Pixie-4 Express

Pulse Shape Analysis Functions

Version 4.50

September 12, 2018

Hardware Revision: B

Software Revision: 4.50

XIA LLC
31057 Genstar Rd
Hayward, CA 94544 USA
Email: support@xia.com
Tel: (510) 401-5760; Fax: (510) 401-5761
http://www.xia.com/

Table of Contents

1 Overview 3
2 Installation Files 4
3 PSA Definitions 5
3.1 Algorithms 5
3.1.1 Pulse Shape Analysis, PSA 5
3.1.2 Fast Constant Fraction Discrimination, CFD 6
3.1.3 Slow Constant Fraction Discrimination, Slow CFD 7
3.2 Input Parameters 8
3.2.1 Pulse Shape Analysis 8
3.2.2 Fast Constant Fraction Discrimination 9
3.2.3 Slow Constant Fraction Discrimination 9
3.3 Return Values 9
4 Pixie Viewer Tools 11
4.1 PSA Tools 11
4.2 CFD Tools 16
5 Coding information 19
6 Multi-File Data Acquisition 20

1 Overview

A variety of radiation detectors have the ability to discern between different types of radiation, such as neutrons and gamma rays, by creating different pulse shapes for different types of interactions. In addition, phoswich detectors, consisting of multiple layers of different scintillators, produce different pulse shapes depending on which layer absorbs the radiation. A suitable pulse shape analysis (PSA) can detect such differences, which then can be used for particle identification.

While initially digital readout electronics has been used to simply capture detector waveforms for offline analysis, the on-board processing capabilities of the Pixie-4 Express allow PSA to be performed online in the digital signal processor (DSP) and Field Programmable Gate Array (FPGA). The main benefit is that this allows much higher throughput rates since full waveforms do not have to be transferred from the electronics to hard drive or other storage – only a few result values have to be transferred per pulse. This manual describes a number of functions available on the Pixie-4 Express (and Pixie-500 Express) with the "User PSA" extension. The functions are implemented in the FPGA for those operations that require simple, but time consuming operations, and in the DSP for those with more advanced operations. The DSP functions are also available as source code for compilation into the main code for those users who need to customize the output values.

The standard Pixie Viewer interface is extended by a User code plug in to specify input parameters and view results.

2 Installation Files

The PSA code variant is an addition to the Pixie Viewer. The release package includes several files which should be copied into the installation directories as follows:

- Pixie_PSA.pxp — copy into the main Pixie Viewer directory, e.g. C:\XIA\Pixie4e
- User_PSA.ipf — copy into the main Pixie Viewer directory, e.g. C:\XIA\Pixie4e
- Time_Analysis.ipf — copy into the main Pixie Viewer directory, e.g. C:\XIA\Pixie4e
- *.set — copy into the “Configuration” subdirectory
- *.pdf — copy into the “Doc” subdirectory, then please read.

Start the Pixie_PSA.pxp program in the same way as the standard version. Then execute the macro “Pixie_UseHomePaths” from the top menu bar **before** booting the Pixie module to ensure the correct files are in use. The .set files provide examples for a variety of applications.

3 PSA Definitions

3.1 Algorithms

3.1.1 Pulse Shape Analysis, PSA

The approach used here is a digital version of the Charge Comparison Method, where two sums over characteristic regions of the pulse are accumulated and a suitable ratio expresses the difference in pulse shape. This is a well established method used in a variety of applications, well suited for online processing due to its simplicity. In this case, the functions compute baseline average sum *B*, amplitude *A*, and two sums *QDC0* and *QDC1* over characteristic areas of the pulse, as shown in Fig.1. The first 8 samples of the waveform are summed and normalized to obtain *B*. The maximum sample M in the waveform is located, then subtracted by *B* to obtain *A*. Four input parameters (L0, L1, S0, S1), specified prior to the data acquisition, define the length and delay of the sums *QDC0* and *QDC1* relative to the trigger T, as shown in Fig, 1. The sums *QDC0* and *QDC1* are baseline subtracted by *B* (scaled according to L0 and L1, respectively). Another return value, or PSA ratio *R* = *QDC1/QDC0* is computed as the final result to differentiate pulse types. (Other ratios or combinations can be implemented on request). These data, plus the timestamp *TrigTime* and overall pulse height *E* computed with a trapezoidal filter, are written into the list mode data stream, followed by optional storage of the full waveform for each event.

The Pixie-4e also accumulates online 2D histograms of *R* vs energy in external memory, applies a discrimination threshold T_{ng}, and reports counts above and below threshold.

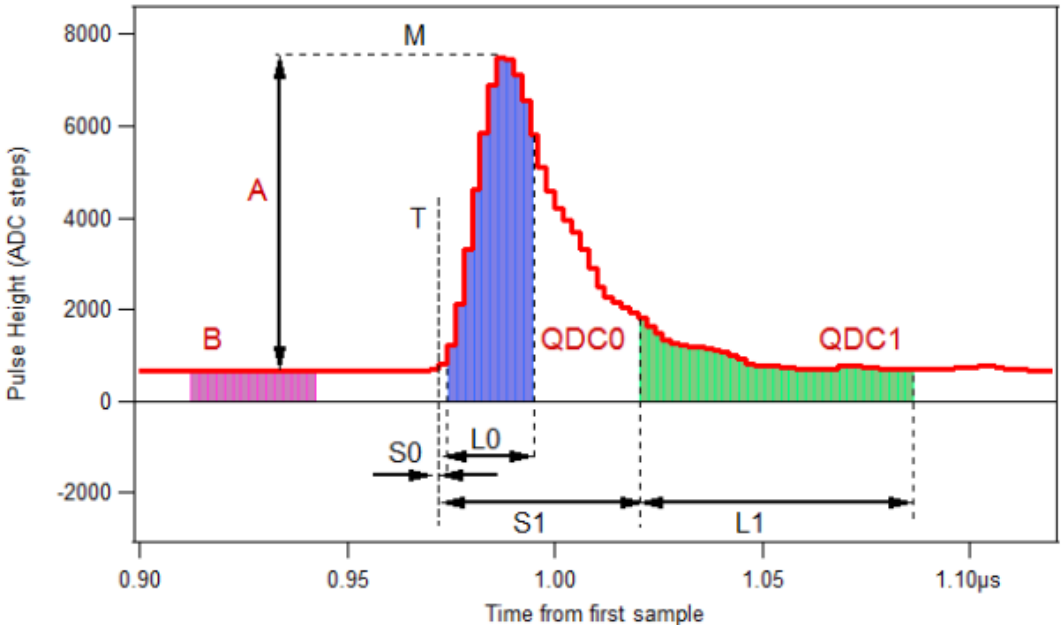


Figure 1: The PSA quantities and input parameters

3.1.2 Fast Constant Fraction Discrimination, CFD

An additional function of the PSA firmware is to compute a constant fraction timing value for the rising edge. This is illustrated in Fig.2. Having found the maximum of the pulse, the firmware logic computes the CFD level as 50% of the amplitude, and finds the two closest points to that level, cfdlow and cfdhigh. The CFD time of arrival x is then computed by the DSP through interpolation between cfdhigh and cfdlow:

x /1 = (cfdhigh – cfdlow) / (CFD level – cfdlow)

In the Pixie-4e, time stamps are latched by local or distributed triggers issued at the rising edge of a pulse. To accommodate delayed pulses in coincidence systems, the *CFD time* is captured at the end of the user specified coincidence window (1/2 of the "Window width" specified in the Pixie Viewer). The reported *CFD time* value is the time from the interpolated CFD level crossing to the end of the coincidence window. Therefore one can compute [in units of 2ns sampling intervals] the fractional (subsampling) time of arrival x from cfdlow

x = 1 – frac((*CFD time*)/256)

and the absolute time of the CFD crossing is

CFD crossing = *TrigTime* – (*CFD time*)/256+ ½ Window width

For example, for a user specified coincidence Window width of 400ns, the Pixie module may report an event with a *TrigTime* of 1234 and a *CFD time* of 12832. Then one can compute

x = 1-frac(50.125) = 0.875 (or 1.75ns)

and

CFD crossing = 1234 [x 2ns] – (12832/256) [x 2ns]+ ½ x 400ns
= 2567.75 ns after last reset of the time stamp counter

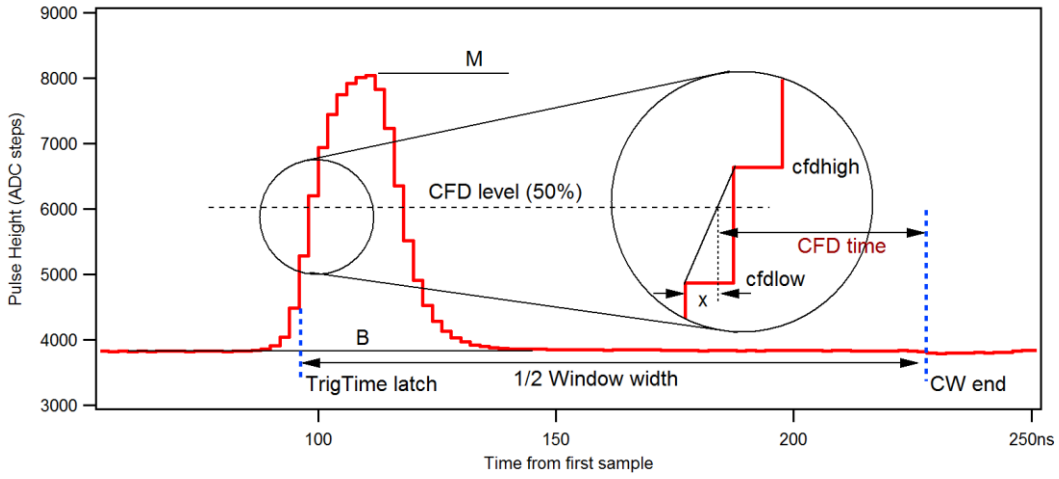


Figure 2: CFD timing definitions

Restrictions for the CFD:

- The CFD level is fixed to 50%
- The rising edge, from 50% level to maximum, must be less than 12 samples (24ns in the 500 MHz variant, 96ns in the 125 ns variant)

- If more than one pulse occurs within the coincidence window, results are invalid
- To measure time-of-arrival differences between channels, all channels must be in group trigger mode so data is captured on the same trigger.
- As *CFD time* has a maximum value of 512 ns, the coincidence Window width must be less than 1024ns

3.1.3 Slow Constant Fraction Discrimination, Slow CFD

For the 16/125 variant of the Pixie-4e, an additional slow CFD algorithm is implemented in the firmware. Assume the digitized waveform can be represented by data series *Trace[i]*, $i = 0, 1, 2, \ldots$ First the fast filter response (FF) of the digitized waveform is computed as follows:

$$FF[i] = \sum_{j=i-(FL-1)}^{i} Trace[j] - \sum_{j=i-(2*FL+FG-1)}^{i-(FL+FG)} Trace[j]$$

Where *FL* is called the fast length and *FG* is called the fast gap of the digital trapezoidal filter. Then the *CFD* is computed as follows:

$$CFD[i+D] = FF[i+D]*(1-w/8) - FF[i]$$

Where *D* is called the CFD delay length and *w* is called the CFD scaling factor (w=0, 1, …, 7).

The CFD zero crossing point (ZCP) is then determined when $CFD[i] \geq 0$ and $CFD[i+1] < 0$. The timestamp is latched at Trace point *i*, and the fraction time *f* is given by the ratio of the two CFD response amplitudes right before and after the ZCP.

$$f = \frac{CFDout1}{CFDout1 - CFDout2}$$

where *CFDout1* is the CFD response amplitude right before the ZCP, and *CFDout2* is the CFD response amplitude right after the ZCP (subtraction is used in the denominator since *CFDout2* is negative).

The Pixie-4e DSP computes the CFD value as follows and stores it in the output data stream for online or offline analysis.

$$CFD = \frac{CFDout1}{CFDout1 - CFDout2} \times N$$

where *N* is scaling factor, which equals to 256. With f in the range $0 \leq f \leq 1$, the DSP thus reports the zero crossing with a precision of 1/256 clock cycle (31.25ps) as an 8 bit integer number. In addition the DSP reports 8 bits of the real time clock (RTC) at the time of zero crossing, so that the final CFD value reported is

$$TS.CFD = 256*RTC[7:0] + CFD$$

The time unit of TS.CFD is in 1/256 clock cycle (31.25ps), i.e. divide by 256 to scale in clock cycles.

Time differences between close records of two channels can be computed as

$$\Delta T = (TS.CFD_1 - TS.CFD_2) * 0.03125 \text{ [in ns]}.$$

For events that are separated by more than 256 clock cycles, use the 56 bit event time stamps ($ET_{1,2}$) to compute a coarse time difference in clock cycles, then correct for the difference in the 8 bits of TS_i and the lower 8 bits of ET_i, then add the difference of (CFD_1-CFD_2)/256 for the total result in clock cycles, i.e.

$$\Delta T = (ET_1 - ET_2) - (ET_1[7:0] - TS_1) + (ET_2[7:0] - TS_2) + (CFD_1 - CFD_2)/256$$
[in 8ns clock cycles]

Make sure to correctly handle potential overflows in the lower 8 bits of ET vs TS. The raw values of CFDOUT and a 32 bit value of RTC are also reported in the list mode data file in case more precise results are required.

3.2 Input Parameters

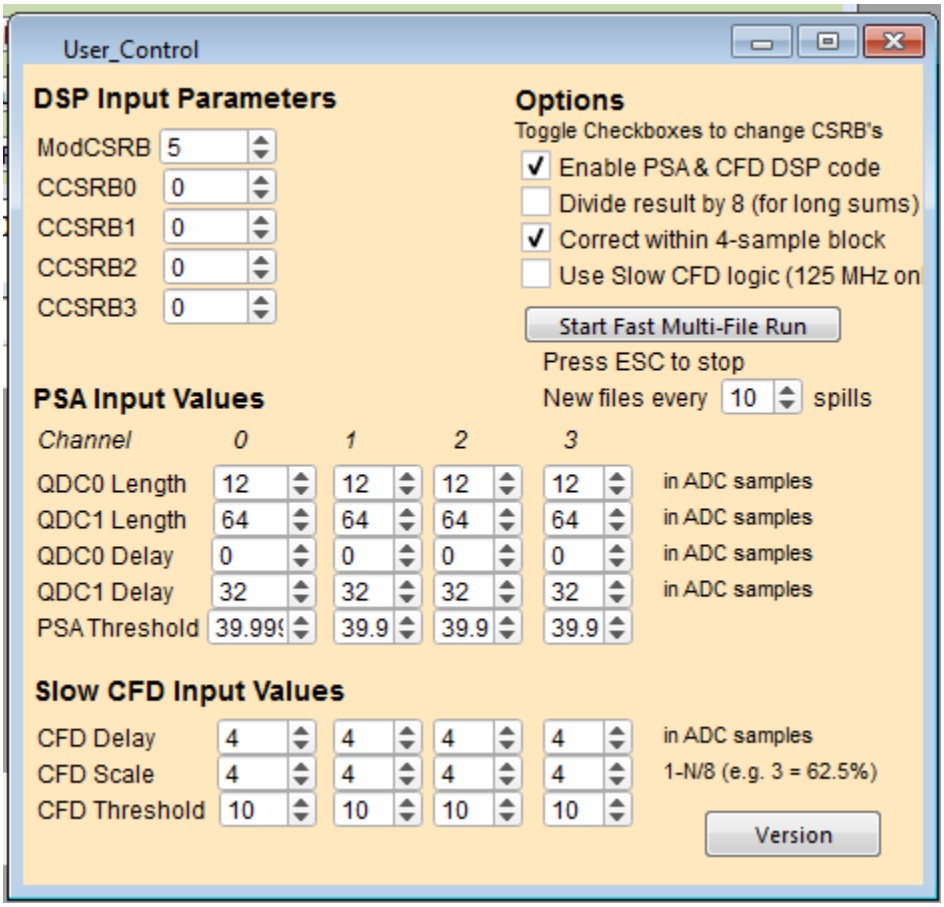


Figure 3: Pixie Viewer User_Control panel for PSA input parameters

3.2.1 Pulse Shape Analysis

PSA input parameters are specified in the User_Control panel (top menu User PSA → User Control Panel) In the upper left corner of the panel are control bits that activate the PSA code and control its processing options. The ModCRSB control affects a module as a whole, the CCSRB bits control for each channel individually a number of processing options. Usually it is not necessary to change the CSRB values directly, instead toggle the checkboxes until the desired option is set. Current options are

- Enable PSA & CFD DSP code
  Check box to perform the PSA computations in the DSP. Uncheck to speed up the processing for higher throughput when no PSA is needed.

- Divide result by 8
  Check box to enable this division for long sums, where the result of QDC0 or QDC1 may overflow its 16-bit variable
- Correct within 4-sample block
  Check box to refine the PSA sum location within a block of 4 samples. Should always be on.

In the center region, for each channel the length (L0, L1) and delay (S0, S1) of the sums is specified for each channel, as well as a threshold to detect the rising edge of a pulse. The threshold applies to both sums and is also used for the CFD maximum detection. The sum length and delay can be set individually, with the following limitations:

- QDC1 must finish last (i.e. S1+L1 > S0+L0)
- L0, L1 must be between 4 and 120, and a multiple of 4
- L0+S0, L1+S1 must be between 0 and 380
- The difference of S0 and S1 must be a multiple of 4 (e.g S0 = 1, S1 = 17)

The values shown in Fig.3 are good general purpose starting points. S0 is usually close to zero. Note that the checkboxes do not always reflect the state of the CCSRB variables shown. Toggle checkboxes at least twice to change the CCSRB bit, or enter the CCSRB value directly as a hexadecimal number (e.g. “0x14” for hex 14 = decimal 20).

For further DSP parameters for online 2D MCA spectra, please see below.

3.2.2 Fast Constant Fraction Discrimination

There are no input parameters for the fast CFD logic. The CFD level is fixed to 50%.

The following checkbox must be **cleared** to use the fast CFD

- Use Slow CFD logic (125 MHz only)

3.2.3 Slow Constant Fraction Discrimination

The input parameters for the slow CFD logic are grouped at the bottom of the User_Control Panel. Specify the delay D, the scaling factor w and the threshold as described above.

The following checkbox must be **checked** to use the slow CFD

- Use Slow CFD logic (125 MHz only)

3.3 Return Values

The DSP writes the PSA return values to the list mode data stream. The PSA values are placed into words 10-19 of the channel header as described in the Pixie-4 Express User Manual. Table 1 lists the mapping of the values to the channel header locations. To accommodate fractions for the ratio R, the number recorded by the DSP is 1000*R. To accommodate fractional samples for the CFD time, the number recorded by the DSP is 256* CFD time (1 LSB = 7.8125ps). Divide the reported value by 256 to match units with the Trigger time (2ns).

Word #	Variable	Description
0	EvtPattern	Hit pattern.
1	EvtInfo	Event status flags.
2	NumTraceBlks	Number of blocks of Trace data to follow the header
3	NumTraceBlksPrev	Number of blocks of Trace data in previous record (for parsing back)
4	TrigTimeLO	Trigger time, low word
5	TrigTimeMI	Trigger time, middle word
6	TrigTimeHI	Trigger time, high word
7	TrigTimeX	Trigger time, extra 8 bits
8	Energy	Pulse Height
9	ChanNo	Channel number
10	**User PSA Value**	**Amplitude A**
11	**XIA PSA Value**	**CFD time *256**
12	**Extended PSA Values**	**Base B**
13	**Extended PSA Values**	**QDC0**
14	**Extended PSA Values**	**QDC1**
15	**Extended PSA Values**	**Ratio R *1000**
16	**Slow CFD values**	**CFDout1**
17	**Slow CFD values**	**CFDout2**
18	**Slow CFD values**	**CFD time stamp, low word**
19	**Slow CFD values**	**CFD time stamp, high word**
20--32	reserved	

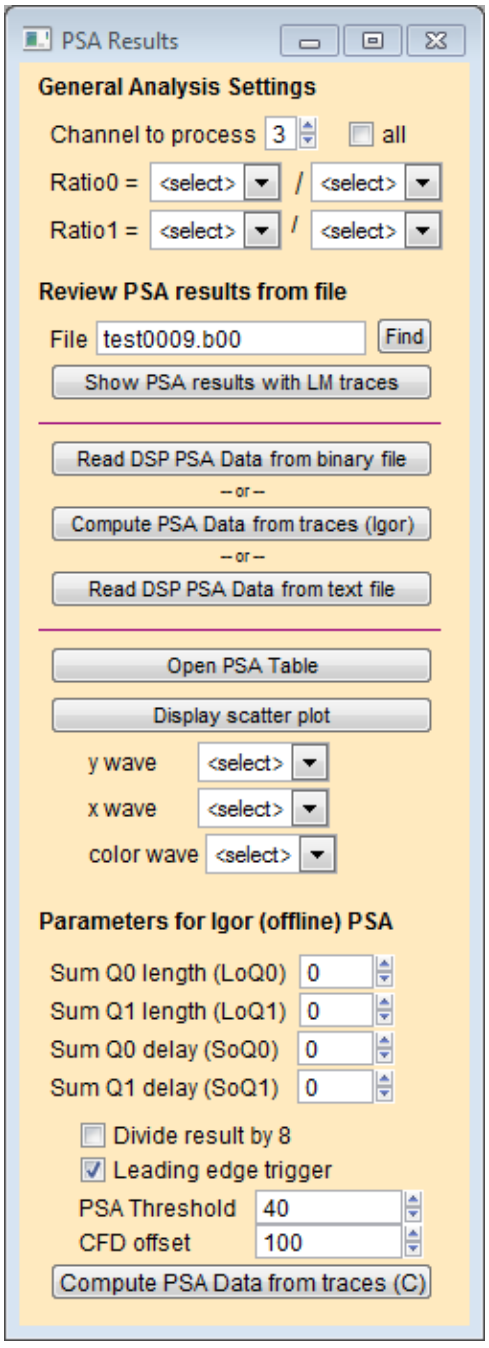
Table 1: Channel header data format.

For further return values related to online 2D MCA spectra, please see below..

4 Pixie Viewer Tools

4.1 Offline PSA Tools

To assist in the analysis of the acquired data, a number of processing and display functions have been added to the Pixie Viewer software. These functions can be accessed under top menu UserPSA → PSA Analysis, see left.



At the top of the panel, specify the Pixie channel for which the data will be analyzed. (Only one channel is active at a time, but if the "all" checkbox is checked, Igor processes 4 channels in sequence). The controls underneath allow to define ratios of any two PSA parameters.

The button "Show PSA results with LM traces" opens the standard list mode trace display and an additional panel where the PSA results are displayed. In three columns, the results from Igor, DSP, and FPGA computations are shown for comparison. Igor computes the results from captured waveforms (if present) using the parameters specified at the bottom third of the panel. This function is most useful to view one event at a time and debug or develop the PSA algorithms and settings.

The three buttons in the next section gives the options to either read all the PSA results computed by the DSP and/or FPGA from a binary data file, or read the data from a .dt3 text file, or process all waveforms in a file offline in Igor to compute the PSA results with the parameters specified at the bottom third of the panel. The latter can be rather time consuming, but allows offline optimization of PSA parameters on the exact same data. A faster version of the offline computation has been implemented in C. The button "Compute PSA Data from traces (C)" at the very bottom calls this C function with the parameters specified for Igor. The C function generates a

.dt3 file[1] which then can be loaded into Igor with the button "Read DSP Data from text file" (though it is not computed by the DSP, but by the C function).

The next section has buttons to present the PSA results for all events – as read from the file or as processed from waveforms, depending on the last action in the section above. The first button opens a simple table, the other creates a scatter plot in which the parameter specified in "y wave" is plotted against the parameter specified in "x wave" and the dots are colored by the parameter selected in "color wave". Standard Igor Pro functions can be used to change scale, size, and color ranges of the plot.

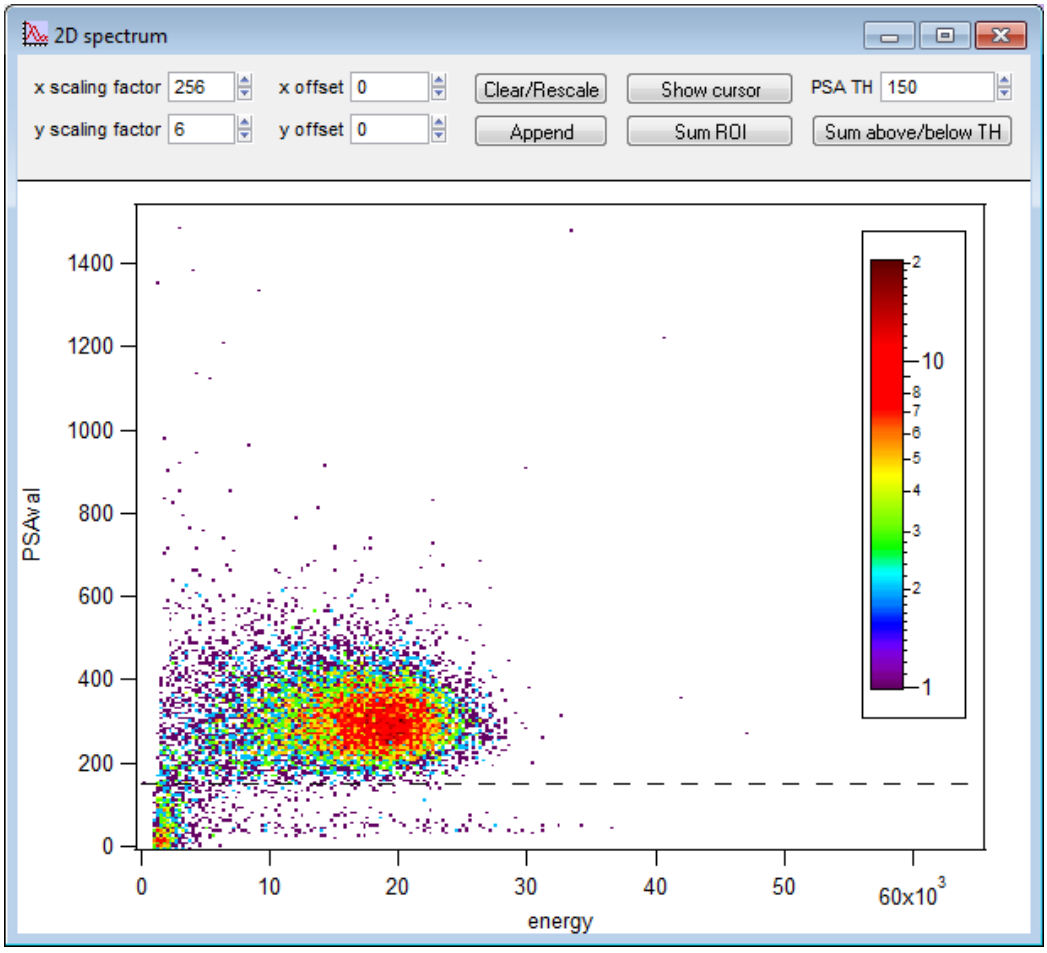


Figure 4: Offline 2D MCA from list mode data

In some cases, a scatter plot can become too dense to view clearly. Under top menu UserPSA → 2D MCA, a new window opens that accumulates the data from the scatter plot into a two-dimensional histogram (256x256 bins). PSA results are subtracted by "x [y] offset" and divided by "x [y] scaling factor" to bring them in range of the histogram. (For example, if energies range from 0 to 2048, set offset = 0 and scaling factor = 8 to show the full range, or set offset = 1024 and scaling factor = 2 to "zoom in" on the range 1024 to 1536.) Click "clear/rescale" after modifying scaling factor and offset, then "append" to add

[1] To generate a .dt3 file, the option Run Parameters > Record > Data Record Options > Auto process list mode data into .dat file type must be set to 3.

data from the last processed file. There are also buttons to show cursors and sum the region of interest (ROI) between them. Further controls allow to specify a threshold (horizontal dashed line) and sum the number of events above and below. Clicking the button “Sum above/below TH” prints the sum values in the history window and also computes MCA projections to the x and y axis. These projections can be displayed via top menu UserPSA → ROI MCAs

Data from multiple files can be added into the same 2D histogram by repeating the following sequence:

1. specify file in list mode trace display,
2. click “read DSP PSA data from file” and
3. click “append”.

The parameters for Igor processing specified in the PSA Analysis panel are equivalent to those in the User Control panel. The field “CFD offset” is used to adjust for effects of pre-trigger trace delay and coincidence window on the combination of time stamp and CFD refinement: The DSP computation gives the time relative to the end of the coincidence window, the Igor computation relative to the start of the waveform, and rather than carrying this around in the code, users can enter an empiric value to compensate the offset, valid for all events in the file to a precision of a few clock cycles. The “leading edge trigger” option should always be checked; else Igor will use the 10% level as the starting point for the PSA sums which may differ from the FPGA’s absolute leading edge threshold.

Notes:

- The fields to define “ratio” and the scatter plot “x/y/color waves” by default show “energy”, but the fields have to be toggled at least once to properly define the waves.
- Files with more than ~50,000 events may slow down Igor Pro's plotting performance, depending on the resources of the PC.

4.2 Online PSA Tools

Similar to the offline 2D MCA, the Pixie-4e also maintains an online 2D histogram of PSA value vs energy for each channel. These spectra are located in the additional 1MB memory space and are saved as a separate file with extension .mca2D.

The online 2D MCA has 256 x 256 bins per channel. Since energy and PSA values are computed as 16bit numbers (max 65535), the result will likely have to be divided by up to 256 to fit into the available bins. The 2D MCA memory is organized into 4 channel blocks with 256 x 256 bins of 4 bytes. The address A for the 32bit word corresponding to a given channel C, energy E and PSA value R is

$$A = C * (256x256) + R * 256 + E.$$

The online 2D MCA uses the following input parameters defined in the panel “2D PSA MCA from DSP”, opened from top menu User PSA > 2D MCA (DSP online), and shown in Figure 5:

- X scale
  Scaling factor for x axis (energy) of the 2D MCA. A number N entered will result in a division of energy by 2^N. Set to 8 to divide by 256 and thus display the full energy range.
- Y scale
  Scaling factor for y axis (PSA value) of the 2D MCA. A number N entered will

result in a division of energy by 2^N. Set to 8 to divide by 256 and thus display the full PSA value range.

- N/G threshold
  As in the offline spectrum, a threshold can be defined to sum events above and below (e.g. gammas and neutrons). This threshold value is applied after the division and thus ranges from 0 to 255.

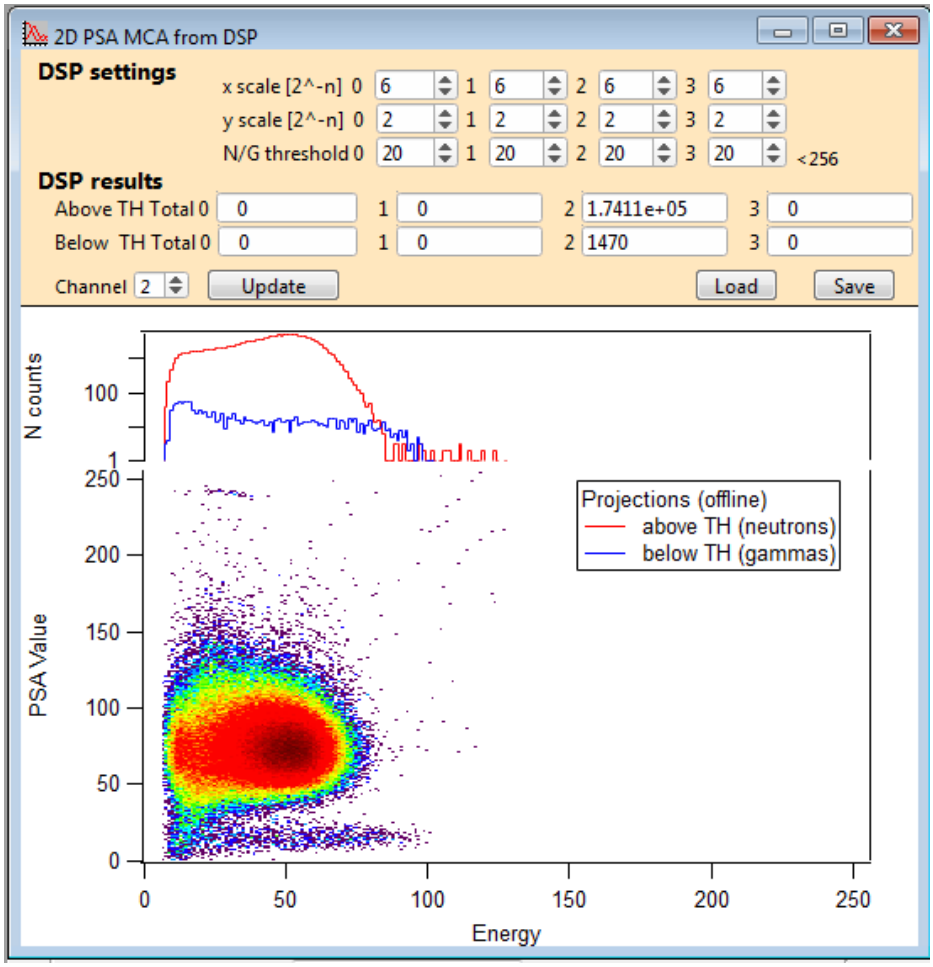


Figure 5: Online 2D MCA from Pixie-4e memory

The panel "2D PSA MCA from DSP" shows the 2D MCA in the lower section as an "Igor image" plot. Right click on the 2D MCA area to change appearances such as color palette, log color scale, etc. with the standard Igor functions. Use the "channel" control to switch between the input channels of the Pixie-4e.

In the plot area above the 2D MCA image are projections to the energy axis for points above and below the threshold. These projections are computed by Igor (offline), not the DSP, but refreshed for every update.

Below the DSP settings, the panel are also lists the results of the DSP's above/below threshold count. These numbers are reported in the .set and .ifm files in the UserOut parameters. Use (high) * 65536+ (low) to compute the total.

Location	Value	
UserOut +0	Above TH ch.0 (high)	
+1	Above TH ch.0 (low)	
+2	Above TH ch.1 (high)	
+3	Above TH ch.1 (low)	
+4	Above TH ch.2 (high)	
+5	Above TH ch.2 (low)	
+6	Above TH ch.3 (high)	
+7	Above TH ch.3 (low)	
+8	Below TH ch.0 (high)	
+9	Below TH ch.0 (low)	
+10	Below TH ch.1 (high)	
+11	Below TH ch.1 (low)	
+12	Below TH ch.2 (high)	
+13	Below TH ch.2 (low)	
+14	Below TH ch.3 (high)	
+15	Below TH ch.3 (low)	

4.3 CFD Tools

For timing analysis, go to top menu Timing → Time Panel. This opens the panel shown in Fig.6. Its purpose is to extract time differences ΔT between pulses "A" and "B" from different channels and modules, and accumulate ΔT timing histograms. The code and panel are open source for user modification. Therefore it is described here only briefly:

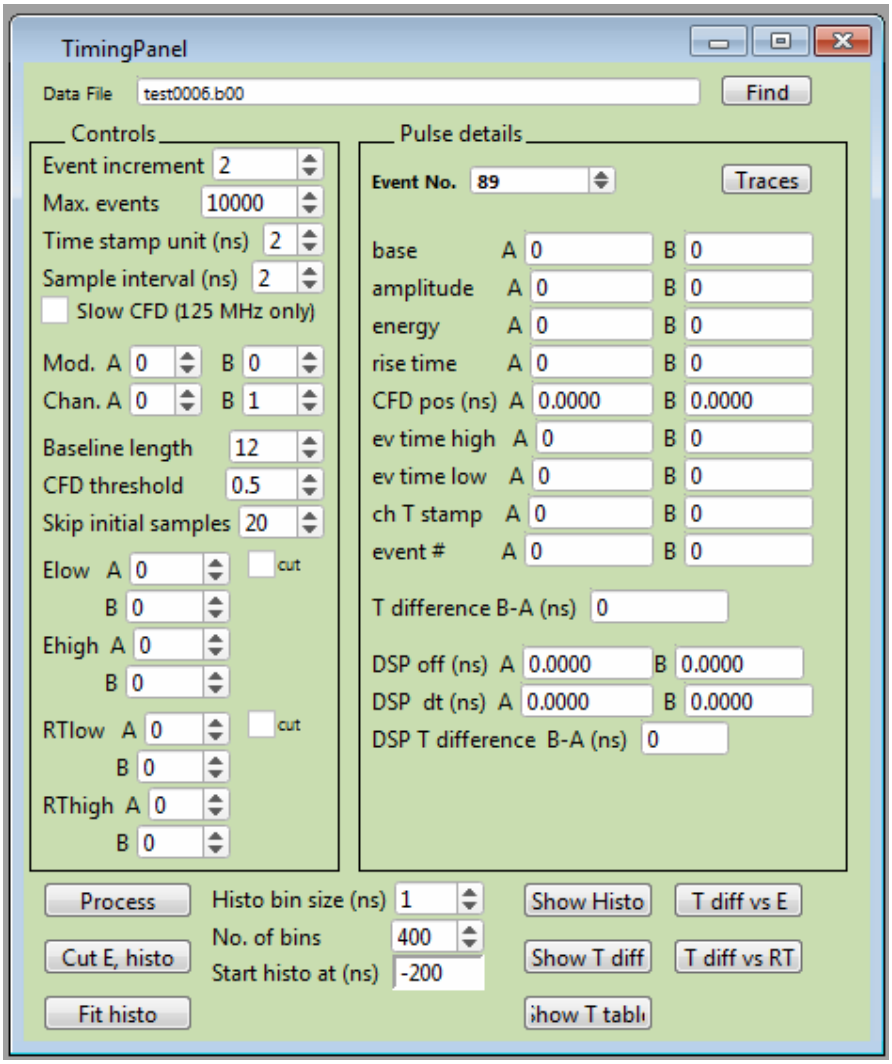


Figure 6: Timing Analysis panel

- The **Event increment** control should be set to 2 for acquisitions in run type 0x400 with 2 active channels per module, 3 for 3 active channels, and so on.
  In run type 0x402, it should be set to 1 no matter how many channels are active. However, in run type 0x402 the DSP computed CFD values are not reported in the file.
  Note: Verify the "Show 4 pulses…" option is enabled in the list mode trace display for analyzing 0x400 files with more than one active channel per module.
- The **Max events** control is used to process only a portion of the file (up to the chosen event number). This is useful for tweaking parameters with larger files.

- The controls **Time stamp unit** and **Sample interval** specify the time scale for the time stamps and waveforms, respectively. They are user controlled variables to accommodate data taken with different devices. Defaults are 2ns/2ns for the 14bit, 500 MHz variant of the Pixie-4e.
  For the 16bit, 125 MHz variant of the Pixie-4e, choose 2ns/8ns. Toggle the check box for slow/standard CFD to display the DSP CFD results appropriately.
  For the 12bit, 250 MHz variant of the Pixie-Net, choose 1ns/4ns.
- Below that are parameters for the Igor computation:
  - **Baseline length** (how many points to use at the beginning of the trace to define the DC level),
  - **CFD threshold** (0.5=50% of amplitude), and a
  - **Default trigger position** (the earliest point the code starts looking for a rising edge).

  These values should default to 12, 0.5, 20
- Below the Igor parameters are thresholds to apply energy and rise time cuts before histogramming, for example to focus only on pulses with 1.3 MeV. The cuts can be enabled or disabled by the checkboxes. Both cut and uncut values are reported. Ignore at first.
- On the top and the right side, you can choose file and event number, similar to the list mode trace display. There will probably be an error about a null or missing wave the first time you change event number, which can be ignored.
- The [**traces**] button opens a window with the 2 waveforms of the current event.
- Below that are a number of results, either computed by Igor from the waveforms or DSP results read from the file:
  - [Igor] base = the average of the first N samples of the trace
  - [Igor] amplitude = max pulse height found
  - [DSP] energy = energy computed by P4
  - [Igor] rise time = pulse rise time
  - [Igor] CFD pos = time from beginning of trace to CFD level
  - [DSP] ev time high/low = event time stamps from file
  - [DSP] ch time stamp = channel time stamp
  - event # = should be same as the one selected above
  - [Igor] T difference ΔT = difference of CFD time and channel time stamps
  - [DSP] DSP off = offset from time stamp to CFD time
  - [DSP] DSP dt = fractional CFD time as described above as x
  - [DSP] DSP T difference ΔT = difference of CFD time and channel time stamps
- **Operation**
  Typically one would initially increment the event number to investigate events pulse by pulse.
  Clicking the [**process]** button at the bottom runs over all events in the file, computes the time differences ΔT, applies any rise time or energy cuts, and histograms them into the ΔT spectrum with the histogramming parameters defined to the right. The histogram range is symmetric around 0.
  - [**Cut E, histo]** reapplies any cuts and repeats the histogramming.

- [**Show Histo]** opens a plot of the ΔT spectrum
- [**Show Tdiff]** opens a plot of the ΔT plotted vs event number.
- [**Show T table]** opens a table of the values computed or read from the file.
- **[Tdiff vs E]** and [**Tdiff vs RT]** open scatter plots of these properties (one for each channel) so one can look at any dependency and figure out if/where to cut.
- **[Fit histo]** fits the ΔT histogram with a Gaussian (use cursors to define fit range).

5 Coding information

In the current firmware, the FPGA computes B, M, and the raw sums QDC0 and QDC1. The DSP subtracts B (properly scaled for length) from M, QDC0 and QDC1 to obtain the amplitude A, and the final values for QDC0 and QDC1. The QDCs are then scaled to fit into the 16 bit return value. The DSP also computes the ratio R.

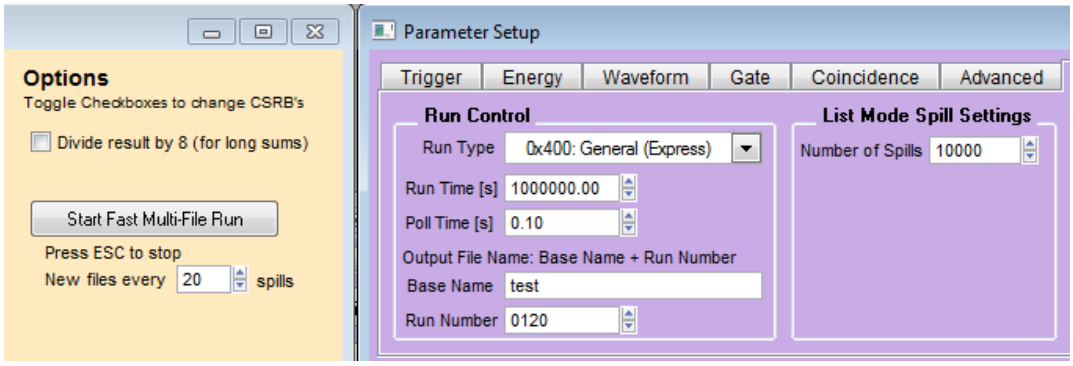
These computations are performed in a single subroutine in the DSP code (user.asm). Interested users can obtain a precompiled version of the DSP code from XIA and the source code for user.asm. They can modify the user subroutine to better match their application and rebuild the complete DSP code. (Analog Devices Visual DSP required)

Alternatively, XIA is available to implement custom ratios or quantities upon request.

6 Multi-File Data Acquisition

For long data acquisition runs, data files easily can exceed a couple of GB in size. This is problematic not only because such files are unwieldy (e.g. some file repositories can only handle files < 2GB), the huge number of events makes it also extremely slow and cumbersome to process and display such files in Igor Pro. Therefore it is recommended to break up long acquisition runs into multiple files (see User Manual). Specifically in this code variant, a "fast" multi-file acquisition routine has been added that skips some of the complexities of the general code, and uses a direct loop polling the module rather than an Igor background task. The direct loop is executed faster, but no other Igor functions can be executed while it is in progress – essentially Igor freezes until the requested acquisition time or number of spills is reached. However, Igor does monitor the ESC key, which can be pressed to abort the run.

In this mode, the first data file is created with the base name and run number specified in the run control tab. Subsequent files are created when the specified number of spills is reached by incrementing the run number. At the end of the acquisition, statistics and settings are save to files with the first run number.



For example, in the screenshot above, clicking the "Start Fast Multi-File Run" button asks for 1000 spills broken up into files of 20 spills. This creates (in this order)

- data file test0120.b00*, containing 20+ spills
- data file test0121.b00*, containing 20+ spills
- …
- data file test0169.b00*, containing 20+ spills
- settings file test0120.set, containing settings (including raw run statistics for the whole acquisition)
- statistics file test0120.ifm, containing statistics for the whole acquisition and settings
- spectrum file test0120.mca, containing spectra for the whole acquisition

* .dt3 text file in run type 0x401.

1.