O	
RJ-45 Port	10/100M Ethernet for control network connection labeled LAN 1
Power	12V DC in. AC adapter requirements: 12V, 60W with a barrel plug 2.1mm I.D. x 5.5mm O.D., center positive
Clocks and Trigger	1 MMCX coaxial connector "CLK OUT", programmable clock out (3.3V) 1 MMCX coaxial connector "CLK IN", external clock input (3.3V) 1 MMCX coaxial connector "TRIG OUT" programmable trigger (e.g. PPS), 3.3V, output only These connections do not support 5 V logic signals.
Data Interface	
SFP (2x)	2x SFP cages labeled SFP 0 and SFP 1 used to send list-mode data via UDP.
Embedded webpage	Provides access to trace, MCA and statistics data.
Digital Controls	
Digital Gain	Arbitrary gain factor for channel matching
Offset	DC offset adjustment from –1.5V to +1.5V, in 65536 steps
Energy (Slow) Filter	Trapezoidal filter with peaking times 0.048 to 63.4 µs with adjustable flat top to eliminate ballistic deficit effects.
Trigger (Fast Filter)	Digital trapezoidal filter with adjustable rise time, flat top and threshold for pulse trigger detection.
Constant Fraction Discriminator (CFD) trigger	Programmable CFD length, delay and scaling factor
Coincidence	Programmable coincidence window: 40 to 1016 ns and reject unwanted hit patterns of the channels

Data collection	MCA number of bins 1Ki to 32Ki Waveform lengths and pre-trigger delay List mode data in binary form via UDP
Data Outputs	
MCA spectrum	32,768 bins, 32 bit deep (4.2 billion counts/bin) for each channel
List-mode event data	Pulse height (energy), time stamps, pulse shape analysis results, waveform data (up to 4Ki samples), constant fraction timing, and ancillary data.
Statistics	Real time, counting time, filter dead time, input and throughput counts.
Diagnostics	ADC trace
Dimensions	
H x W x L	3” x 8” x 8”

5.3 LEDs

The Pixie Link’s front panel has a total of 6 LEDs that display status information about the hardware state.

LED Name	Color	Function
ACTIVE	Orange	Indicates that data acquisition is in progress
ERROR	Red	Indicates that the system is in an error state. A slow blink indicates that the FPGAs are in sleep mode to save power.
LINK (SFP Z)	Variable	Unused
LINK (SFP 0)	Variable	Provides link status for SFP 0 cage
LINK (SFP 1)	Variable	Provides link status for SFP 1 cage
POWER	Green	Indicates the device is turned on and booted. Blinks if powered but not booted. Constantly on when powered and booted OK.

5.4 Signal Processing Pipeline

5.4.1 Analog Signal Conditioning

Each Pixie Link module features 8-16 independent analog input channels with an analog-to-digital converter (ADC). Each channel features a signal conditioning circuit (AFE) to minimize distortion in DC-coupled analog signals. The AFE’s primary job is to prevent aliasing using an anti-aliasing filter that removes high-frequency components from the incoming signal. This filter cuts off sharply at the Nyquist frequency, which is half the ADC sampling frequency.

Although the Pixie Link can work with many different signal forms, you can expect the best performance with unshaped output from a charge-integrating preamplifier or photodetector (e.g., photomultiplier tube or SiPM). You should ensure that the signal stays within the ADC's 2 Vpp input range. You can use the channel's offset DAC (VOFFSET) to help ensure that you maximize your signal's dynamic range.

5.4.2 Pulse Processing

A field programmable gate array (FPGA) implements real-time pulse processing and incorporates a first-level FIFO memory for each channel. The Analog-to-Digital Converters (ADCs) send their data stream to these units at the full sampling rate. While modern FPGAs can capture high-speed data, the complexity of the logic limits internal processing speed.

Therefore, the Pixie Link FPGA processes the data stream at a consistent internal clock speed. For 250 MHz variants, it deserializes each channel's 14-bit data stream into a 28 or 32-bit stream at 125 MHz. Using a pipelined architecture, the FPGA processes signals at this high rate without help from the onboard Digital Signal Processor (DSP).

The FPGA processing applies digital filtering to perform the same action as a shaping amplifier. The key difference is the filter type. In a digital environment, it's easy to implement finite impulse response filters. The Pixie Link uses trapezoidal filters. The filter's flat top typically covers the incoming signal's rise time, which makes the pulse height measurement less sensitive to variations in signal shape.

The FPGA contains several core processing elements that run continuously:

- **Trigger and Energy Filters**: A fast filter handles triggering, while a slow filter performs pulse height (energy) measurements. The FPGA issues triggers on each detected rising edge to latch timestamps and initiate other processes. The energy filter then sums the data at the appropriate time after each trigger.
- **Pile-up Inspector**: This logic checks if a second pulse arrives too soon after the first. If the second pulse arrives within the energy filter window it would corrupt the first pulse's height measurement. The logic flags both pulses as piled up. The inspector is not effective at detecting pileup on the rising edge of a pulse. To be recognized as distinct, pulses generally must be separated by at least their rise time. Therefore, for high count rate applications, you should use signals with the shortest possible rise times to minimize pileup.

The FPGA also manages data buffering and run statistics.

- **FIFO Memory**: The First-In, First-Out (FIFO) memory has two blocks. A smaller delay FIFO (1K samples) buffers ADC data to correctly position captured waveforms relative to a

user-defined pre-trigger delay. A larger storage FIFO (4K or 8K samples) captures waveforms of a user-defined trace length.

- **Data Readout**: The FPGA buffers event data until the DSP reads it out. For each event, the FPGA stores a complete set of timestamps, energy filter sums, pileup flags, and waveforms. User-defined acceptance settings specify which events are valid (e.g., accepting only events without pileup).
- **Counters**: The final processing elements are several counters that maintain run statistics, such as total counting time and the number of triggers.

6 Installation

6.1 Physical Set Up

The following steps detail the Pixie Link installation procedure. We do not include detector signal cabling in these instructions. They are not required to confirm system operation. We cover detector cabling in later sections.

Note: Please be aware that these instructions connect the Pixie Link to your local network. Consider contacting your network administrator prior to powering the device.

1. Connect an ethernet cable (1 Gbps capable) to the LAN 1 RJ-45 port.
2. Connect the other end of the ethernet cable to your network adapter.
3. Plug the supplied 12 V power adapter into the power connector on the Pixie Link's back panel.
4. Connect the power adapter to an AC power source (100-240 VAC, 50-60 Hz).
5. Verify that the POWER LED on the front panel is lit.

The system is "always-on" and will boot as soon as you plug it in. The system is operational when the boot operation completes.

6.2 Network Set Up

6.2.1 Control Network

Now that the Pixie Link has power and a network connection, it's time to set up the network configuration. You can find the system's MAC addresses listed on the product label under the device. The Pixie Link's control connection **uses DHCP by default**. You can change this to a static IP address if necessary.

You can find the DHCP assigned IP address using the control MAC address. The easiest way to do this is using tools like `nmap` or `arp`. Please be aware that these tools may require elevated

permissions. We recommend discussing the deployment with your IT team to ensure smooth integration with your network.

Once you have found the device's IP address. You can navigate to the embedded webpage using the IP. The webpage interface provides sufficient capability to configure the system and collect basic data like diagnostic traces, statistics and MCA histograms.

6.2.2 Data Network

The control network provides sufficient capability to configure the system and collect basic data like diagnostic traces, statistics and MCA histograms. List-mode data requires a few more pieces of commercial off-the-shelf networking gear:

1. a 10G networking switch with a minimum of 3 open SFP+ connections (ex. CRS309-1G-8S+IN),
2. 2x SFP modules for connecting SFP 0 and SFP 1 on the Pixie Link to the switch, and
3. a Windows or Debian host with the XDAQS API installed and connected to the switch.

You can find SFP 0 and 1's MAC addresses on the product sticker located on the bottom of the module. This concludes the physical connections and networking necessary for control and data collection.

7 Web Server

The Pixie Link has an embedded single-page web application that allows you to

- monitor hardware status,
- modify configuration parameters,
- view real-time data plots, and
- manage system settings.

You access this page via the DHCP assigned IP address. The webpage provides a no-code interface to reduce the amount of time it takes to configure and collect data.

The page header contains a colored indicator icon in the upper right corner. This indicator is green for online and red for offline.

Warning: Please ensure that you access the embedded webpage from a host computer on the same network as the Pixie Link device. VPNs or broader network access (ex. over the internet) may introduce display artifacts in the data streams.

7.1 Status

The Status tab displays the number of active sessions, network ping statistics (Current, Average, Minimum, Maximum RTT), and allows you to manually connect or disconnect sessions. The system allows up to a maximum of 5 active connections.

7.2 Parameters

The Parameters tab provides a comprehensive, searchable table of all configurable system parameters.

- **Searching and Filtering:** Use the search bar or the "Groups" dropdown to filter parameters by name or functional group.
- **Editing:** Click on a parameter's value to edit it. Changes are highlighted but not applied immediately.
- **Actions:**
    - **Submit:** Applies your edited values to the system.
    - **Refresh All / Hold All:** Syncs values from the device or locks them from refreshing.
    - **Load / Save:** Loads or saves a `.json` configuration file containing the current parameter configuration from/to your computer.
- **Information:** Click the "Info" icon next to any parameter to open a modal detailing its description, valid mode/type, word size, and specific bit fields.

7.3 ADC Trace

This tab provides a plot of ADC Traces.

- **Controls:** Click the "Run" (Play) button to start plotting data, and "Stop" to halt the trace.
- **Channel Selection:** Use the "Channels" dropdown menu to check or uncheck specific channels to isolate the data you want to see, or use "Clear"/"All" to manage bulk selections.
- **Periodic Polling:** Check the "Periodic" box and specify a period (in seconds) to continuously fetch and update the trace.

7.4 MCA

The Multi-Channel Analyzer (MCA) tab plots histogram data for active channels. Starting the MCA capture starts a data run that ends when you hit the stop button.

- **Controls:** Similar to the ADC Trace tab, you can start or stop the MCA capture.
- **Display:** Check or uncheck specific channels using the dropdown to toggle their visibility on the chart.
- **Periodic Polling:** Enable periodic mode to update the histogram at a set interval.

7.5 Omnitool

The Omnitool tab offers a built-in terminal console. This console provides you with the ability to directly update parameters, collect data, and view advanced information about the system.

- **Connecting:** Click the green "Connect" button to open a WebSocket connection and start a shell session.
- **Disconnecting:** Click the red "Disconnect" button to close the session.

7.6 System

The System tab manages device infrastructure and software.

- **Firmware:**
  - Allows you to upload and install Application Firmware (AFW), Bitfile Firmware (BFW), or Configuration Firmware (CFW).
  - Select a file, verify the type and version displayed, and press "Install". A progress bar will track the upload and installation phases.
- **Controller:**
  - Allows you to set the controller number and group ID
- **Network:**
  - Displays the current MAC and IP addresses.
  - Allows you to modify the Hostname, Subnet Mask, and toggle the IP Assignment Method between "DHCP" and "Static". Click "Submit" to apply network changes.

7.7 Log

The Log tab provides a tabular view of system events, debugging information, and errors.

- **Filtering:** Use the radio buttons at the top to filter logs by severity level: Error, Warning, Info, or Debug.
- **Exporting:** Click the "Download" button to save the current log table to your computer as a CSV file (`spu-log.csv`).

8 System Configuration

You configure the PixieNet XL using a comprehensive set of runtime parameters. These parameters allow you to precisely control the data acquisition pipeline, from the initial analog signal conditioning to the final network packet transmission.

The system categorizes these parameters into two primary scopes:

- **Controller Parameters:** These global settings apply to the entire module, covering clock generation, run timing, and network interfaces (e.g., `REQ_RUNTIME`, `CLK_CTRL`).
- **Channel Parameters:** These settings apply individually to specific signal paths, enabling independent configuration of gains, filters, and triggers (e.g., `TRIGGER_THRESHOLD`, `ENERGY_RISETIME`).

Functionally, the following operational groups organize the user-facing parameters:

- **Analog & Signal Conditioning:** These controls manage the Analog Front End (AFE), including DC offsets (`VOFFSET`) and baseline targets (`BASELINE_PERCENT`), to optimize dynamic range before digitization.
- **Triggering & Coincidence:** These settings define how the system detects and validates events. They include Constant Fraction Discriminator (CFD) settings for precise timing, energy thresholds (`ELO`, `EHI`) for veto logic, and inter-channel gating rules.
- **Digital Filtering:** These parameters configure the trapezoidal energy (`TAU`, `ENERGY_RISETIME`) and trigger filters to maximize energy resolution and throughput based on specific detector characteristics.
- **Data Acquisition & Waveforms:** These settings manage how the system accumulates histograms (MCA), captures raw waveforms (`TRACE_LENGTH`), and calculates Pulse Shape Analysis (PSA) variables.
- **Network & I/O:** These parameters control the flow of UDP list-mode data packets to ensure reliable communication with the host computer.

You can find a comprehensive list of parameters in Appendix C. You configure parameters either through the web page's Parameter tab or C/C++ API functions. When using the C/C++ API it's often helpful to read a parameter and store the value before modification.

WARNING: You should only modify parameters using the provided APIs. Modification of these parameters outside of these functions may result in system instability and data loss.

The system boots using default parameters. The system **does not persist** parameter updates across reboots. You can save your configurations using the "Save" button on the web page's Parameters tab. You can then load these settings files after the system boots using the Parameter tab "Load" button.

Warning: Do not modify the contents of the settings file directly. This will lead to system instability and potential data loss.

8.1 ADC Waveforms

The module collects and stores untriggered ADC waveforms for diagnostic purposes. Pulses from the detector should have the following characteristics:

1. fall in the range of the ADC,
2. have a positive amplitude with a rising edge,
3. have an exponentially decaying tail.

Proper conditioning of the digitized signal plays a critical role in the system performance.

8.1.1 Signal polarity

The FPGA's filtering only works on signals with a rising edge. If the signal has the wrong polarity, you can adjust this by modifying `CHANNEL_CSRA Bit 5`. Setting the bit will invert the signal before it enters the filtering pipelines.

8.1.2 Baseline

The Pixie Link continuously calculates the signal baseline and feeds the value into a running average. The baseline is a measurement of the DC offset of the input signal with corrections to the exponential decay of the pulse. It is not equivalent to the DC offset of the signal in value. It is proportional to the DC offset.

The baseline is a critical component of the pulse height computation. The computation assumes the signal decays exponentially and has a single decay constant. If these assumptions are not met, then the accuracy of the baseline measurement will be compromised.

If the signal is not in range, you can adjust each channel using the `VOFFSET`. We recommend setting the baseline at ~10% of the ADC range to allow for drifts, undershoots or mirror charges.

8.1.3 Visualization

You can validate your signal polarity and baselines by collecting the ADC waveforms and viewing them via the on-board webpage.

1. Navigate to the ADC Trace tab
2. Choose the channels of interest from the drop down
3. Click the "Play" button.

You can adjust the time between samples using `XDT`. This can be useful when viewing signals with long tails.

Once the digitized signals look good, you can move onto setting up the filter parameters.

8.2 Trigger filter

The trigger filter detects the incoming signals. When the filter reaches the specified threshold the system latches this time as the signal's arrival time. Configuration of this filter ensures that the system processes your input signals correctly. Incorrect configuration of this filter may lead to double triggering, triggering in the noise, or create artifacts in the data stream.

There are three main parameters necessary to configure the trigger filter.

1. `TRIGGER_RISETIME` : A longer trigger filter rise time averages more samples and thus allows setting lower thresholds without triggering on noise.
2. `TRIGGER_FLATTOP`: A longer trigger filter flattop time makes it easier to detect slow rising pulses.
3. `TRIGGER_THRESHOLD`: Typically the threshold should be set as low as possible, just above the noise level.

Setting the trigger filter parameters requires a balance between length and noise requirements. Longer filters allow you to reduce the threshold but may cause you to miss fast signals. A filter that's too short makes you more susceptible to signal noise and can lead to extraneous triggers.

8.2.1 CFD Triggering

The Pixie Link system employs Digital Constant Fraction Discriminator (CFD) triggering to precisely determine an event's arrival time. Unlike a simple leading-edge discriminator the CFD technique processes the input signal through a delay, attenuation, and subtraction network. This network generates a trigger at a constant fraction of the pulse height. It produces a bipolar signal, and the system uses the zero-crossing point of this resulting signal as the robust, amplitude-independent time stamp for the event.

The FPGA runs at a slower clock speed than the ADC. The ADC creates a sample every 4 ns, while the FPGA runs on an 8 ns clock cycle. To keep up, the FPGA processes multiple ADC samples in parallel. The system records the **CFD source trigger bit** to account for difference in clock speeds between the ADC and FPGA. The system records the CFD values on either side of the zero crossing point. You extract these values from the list-mode data to calculate the **CFD fractional time.** This value provides sub-sampling accuracy for the signal arrival. See the appendix for detailed information on the time reconstruction.

You can enable the CFD Triggering by enabling CHANNEL_CSRA bit 10. You then use `CFD_DELAY`, `CFD_SCALE` and `CFD_THRESHOLD` to adjust the CFD performance for your system. Use `CFD_THRESHOLD` to suppress noise induced zero-crossings. If you set the `CFD_THRESHOLD` too high, the system may force the CFD to trigger and render the calculation useless. The system records this error in the list-mode data and sets the CFD outputs to 0.

8.3 Advanced triggering

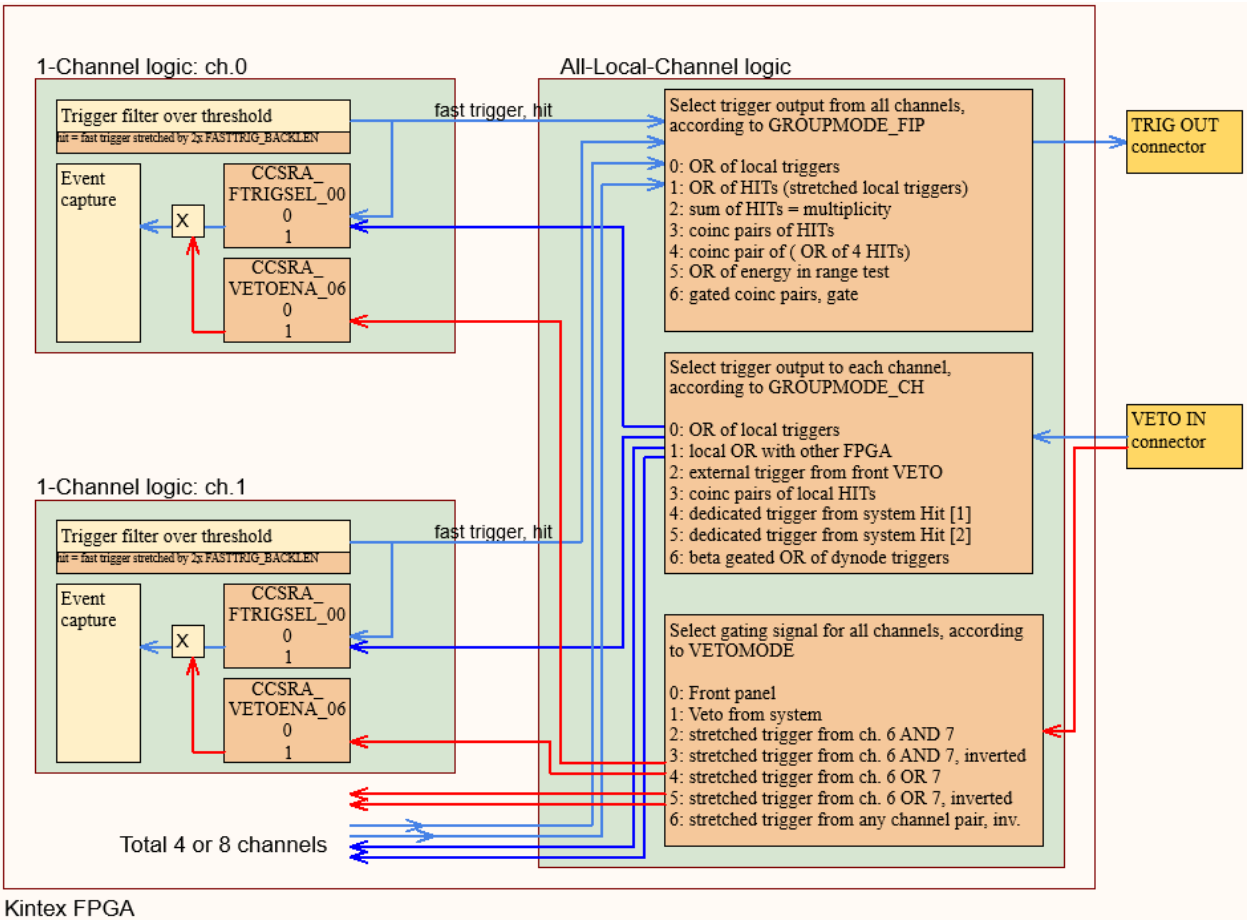
A Pixie Link channel uses a number of signals and FIFOs to facilitate complex triggering requirements encountered by experiments. These signals can be internal to the controller or come from external sources.

Be aware that imposing complex triggering rules and logic may result in an unrecoverable loss of data. This isn't unlike traditional electronics where improper gate signal generation or cable delays prevent signal processing. We recommend using the simplest possible triggering for your application, and impose more complex rules in offline analysis.

In the simplest case, each channel operates independent from all other channels. When the channel's trigger filter generates the "local fast trigger" the record moves through the data pipeline to the output. You set this mode by disabling `CHANNEL_CSRA Bit 0`.

Enable distributed (group) trigger mode by enabling `CHANNEL_CSRA Bit 0`. Enabling this mode combines local triggers and external gates/vetos with the local fast triggers. The triggering occurs in two distinct domains: FIPPI and channel.

You control the FIPPI triggering using `GROUPMODE_FIP` and channel triggering using `GROUPMODE_CH`. The following picture describes the triggering logic in more detail. The logic for the two domains reside in the right-hand box labeled "All-Local-Channel logic".



8.3.1 FIPPI domain

Trigger decisions made within this domain select trigger output from all channels. The output of this trigger can be used in other systems using the `TRIG OUT` connector. You control this logic using the `GROUPMODE_FIP` parameter. This parameter may take on the following values.

Value	Description
0	A 1-cycle pulse built using the OR of local fast triggers
1	Stretches the local triggers by 2x `FASTTRIG_BACKLEN` (=hit) then builds an OR
2	Determine multiplicity by summing HITs. **Does not** produce output to `TRIG OUT`.
3	Trigger on coincident pairs of HITs. **Does not** produce output to `TRIG OUT`.
4	Produce a trigger pulse when any Channel 0-3 AND any Channel 4-7 are hit
5	Generate a trigger using an OR of the `energy in range` test result for all channels

8.3.2 Channel domain

The channel logic gets applied to each channel by enabling `CHANNEL_CSRA[1]`. This bit **must be** enabled to use the group trigger for event capture.

Value	Description
0	Trigger using an OR of local triggers
1	Uses an OR combination of local triggers and the TRIG OUT signal from FPGA 0. Channels in FPGA 1 (4-7 or 8-15) can be triggered by any channel in FPGA 0 (0-3 or 0-7). This **cannot** be applied in reverse.
2	Generate the trigger using the VETO front panel connector. Choose the VETO's rising edge using `MODULE_CSRA[7]`. Otherwise use the falling edge.
3	Generate a trigger when two neighboring channels are hit.

8.3.3 Additional notes

1. The word `stretch` or `length` refers to how long the signal persists in time. `Delay` refers to when the signal starts in time.
2. If your channel should only be triggered by other channels, then set the `TRIGGER_THRESHOLD` to zero.
3. The list-mode data encode the hit signal pattern latched at time `FASTTRIG_BACKLEN` after a trigger to allow offline selection of coincident events even when running in local trigger mode.
4. If CFD mode is enabled, the distributed trigger "arms" the timestamp and waveform capture, but the timing is determined by the CFD zero crossing.
5. When triggering a channel with another channel or with an external signal, the energy computation may be incorrect. Enable `CHANNEL_CSRC[9]` to use the channel's local trigger for the energy computation in group trigger mode.

8.4 Gating

8.4.1 VETO signals

Unwanted triggers may be gated using the VETO MMCX connector on the module's front panel. Enable use of this signal by enabling `CHANNEL_CSRA[6]`, and configure it by setting `VETOMODE` to 0. Enabling these conditions have the following effects on the channel

1. The channel's fast trigger gets suppressed and will not contribute to
    a. list-mode data records,
    b. the channel's dead-time calculation,

    c. output counts, and
    d. passed pile-up counts.
2. Input count statistics will not be affected.
3. Fast triggers may still be used for coincidence purposes.
4. Pile-up inspection logic shall still apply.
5. A "gate dead-time" counter shall count the time that the VETO is active.

The EXTERN_DELAYLEN acts like a cable delay. This delay affects the overlap of the signal with the VETO. To aid delay set up, you can enable CHANNEL_CSRC[11] to include trigger and delay information in the untriggered trace output. You **must** also ensure that XDT is no larger than 0.064 us to avoid missing signals.

We encode the information in the first 5 bits of the trace sample. You can decode these bits to overlay the listed information with the trace.

Bit	Signal
0	TTCL Approved
1	Latch used to capture energy sums
2	Latch used to capture coincidence information
3	VETO
4	Local fast trigger

8.5 Energy filter

The energy (slow) filter determines the signal's amplitude. This is not simply the difference between baseline and maximum sample of the pulse. It's the result of the energy filter corrected for the exponential decay of the signal and DC offset. The result of this filter maps back to the energy of the incident particle in the detector. You configure the energy filter using four primary parameters:

1. ENERGY_FLATTOP: The flattop should be larger than the longest pulse rise time.
2. ENERGY_RISETIME: The risetime can be varied to balance the resolution and throughput.
    a. Energy resolution often improves with increasing energy filter rise time, up to an optimum when longer filters only add more noise into the measurement. The energy filter dead time TD is about $2 \times (T_{rise} + T_{flat})$, and the maximum throughput for Poisson statistics is 1/(TD*e). For HPGe detectors, a rise time of 4-6 µs and a flat top of 1 µs are usually appropriate.

3. TAU: Represents the incoming signal's decay time. The system assumes the signal has a single decay constant. The system uses the decay time to compute and remove the tail of previous pulses when computing the pulse height of the current pulse.
4. SLOW_FILTER_RANGE: A module parameter that sets the number of ADC samples to average ($2^{SLOW\_FILTER\_RANGE}$) before entering the energy filter logic. For example, setting SLOW_FILTER_RANGE to 2 will average 2^2 samples. This allows the system to accommodate longer signals.

NOTE: Long energy filter settings may increase the rate of pulse pile-up at high rates. This happens because you increase the likelihood of a second signal arriving within the energy filter window for the first signal.

NOTE: If an event is a piled up event, or if the event's trace out-of-range flag is set to 1, the event's energy will be defaulted to 0.

8.5.1 Filter Range

The filter range allows the system to accommodate signals from tens of nanoseconds to tens of microseconds. Choose the smallest energy filter range that allows setting the optimum energy filter rise time. Larger filter ranges allow longer filter sums, but increase the granularity of possible values for the energy filter risetime and flattop and increase the jitter of latching the energy filter output relative to the rising edge of the pulse. This is usually only important for very fast pulses.

The energy filter rise time and flattop sum must be 127 decimated clock cycles. This affects your filter granularity and lengths according to the following table.

Filter range	Filter granularity	max. $T_{rise}+T_{flat}$	min. T_{rise}	min. T_{flat}
1	0.016µs	2.032µs	0.032µs	0.048µs
2	0.032µs	4.064µs	0.064µs	0.096µs
3	0.064µs	8.128µs	0.128µs	0.192µs
4	0.128µs	16.256µs	0.256µs	0.384µs
5	0.256µs	32.512µs	0.512µs	0.768µs
6	0.512µs	65.024µs	1.024µs	1.536µs

8.6 List-mode data

The Pixie Link's list-mode data output has a number of options to support a wide array of applications.

8.6.1 Full-speed ADC waveform capture

The Pixie Link can capture full-speed ADC waveforms as part of the list-mode data output. These data are different from the ADC waveforms referenced earlier. These are **not sampled** and represent the signal as digitized by the ADC at full speed.

These data may be enabled via CHANNEL_CSRA Bit 8. You control both the length of the capture and the pre-trigger trace delay using the parameters TRACE_LENGTH and TRACE_DELAY. The trace delay cannot be longer than the trace length. **The maximum trace length may not exceed 2.688 us (672 samples at 250 MSPS).**

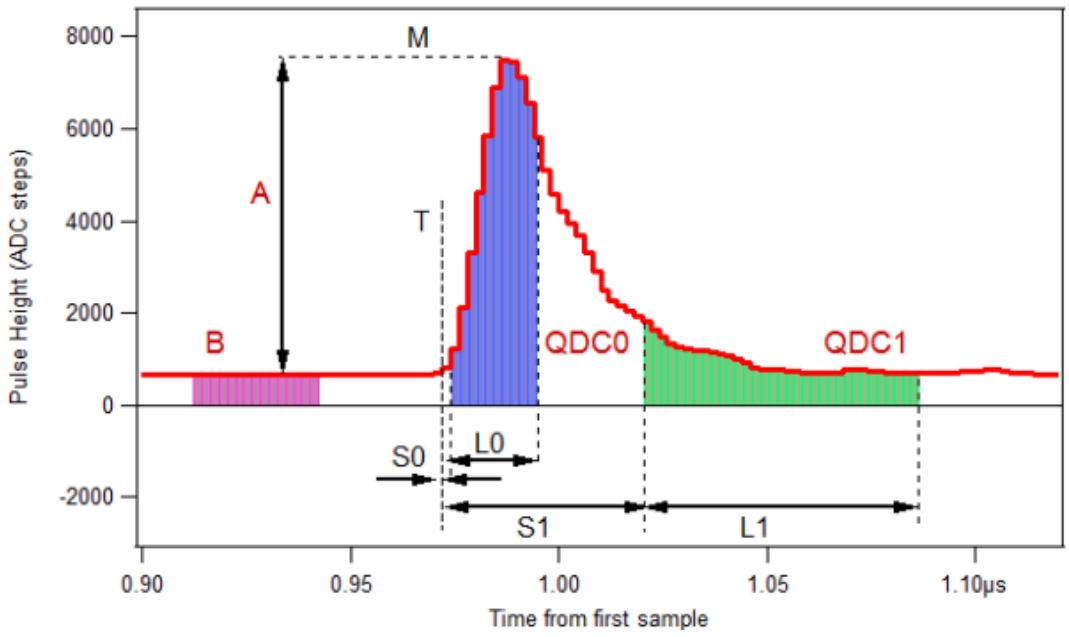
8.6.2 Pulse Shape Analysis

Some detector materials generate different responses to incident particles. For example, a liquid scintillator generates different light responses to neutrons and gamma rays. Phoswich detectors consist of multiple scintillating layers that produce different pulse shapes depending on which layer absorbs the radiation. A suitable pulse shape analysis (PSA) can detect such differences.

The Pixie Link implements a variation on the Charge Comparison Method. The system calculates two sums over user defined regions. The system calculates the regions based on the trigger position (T). The trace baseline information (B) and maximum pulse height (M) provide the signal amplitude (A). You access these data via the list-mode data, and can use them to calculate the PSA Ratio using the following formula:

$$R = QDC1 / QDC0$$

The following image demonstrates the relationship between each of these parameters.



Warning: The PSA calculation uses the same memory structures as the QDC, which prevents these functions from operating at the same time.

To enable the PSA sums, you should disable CHANNEL_CSRA Bit 9. You may customize the PSA calculations using the following parameters.

- QDC_DELAY[0] and QDC_DELAY[1] set S0 and S1.
- QDC_LEN[0] and QDC_LEN[1] set L0 and L1.
- PSA_THRESHOLD allows you to detect the pulse's rising edge.
- QDC_DIV is a dividing factor of 4 or 32 on the PSA sums. You use it to keep the sums from overflowing the 16-bit value (65535).

The above parameters have the following constraints:

1. QDC1 must finish last (i.e. S1+L1 > S0+L0)
2. QDC1 must finish within the energy filter (for pileup rejection)
3. L0, L1 must be between 2 and 60, and a multiple of 2
4. L0+S0, L1+S1 must be between 0 and 250
5. The difference of S0 and S1 must be even (e.g S0 = 1, S1 = 17)

9 Data Acquisition

9.1 Run Synchronization

You can configure the Pixie Link channels to synchronize the timestamps to 0 at the beginning of a data run by enabling SYNC_AT_START.

9.2 MCA

This data mode is ideal for situations where the energy spectra and statistics are the only data of interest. MCA data runs histogram the 16-bit energy calculation into the MCA memory for each channel. The MCA memory has 32K bins. To keep all values in range, the system divides the energy value by the parameter BINFACTOR. For example, setting this value to 1 (default) will divide by 2.

The easiest way to view the histograms and statistics data is via the embedded webpage. The webpage provides a no-code way to collect these data and validate system performance.

9.3 List-mode

The system collects list-mode data for each trigger, capturing detailed event information essential for data storage, playback and analysis. The system efficiently packages the data as a compact binary record. The record's data elements are 16-bit unsigned integers in little endian format. The system transmits the records via UDP to the designated receiver using the SFP 0 and SFP 1 network ports.

You will need to configure the Source IP addresses and ports so that the FPGAs can send the list-mode records over UDP. You will also need to configure the Destination IP addresses, MAC addresses, and ports. Misconfiguration of these items may result in data loss.

9.3.1 List-mode data structure

The list-mode data structure comprises two parts: the event header and the trace. The trace data will always follow the header if you enable trace capture. The event header contains all the necessary information to validate the data and decode the trace.

You can use the `Event header length` and the `trace blocks` as a sanity check for the data record. The record length in words should be equal to the event header length plus the number of trace words. For example,

```
C/C++
TRACE_WORDS_PER_BLOCK = 32
record_length_words = header_length + trace_blocks * TRACE_WORDS_PER_BLOCK
```

If your UDP payload size in words doesn't match the calculated record length, then you should assume data corruption and discard the record.

Event header

The list-mode event header contains 31 16-bit unsigned integers in little endian format.

Word	Name	Description
0	Header length	The length of the event header, which is fixed at 31 (`0x1F`) words. Useful as a validation anchor.
1	ModuleType	A number identifying the Pixie module type, including ADC bits and rate.
2	Data format	A number identifying the data format and fixed at `0x410`.
3	Event pattern	Unused. Ignore.
4	Event information	Event status information encoded into bits of the value. Each bit represents a different state. We reserve some bits for internal use. User facing bits are Bit 1 -> Set if TTCL logic approved the event Bit 2 -> Set if the event was piled-up Bit 5 -> Set if the signal saturated the ADC range at the time of the fast trigger. Bit 6 -> Set if the front panel / VETO logic was high at the time of the trigger.

Word	Name	Description
5	Trace blocks	The number of trace data blocks following the event header. Each block contains 32 16-bit words and is 64 bytes in length.
6	reserved	Unused. Ignore.
7	TrigTimeLO	The lower 16-bits of the trigger time in units of 1 ns.
8	TrigTimeMI	The middle 16-bits of the trigger time in units of 1 ns.
9	TrigTimeHI	The upper 16-bits of the trigger time in units of 1 ns.
10	TrigTimeX	The final 8-bits of the trigger time in units of 1 ns. This time should not be used to reconstruct the signal arrival time.
11	Energy	The result of the trapezoidal energy filter calculation of the signal's energy.
12	ChanNo	The system channel number that captured this record.
13	PSA max (or ampl)	Signal amplitude as determined by the PSA algorithms
14	reserved	Unused. Ignore.
15	PSA base	Baseline calculated by the PSA algorithms
16	PSA Sum 0	The value of the early PSA sum (QDC0) over the signal.
17	PSA Sum 1	The value of the late PSA sum (QDC1) over the signal.
18	reserved	Unused. Ignore.
19	CFD_raw[0]	Contains encoded information about the CFD trigger source and whether or not the system forced the CFD trigger. See the CFD appendix for more information.
20	CFD_raw[1]	Contains the low 16-bits of the first CFD sum. See the CFD appendix for more information.
21	CFD_raw[2]	Contains the upper 8-bits first CFD sum and the lower 8-bits of the second CFD sum. See the CFD appendix for more information.
22	CFD_raw[3]	Contains the upper 16-bits of the second CFD sum. See the CFD appendix for more information.

Word	Name	Description
23	EXT_TS	The lower 16-bits of the external timestamp in units of 1 ns.
24	EXT_TS	The middle 16-bits of the external timestamp in units of 1 ns.
25	EXT_TS	The upper 16-bits of the external timestamp in units of 1 ns.
26	reserved	Unused. Ignore.
27	reserved	Unused. Ignore.
28	TransferInfo	15-8: multi-package event segment number 7-0: channel number
29	reserved	Unused. Ignore.
30	reserved	Unused. Ignore.

Trace data

If trace recording is enabled, trace data follows the last word of the event header. Use `Trace blocks` from the list-mode event header to determine how many 16-bit words the trace contains. Each trace block contains 32 16-bit words. For example, if you set a length of 2.688 us, then the system records 672 samples (i.e. 16-bit words), which is 21 trace blocks.

9.4 Run Statistics

The system collects statistics data for both MCA and List-mode data runs. The module updates these onboard statistics once per second. The onboard run statistics consist of four primary values: run time, live time, input counts and output counts. We detail each of these below.

The **run time** tracks the time during which the Pixie Link had an active data run. This time does not include time spent during run initialization. You use this time to calculate the output count rate from the hardware. This value is the same for all channels in a module.

The FPGAs keep track of the channel's **live time**. This time represents the amount of time the channel was capable of processing a trigger. It starts counting when the DSP finalizes all the run initialization routines. It ends when receiving the "end run" signal. The count pauses under the following conditions:

1. when the ADC signal goes out-of-range,
2. when the on-board FIFOs fill, or

3. when the trigger filter is unable to issue triggers because the filter output is above threshold.

This value is the most accurate measure for how active the channel was.

NOTE: The channel's live time should always be less than or equal to the module's run time.

The Pixie Link's FPGAs count the number of triggers in each channel as the **input count**. This represents the number of incoming signals to the system and is not affected by coincidence settings. You can calculate the input count rate by dividing this number by the channel's live time.

The **output count** comes from the number of events processed by the system while the system is capable of collecting data. It only counts events that pass pile-up, saturation, veto, gate, or other coincidence logic. You can calculate the output count rate by dividing this value by the live time.

Note: You may notice channels with no inputs having an output. This happens when the channel gets processed as part of a "group trigger".

10 Appendix A - CFD Computation

10.1 Decoding CFD Information

The CFD information contained in `CFD_raw[0]` consists of 2 pieces of information: the **CFD trigger source bit** and the **CFD forced trigger bits**.

The FPGA processes samples in groups, but the CFD logic still checks every ADC sample for a zero-crossing. The **CFD trigger source bit** identifies the specific sample that caused the trigger . This information allows you to reconstruct the event's precise timing.

The system sets the **CFD forced trigger bits** when it forced the CFD to trigger. A forced trigger is a failsafe that prevents the system from freezing if it cannot find a trigger signal's zero-crossing point within a specific time window. In this situation, the CFD fractional times will always be 0.

For example, if decoding in python:

```python
cfd_src = 0
cfd_forced = False
cfd_info = cfd_raw0 & 0x7
  if cfd_info == 1:
    cfd_src = 1
  elif cfd_info == 3:
```

```
    cfd_forced = True
```

10.2 CFD Fractional time

You reconstruct the CFD fractional time using the two data points on either side of the zero crossing point (ZCP). The system stores these values in the list-mode data in `CFD_raw[1-3]`. Reconstruction of these values requires a bit of bit arithmetic. The following code snippet demonstrates how to reconstruct the CFD fractional time in seconds with the CFD information.

Python
```
# The ADC Sampling period in seconds.
ADC_PERIOD_S = 4e-9
cfd_out1 = cfd_raw1 + ((cfd_raw2 & 0x00FF) << 16)
cfd_out2 = 0x1000000 - (((cfd_raw2 & 0xFF00) >> 8) + (cfd_raw3 << 8))
cfd_phase = float(cfd_out1 / (cfd_out1 + cfd_out2)) - cfd_src
cfd_fractional_time = cfd_phase * ADC_PERIOD_S
```

11 Appendix B - Arrival time reconstruction

The Pixie Link's signal processing FPGAs use a 125 MHz clock to calculate the trigger filter timestamp. The time stamp is 48-bits. Its resolution matches the FPGA's at 8 ns. CFD triggering allows for sub-sampling time resolutions. Arrival time reconstruction within the system requires taking both of these times into account.

11.1 Filter time

Time reconstruction begins with the trigger filter timestamp. You obtain these from the list-mode data words 7-9. The data are stored in units of 1 ns, so we convert to seconds by multiplying by this conversion factor.

Python
```
trigger_time = (trigger_time_low | (trigger_time_medium << 16) |
                (trigger_time_high << 32)) * 1e-9
```

NOTE: This is the same equation you'd use to calculate the trigger time when the CFD is forced or to calculate the external timestamp.

11.2 Including CFD time

Once you have the CFD time calculated, just add it to the filter time above.

```
None
time = trigger_time + cfd_fractional_time
```

12 Appendix C - Parameter Listing

12.1 AUX_CTRL

Scope: controller | **Group:** control | **Unit:** none | **Tag:** advanced

The only user facing bit in this value is Bit 0, which turns the front panel PULSER signal on (=1) or off (=0).

12.2 BASELINE_PERCENT

Scope: channel | **Group:** data.baseline | **Unit:** percent | **Tag:** none

The percentage of the ADC's bit resolution that the system will attempt to put the signal baseline when executing an automatic adjustment. Can vary between 0 and 99.

12.3 BINFACTOR

Scope: channel | **Group:** data.mca | **Unit:** none | **Tag:** mca

Divides the measured energy by 2^N before binning. Where N can range from 1 to 16.

12.4 BLAVG

Scope: channel | **Group:** filter.baseline | **Unit:** none | **Tag:** none

Used to set the exponent in the geometric weighted algorithm used to average the signal baseline for energy calculations. The baseline standard deviation is proportional to sqrt(2^BLAVG). Optimal energy resolution is generally found at a value of 65533 (3) or 65532 (4). This value is input as a 2's complement, and can take the following values and meanings.

- 0 -> 0
- 65535 -> 1
- 65534 -> 2
- …
- 65528 -> 8

For example, to set BLAVG to 4 input 65532.

12.5 BLCUT

Scope: channel | **Group:** filter.baseline | **Unit:** none | **Tag:** none

If the calculated baseline is above this value, then the system will not include the baseline in the running baseline average. Values can range between 0 and 65535.

12.6 CFD_DELAY

Scope: channel | **Group:** cfd | **Unit:** fippi_clock_ticks | **Tag:** none

Sets the delay used to calculate the on-board CFD. Can vary between 1 and 63 in 8 ns samples. See the CFD discussion for more details.

12.7 CFD_SCALE

Scope: channel | **Group:** cfd | **Unit:** none | **Tag:** none

Sets the scaling factor used to calculate the on-board CFD according to the following table. See the CFD discussion for more details.

Value	Scale
0	1.000
2	0.75
3	0.625
4	0.5
5	0.375
6	0.250
7	0.125

12.8 CFD_THRESHOLD

Scope: channel | **Group:** cfd | **Unit:** none | **Tag:** none

Sets the threshold used to calculate the on-board CFD. Can vary between 1 and 65535. See the CFD discussion for more details.

12.9 CHANNEL_CSRA

Scope: channel | **Group:** control | **Unit:** none | **Tag:** adc, mca, list-mode, coincidence, pileup

Controls many aspects of the channel's behavior.

Bit name	Bit	Description
CCSRA_FTRIGSEL	0	Channel fast trigger selection (=1: use distributed trigger from internal or external sources; =0: use this channel's local fast trigger)
CCSRA_GOOD	2	Set this channel as a Good channel (=1) or a not Good channel (=0). When a channel is set to be a not Good channel, it still generates local fast triggers, which could be used in multiplicity computation, etc., but this channel will not record list-mode data or MCA data or ADC trace data.
CCSRA_SYNCDATAACQ	4	Choose the level of synchronous data acquisition for this channel (=1: stops taking data when the trace or header DPM for any channel is full; =0: stops taking data only when the trace or header DPM for this channel is full)
CCSRA_POLARITY	5	Choose this channel's input signal polarity (=1: invert input signal's polarity; =0: do not invert input signal's polarity).
CCSRA_VETOENA	6	Enable (=1) or disable (=0) this channel's veto. If veto is enabled, this channel's fast trigger will be vetoed by the external or internal veto logic.
CCSRA_TRACEENA	8	Enable (=1) or disable (=0) trace capture in the list-mode run for this channel
CCSRA_CFDMODE	10	Enable (=1) or disable (=0) CFD trigger in the list-mode data for this channel. CFD trigger is used to latch sub-sample timing for the event time of arrival or timestamp.
CCSRA_PILEUPCTRL	15	If disabled (=0), then collect all records regardless of pile-up condition. If enabled (=1), then reject records flagged as pile-up.

12.10 CHANNEL_CSRC

Scope: channel | **Group:** control | **Unit:** none | **Tag:** qc, coincidence, pileup

Controls channel coincidence and enables FIPPI signal simulation.

Bit name	Bit	Description
CCSRC_INVERSEPILEUP	0	If enabled (=1), then CCSRA_PILEUPCTRL is ignored and **only** piled-up events get recorded.
CCSRC_RBADDIS	6	When the input signal goes out of range while the energy filter accumulates the required number of samples, then either record (=1) or don't record (=0) the energy.
CCSRC_PAUSEPILEUP	7	If enabled (=1), then the system pauses pile-up inspection for a short time after a trigger.
CCSRC_CHECK_ELIMITS	8	If enabled (=1), then uses user defined energy value limits to generate a logic pulse on the MMCX output of length VETO_STRETCH.
CCSRC_LOCAL_ENERGY	9	If enabled (=1), then the channel will use its local trigger for the energy filter capture even if group triggering is enabled. This can be useful when the signal in this channel is early or late compared to signals generating the group trigger.
CCSRC_SIMADC	10	If enabled (=1), then the ADC data stream gets replaced with a simulated data stream generated by the FIPPI.
CCSRC_LOCAL_ENERGY	11	If enabled (=1), then the lower 5 bits of the diagnostic ADC traces show triggers and gating information. List-mode traces are not affected.

12.11 CLK_CTRL

Scope: controller | **Group:** clock | **Unit:** none | **Tag:** advanced

Clock generation control bits.

12.12 DEST_IP0

Scope: controller | **Group:** network | **Unit:** none | **Tag:** list-mode

Sets the IP address for the hardware receiving data from SFP 0.

12.13 DEST_IP1

Scope: controller | **Group:** network | **Unit:** none | **Tag:** list-mode

Sets the IP address for the hardware receiving data from SFP 1.

12.14 DEST_MAC0

Scope: controller | **Group:** network | **Unit:** none | **Tag:** list-mode

Defines the hardware that will receive the list-mode UDP packets from SFP 0.

12.15 DEST_MAC1

Scope: controller | **Group:** network | **Unit:** none | **Tag:** list-mode

Defines the hardware that will receive the list-mode UDP packets from SFP 1.

12.16 DEST_PORT0

Scope: controller | **Group:** network | **Unit:** none | **Tag:** list-mode

The port used to receive data from SFP 0.

12.17 DEST_PORT1

Scope: controller | **Group:** network | **Unit:** none | **Tag:** list-mode

The port used to receive data from SFP 1.

12.18 EHI

Scope: channel | **Group:** coincidence | **Unit:** none | **Tag:** advanced, veto

The maximum value the event energy may take to generate a logic pulse on the TRIG OUT MMCX connection with a length defined by the VETO_STRETCH. Only checked when CHANNEL_CSRC Bit 8 is enabled (=1).

12.19 ELO

Scope: channel | **Group:** coincidence | **Unit:** none | **Tag:** advanced, veto

The minimum value the event energy may take to generate a logic pulse on the TRIG OUT MMCX connection with a length defined by the VETO_STRETCH. Only checked when CHANNEL_CSRC Bit 8 is enabled (=1).

12.20 ENERGY_FLATTOP

Scope: channel | **Group:** filter.energy | **Unit:** microsecond | **Tag:** mca, list-mode

The flattop for the trapezoidal energy filter. This value may range from 0.032 to 62.976 us.

12.21 ENERGY_RISETIME

Scope: channel | **Group:** filter.energy | **Unit:** microsecond | **Tag:** mca, list-mode

The amount of time the trapezoidal energy filter takes to reach the flattop and return to baseline after the flattop. This value may range from 0.048 to 62.976 us.

12.22 EXTERN_DELAYLEN

Scope: channel | **Group:** coincidence | **Unit:** microsecond | **Tag:** advanced

Effectively applies a cable delay to the channels that can be used to correct for cable length differences, synchronize signals that come at different times (ex. Time-of-flight measurements). Can vary between 0.032 and 24.528 us.

12.23 FASTTRIG_BACKLEN

Scope: channel | **Group:** coincidence | **Unit:** microsecond | **Tag:** none

Used to set the width of the trigger signal generated by the channel to manage coincidence between channels. Takes a valid range of 0.008 to 32.76 us.

12.24 GATE_LENGTH

Scope: controller | **Group:** coincidence | **Unit:** fippi_clock_ticks | **Tag:** advanced

The time that channels 6 or 7 can gate other channels.

12.25 GROUPMODE_CH

Scope: channel | **Group:** coincidence | **Unit:** none | **Tag:** gate, trigger, advanced

Selects the source of the distributed trigger if Channel CSRA Bit 0 is enabled (=1). Can take the following values:

- 0-> OR of local triggers
- 1 -> OR of local triggers plus first FPGA
- 2 -> External trigger provided by the VETO MMCX connection on the front panel.
- 3 -> Coincidence of 2 channel pairs

See advanced triggering for more details.

12.26 GROUPMODE_FIP

Scope: controller | **Group:** coincidence | **Unit:** none | **Tag:** advanced

Determines what system logic gets used to generate the output signal on the TRIG_OUT MMCX connection. Can take the following values:

- 0 -> OR of local triggers
- 1 -> OR of stretched local triggers (HITs)
- 4 -> Coincident pairs of channels (OR of 4 HITs)
- 5 -> OR of energy in range test

See advanced triggering for more details.

12.27 MODULE_CSRA

Scope: controller | **Group:** control | **Unit:** none | **Tag:** veto, coincidence

Controls various aspects of MMCX front-panel input behavior.

Bit name	Bit	Description
MCSRA_FPTTCL	3	If enabled (=1), then the VETO input starts the TTCL acceptance window.
MCSRA_FP_COUNT	4	If disabled (=0), then use the local clock as the external timestamp in the list-mode data output. If enabled (=1), then use the MMCX CLK IN input edge count as the external timestamp.
MCSRA_FP_VETO	5	If enabled (=1), then enable the MMCX VETO input as global veto.
MCSRA_FP_EXTCLR	6	If enabled (=1), then enable the MMCX VETO input as external timestamp clear.
MCSRA_FP_PEDGE	7	If enabled (=1), then invert the active edge for the front panel pulse counter and counter clear.

12.28 PSA_THRESHOLD

Scope: channel | **Group:** data.psa | **Unit:** adc_steps | **Tag:** psa, list-mode

The amount an ADC sample must be above the baseline average before the PSA sum accumulation will start. Can vary between 1 and 16383. The effective upper limit is set by subtracting the ADC baseline value set with VOFFSET from the ADC bit resolution.

12.29 QDC_DEL

Scope: channel | **Group:** data | **Unit:** fippi_clock_ticks | **Tag:** qdc, psa, list-mode

An array of values that changes the start of the associated PSA summing window with respect to the fast trigger position when CHANNEL_CSRA Bit 9 is disabled (=0). Can vary between -2 and 500 8 ns samples.

12.30 QDC_DIV

Scope: channel | **Group:** data.psa | **Unit:** none | **Tag:** psa, list-mode

A scaling factor applied to the PSA sums that can be used to keep large sums within range. This value is only applied when CHANNEL_CSRA Bit 9 is disabled (=0). Can take a value of 4 or 32.

12.31 QDC_LEN

Scope: channel | **Group:** data | **Unit:** fippi_clock_ticks | **Tag:** qdc, psa, list-mode

An array of values that sets the consecutive summing window widths, which start from the beginning of the captured waveform. They can vary between 2 and 32766 FIPPI clock cycles. If CHANNEL_CSRA Bit 9 is disabled (=0) then the system uses the first two values for PSA. The valid range then becomes 2 - 60 FIPPI clock cycles.

12.32 REQ_RUNTIME

Scope: controller | **Group:** run | **Unit:** second | **Tag:** mca, list-mode

Requested run time in seconds for data runs.

12.33 SLOW_FILTER_RANGE

Scope: controller | **Group:** filter.energy | **Unit:** none | **Tag:** mca, list-mode

Specifies how many ADC samples get averaged before sending the digitized waveform to the energy filter. This can be used to extend the energy filter lengths for long signals. Can vary between 1 and 6. See the energy filter section for more details.

12.34 SYNC_AT_START

Scope: controller | **Group:** clock | **Unit:** none | **Tag:** mca, list-mode, white-rabbit

If disabled (=0), then the FPGA timestamps will not reset between subsequent data runs. If enabled (=1), then the FPGA timestamps reset to zero at the start of the data run.

12.35 TAU

Scope: channel | **Group:** filter.energy | **Unit:** microsecond | **Tag:** none

The exponential decay constant of the incoming signal. Used as part of the energy calculation to correct for the fact that signals may be sitting on the exponential tail of a previous signal. Can range from 0.0 to 65535.0 us.

12.36 TRACE_DELAY

Scope: channel | **Group:** data.list-mode | **Unit:** microsecond | **Tag:** data, list-mode

The number of pre-trigger samples captured in the list-mode trace. Can vary between 0 and 4.092 us in 0.008 us steps.

12.37 TRACE_LENGTH

Scope: channel | **Group:** data.list-mode | **Unit:** microsecond | **Tag:** data, list-mode

Sets the length of waveforms captured and exported as part of the list-mode data output. This value can be between 0 and 2.688 us in 0.128 us steps.

12.38 TRIGGER_FLATTOP

Scope: channel | **Group:** filter.trigger | **Unit:** microsecond | **Tag:** mca, list-mode

The flattop for the trapezoidal trigger filter. This value may range from 0.000 to 1.000 us.

12.39 TRIGGER_RISETIME

Scope: channel | **Group:** filter.trigger | **Unit:** microsecond | **Tag:** mca, list-mode

The amount of time the trapezoidal trigger filter takes to reach the flattop and return to baseline after the flattop. This value may range from 0.016 to 1.016 us.

12.40 TRIGGER_THRESHOLD

Scope: channel | **Group:** filter.trigger | **Unit:** none | **Tag:** mca, list-mode

The threshold that the trapezoidal trigger filter value must reach for the system to issue a trigger. Can range from 0.0 to 4096.0.

12.41 UDP_PAUSE

Scope: controller | **Group:** run | **Unit:** 64 * nanosecond | **Tag:** list-mode

Sets the minimum delay between UDP packages output on SFP 0 and SFP 1 ports. Can vary between 0 - 65535 in units of 64 ns.

If the on-board memory fills up faster than the packages are sent out, then run acquisition and statistics will pause. This scenario increases the system deadtime and leads to possible data loss.

12.42 VETO_MODE

Scope: controller | **Group:** coincidence | **Unit:** none | **Tag:** advanced

Configures the gating options used to VETO signals in the system. Can take the following values.

- 0 -> Use the VETO input from the front panel.
- 2 -> Use a stretched trigger generated by an AND between channels 6 and 7
- 3 -> Use the inverted stretched trigger generated by an AND between channels 6 and 7
- 4 -> Use a stretched trigger generated by an OR between channels 6 and 7
- 5 -> Use the inverted stretched trigger generated by an OR between channels 6 and 7

See the gating section for more details.

12.43 VETO_STRETCH

Scope: channel | **Group:** coincidence | **Unit:** second | **Tag:** none

Sets the length of the logic signal in microseconds output when the energy is between ELO and EHI. Can vary between 0.008 and 32.76 us.

12.44 VOFFSET

Scope: channel | **Group:** afe | **Unit:** volt | **Tag:** adc

Used to compensate for detector baseline offsets before being digitized. Can take a value of -1.25 V to 1.25 V.

12.45 XDT

Scope: channel | **Group:** data.waveform | **Unit:** microsecond | **Tag:** none

Sets the sampling period in microseconds between successive samples in the diagnostic ADC traces. It varies between 0.008 and 256 us.

13 Appendix D - Analog Signal Pin Diagrams