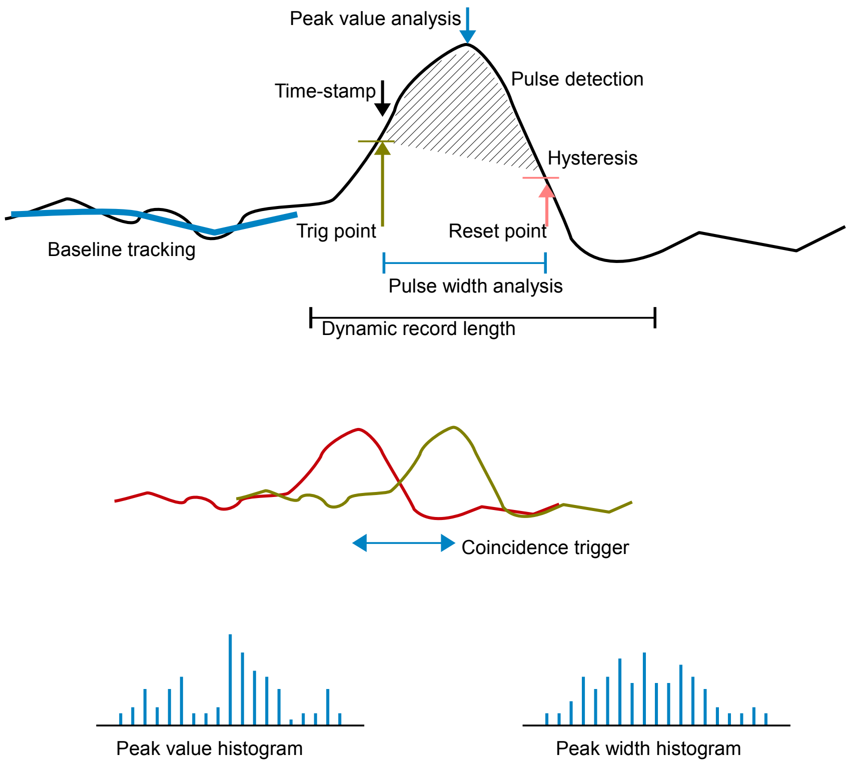
The art of pulse detection

Application note



Author(s): Teledyne SP Devices
Document ID: 18-2118
Classification: Public
Revision: A
Print date: 2018-08-07

Contents

1 Introduction

The core challenge in pulse detection is to design a system for *random* events. The Teledyne SP Devices's firmware option –FWPD, for the ADQ14 and ADQ7, is tailored to detected and analyze pulses in demanding measurement applications where the properties of the pulses (such as timing or length) are not known in advance. With high sampling rate, high vertical resolution, and sophisticated signal processing methods, regions of interest are distinguished and collected. Since only the regions of interest are collected, and all other data is suppressed, the magnitude of data reduction is significant. That in turn, will relax the requirements on the host interface and storing capacity.

The purpose of the firmware option –FWPD is to detect pulses and adapt the data capture to the properties of the pulses. The firmware consist of serveral parts, as illustrated in Fig. 1.

- Digital baseline stabilizer (**DBS**) and moving average (**MA**) filter contribute to a stable baseline and reduced pattern noise (Sec. 2.2.1 2.2.2).
- The sophisticated **Peak detection** offers dynamic recording (Sec. 2), zero suppression, and a dynamic level trigger (Sec. 1.1 and 2.3). It also includes additional timing rules for when pulses should be accepted (Sec. 3).
- The **Peak analysis** in the FPGA allows for real-time characterization of the identified pulse (Sec. 4).
- Several modes of **Data collection** is offered to transfer data to the host (Sec. 5).
- **Latency control** is enabled with optional padding of zeros, to guarantee a minimum throughput (Sec. 6).
- **Timestamp** allows for real time timing of an event (Sec. 5.4).

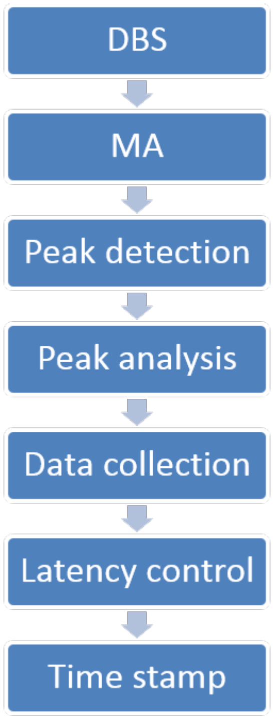


Figure 1: Flow graph of –FWPD.

The peak detection tracks the baseline and identifies when pulses start and end. The recording includes dynamic record length which suppresses zeros, that is, signal without information is discarded and disk space is saved. Coincidence trigger detects simultaneous events on several channels. The pulse analysis finds the peak value and the width of a pulse. There is also built in histogram functions for pulse peak value and pulse width. Calculations are done in the FPGA and in real-time to improve performance and reduce processing load on host CPU.

1.1 Pulse detection for dynamic recording and zero suppression

The main difference to the standard firmware FWDAQ is shown with an example for ADQ14-4C in Figure 2. In the standard ADQ firmware (FWDAQ) the records are synchronized in *all* channels and the records are of *equal* size. The firmware addressed in this application note (FWPD) adds *individual* triggering on all channels and data driven record length. A dynamic record length implies that only data of interest is recorded. This zero suppression[1] relaxes the required data rate and saves disk space.

The FWPD also offers a dynamic level trigger, in the meaning that the trigger level is measured with respect to a baseline of the signal. The measure of baseline tracks the signal and can follow fluctuations. By setting the timing between the baseline tracking and the trigger point, –FWPD implements an activity detector, that is, trigger on the condition $dy/dx > z$.

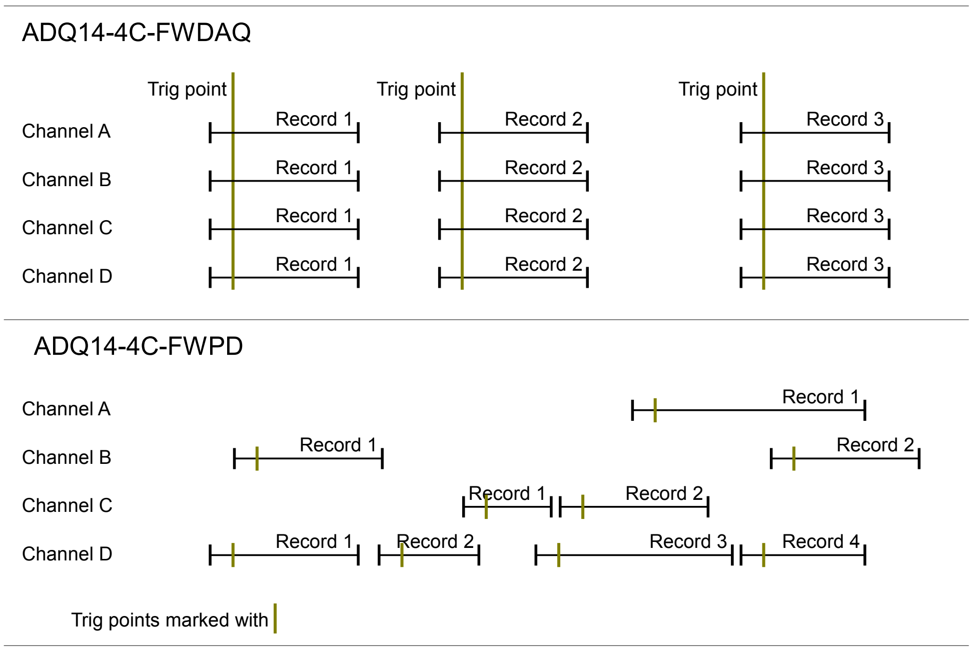


Figure 2: The –FWPD adds dynamic recording, which is illustrated with an ADQ14-4C example.

[1]Data between pulses are rejected.

2 How to find a pulse

The task to find an event in pulse detection applications is two-fold. The first step is to find a stable baseline, and the second step is to find the region of interest, where the action takes place. In this chapter, the basic theory on the subject is given. The main concepts are illustrated in Fig. 3, but before digging into the details (Sec. 2.2, 2.3), the DC-offset feature is worth mentioning (Sec. 2.1).

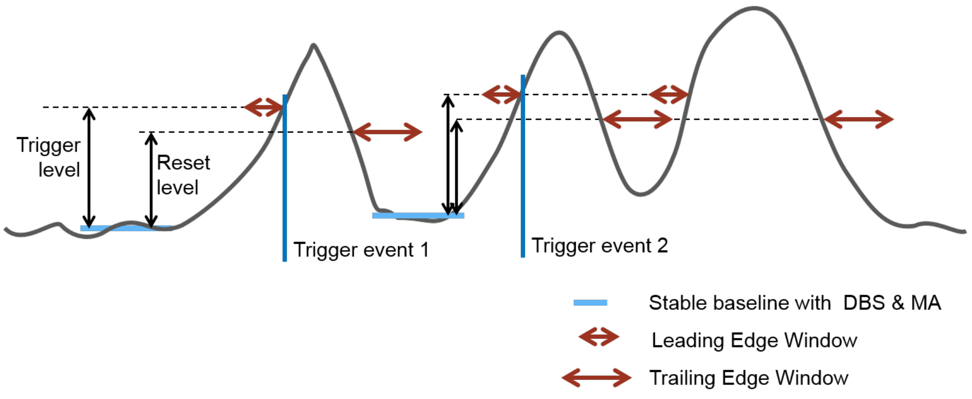


Figure 3: Illustration of the main concepts used to define a pulse. Each concept will be explained below.

2.1 The DC-offset feature

If the signal at hand is unipolar, an analog DC-offset can be added to the original signal to better exploit the input range of the ADCs. An example of such a signal is shown in Fig. 4. In the given example, the input signal is unipolar with negative pulses. The signal is built up by a baseline around zero volts and a signal pulse, Fig. 4 (a)[2]. To fully use the symmetrical input range of the ADC, an analog DC-offset is added to the signal, Fig. 4 (b). This will place the baseline of the signal near one rail in the signal range. The peaks can then cover the whole signal range, Fig. 4 (c), and the resolution is effectively doubled. The DC-offset level is controlled from software and the control range is rail to rail, but preferably a margin of about 10% should be left, to allow for overshoot. See Figs. 4 (d) and (e). The DC-offset compensation is one of several baseline operations, as illustrated in Fig. 5.

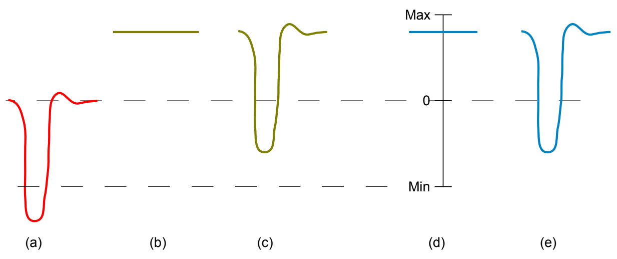


Figure 4: Analog input DC-offset adjustment with margin, to allow for overshoot.

[2]The DC-level of the signal does not have to be zero. The DC-coupled versions will let the DC level of the input signal through. If required, the AC-coupled AFE can be used for removing the input DC level.

2.2 Find the baseline

With large baseline variations, it becomes problematic to define what is a pulse and what is not. Aging, temperature variations, pattern noise, etc, will reduce the accuracy of the pulse detection and analysis, if not compensated for. Pulses could be missed, and false pulses might be detected. A stable baseline is therefore vital when detecting pulses. There are several techniques available to track and compensate for unwanted signal variations as well as suppression of pattern noise. A block diagram of the entire baseline system[3] is given in Sec. 5. The DBS (Sec. 2.2.1 and 2.2.2) is situated right after the ADC, followed by the MA filter operation (Sec. 2.2.3).

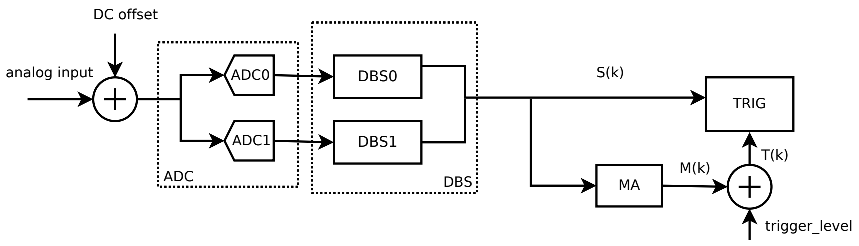


Figure 5: A block diagram of the baseline system. The DBS is situated right after the ADC, followed by the MA filter operation. Together they track both slow and fast baseline variations and suppress pattern noise from the interleaved ADCs.

2.2.1 Track slow baseline variations

Temperature and aging slowly change the behavior of electronics. Tracking and removing theses kind of variations from the signals enables the user to focus on the pulse relative to a fixed baseline. Thereby will the user-controlled trigger level settings be valid over time without any additional calibration procedure by the user. Teledyne SP Devices' proprietary IP, the *digital baseline stabilizer* (DBS), provides a highly accurate and stable baseline, specifically intended for tracking slow variations in pulse applications. The baseline is adjusted to a target value with 22-bit precision. The principle of operation is shown in Fig. 6.

The target value, `dc_target`, is set by the user. The DBS is a blind identification process which operates in the background. Since DBS is always active, it will constantly monitor and follow variations in the baseline and correct for time-invariant behavior, as illustrated in Fig. 7. Note that DBS act on the entire signal chain. If the system contains a temperature sensitive amplifier, these variations will also be corrected for.

2.2.2 Pattern noise suppression

Since one channel in the digitizer consists of more than one interleaved ADC core[4], and these cores are not 100% identical, mismatch errors in terms of pattern noise will disturb the baseline. Interleaving is a commonly used technique for increasing sample rate of ADCs and digitizers. The order of magnitude of the offset interleaving mismatch is typically around 100 codes, thus it is a significant noise source, perhaps the limiting one, if not compensated for. To suppress these mismatches, the DBS IP takes into

[3]This example has two ADC-cores.
[4]At least two, depending on the ADQ model.

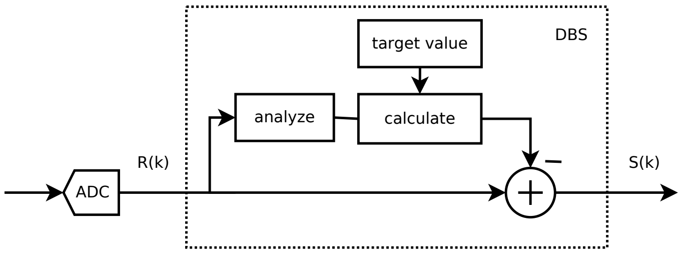


Figure 6: DBS principe of operation. Input data is analyzed and variations are compensated for. The baseline is adjusted to the target value set by the user.

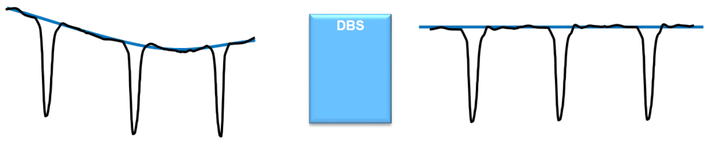


Figure 7: When enabled, DBS will constantly monitor the baseline and correct for variations.

account the number of ADC cores. Fig. 8 shows the analog input signal and an ADC with four interleaved cores and the output from each ADC core, illustrating the effects of component mismatch. Finally, the digital signal is restored by letting DBS estimate and correct for the baseline differences.

2.2.3 Track rapid baseline variations

Pulse leakage, noise and interference can give rise to *rapid* signal variations and potentially false trigger events. To reduce the sensitivity to these rapid variations, a moving average (MA) filter can be added. The filter becomes a slope detector, detecting on an, on average, monotonic increase/decrease of signal level (a pulse edge) rather than on rapid noise variations. A block diagram on how the MA filter is included is shown in fig. 9. The MA filter is set in the range 4 to 100 samples, implying an update rate in the order of 100 *ns*, to be compared with DBS with time constant in the order of 1 *ms*.

2.3 Define the region of interest

Once a stable baseline is found, one can start focusing on data that is *not* baseline data. These samples form regions of interest with either positive or negative pulses, depending on the application at hand. The

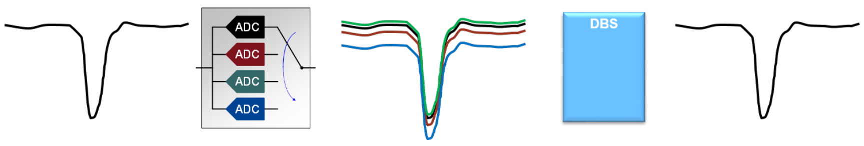


Figure 8: DBS can suppress pattern noise due to offset interleaving mismatch. The given example has four ADC cores.

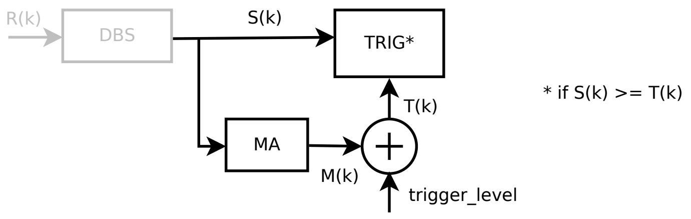


Figure 9: A block diagram illustrating how the MA filter is included after DBS. If enabled, it will affect the trigger threshold $T(k)$.

samples in the regions of interest can be saved and analyzed (Sec. 4), all other data will be rejected[5].

Positive pulses are pulses that grow toward *higher* ADC codes from the baseline.

Negative pulses are pulses that grow toward *lower* ADC codes from the baseline.

In addition to the pulses, the region of interest could also be the samples just before and/or after a pulse. This feature allows for study of the rising and falling edges of the pulses. An example of a region of interest is shown in Fig. 10.

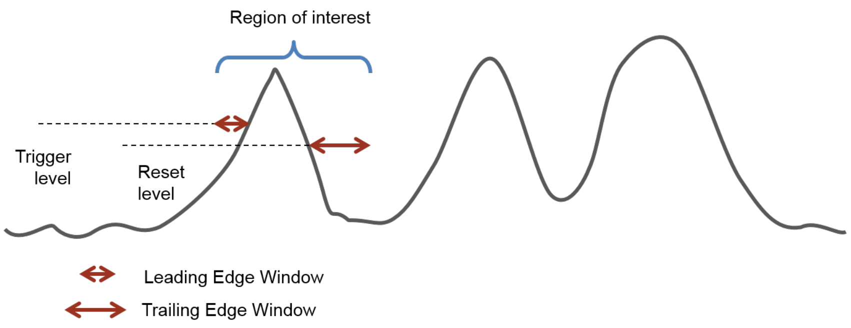


Figure 10: An example of a region of interest, consisting of the region between the start and the end of the pulse, as well as leading and trailing edge windows.

2.3.1 Find the start of the pulse – the trigger event

When searching for a pulse and specifically the start of a pulse (the trigger event) it is beneficial to have a data-driven trigger (instead of e.g. a software trigger or an external one). With such a data-driven *level trigger* the start of a positive pulse is defined as the first sample equal to or above the threshold $T(k)$. Consequently, the start of a negative pulse is the first sample equal to or below the threshold. The trigger event is illustrated in Fig. 12. The `trigger_level` is set relative to the MA filter (Sec. 2.2.3) and a trigger event will occur if

$$S(k) \geq T(k) = M(k) + \texttt{trigger\_level} \tag{1}$$

[5]This is mostly true, but because of the substantial flexibility of the Data collection, (described in Sec. 5) some modes collect samples outside regions of interest as well.

where $M(k)$ and $S(k)$ is defined as in Fig. 11 and `trigger_level` is set by the user. The comparison calculation consists of the following steps:

1. The present sample $S(k)$, where k is the sample index, is to be compared with a threshold level $T(k) = M(k) +$ `trigger_level`.

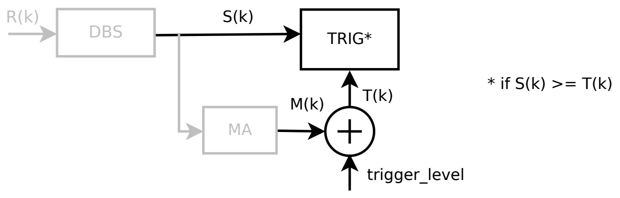


Figure 11: A trigger event will occur if $S(k) \geq (T(k))$.

2. $M(k)$ is calculated from previous samples as an average of a determined batch of samples. This is the output of the MA filter.

3. Which samples to use for calculating $M(k)$ is determined by the parameters `moving_average_delay` and `moving_average_length` as

$$M(k) = \sum_{m=0}^{L-1} S(k + m - (L - 1) - D) \tag{2}$$

where L = `moving_average_delay` and D = `moving_average_length`. Thus,

   - index to first sample is $k - D - (L - 1)$
   - index to last sample $k - D$.

4. The threshold level $T(k)$ is controlled by the user via the parameter `trigger_level`. An active MA filter makes $T(k)$ dynamic, as shown in (3).

$$T(k) \approx \texttt{dc\_taget} + \texttt{trigger\_level} \tag{3}$$

The `dc_target` is the user-defined value to which DBS will adjust the baseline, see Sec. 2.2.1.

5. If the start of the pulse is also the start of a record, k will be the reference point of the Peak value timestamps (Sec. 4.1.3) within that record.

The length of the MA filter, L = `moving_average_length` , and the delay D = `moving_average_delay` together has to be less than 100 samples ($L + D \leq 100$). The L and D are measured in samples and the `trigger_level` is measured in ADC codes.

Note

With an activated MA filter, the threshold $T(k)$ is dynamic.

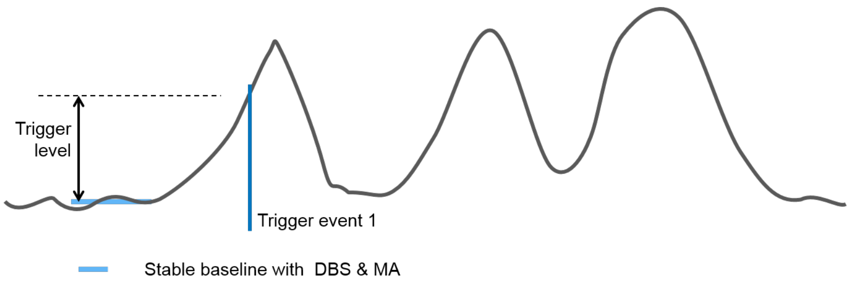


Figure 12: The start of the pulse is the trigger event.

2.3.2 Find the start of the pulse without MA filter

By setting $L = 0$, the MA filter is deactivated. The `trigger level` must then be entered in absolute ADC codes instead of relative to `dc_target`. The threshold level $T(k)$ will be *fixed* as shown in (4) and not dynamic as it is with a active MA-filter (3).

$$T(k) = \texttt{trigger\_level} \tag{4}$$

The start of a pulse will be the first sample satisfying (5) (or (1) with $M(k) \equiv 0$).

$$S(k) \geq \texttt{trigger\_level} \tag{5}$$

Important

The parameter `trigger_level` is interpreted differently depending on if the MA-filter is activated or not.

The difference between using DBS only and DBS in combination with MA-filter is illustrated in Figure 13 where the blue dotted line is $R(k)$ and the red line is $S(k)$. MA_OUTPUT is $M(k)$.

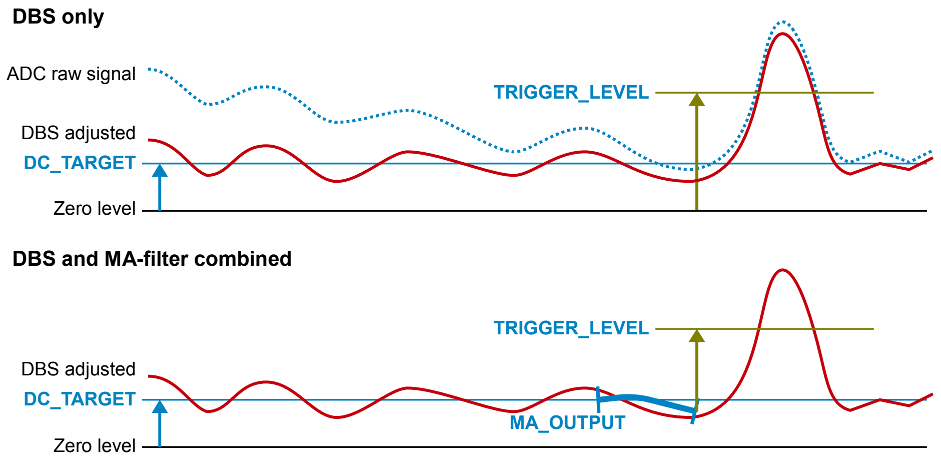


Figure 13: The user-specified `trigger_level` is defined differently, depending on if the MA-filter is enabled or not. With a active MA filter in the `trigger_level` perspective, `dc_target` will be the new zero-level.

2.3.3 Find the end of the pulse

The pulse ends when the signal level falls back towards the baseline again. More precisely, the last sample of the pulse is found when the signal is equal to or passes the *Reset level* on its way back to the baseline. This Reset event is calculated from the user-defined `trigger_level` and a hysteresis (offset). Equations (6) and (7) define the reset level for positive and negative polarity, respectively. The reset level for a positive pulse is also illustrated in Fig. 15a.

$$\text{Reset level} = \text{Trigger level} - \text{Reset hysteresis} = T(k) - \texttt{reset\_hysteresis} \tag{6}$$

$$\text{Reset level} = \text{Trigger level} + \text{Reset hysteresis} = T(k) + \texttt{reset\_hysteresis} \tag{7}$$

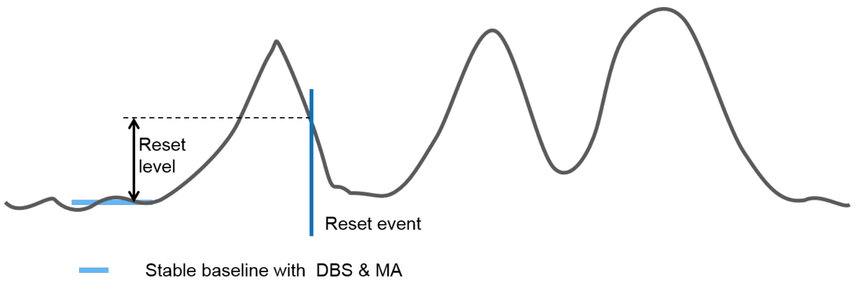
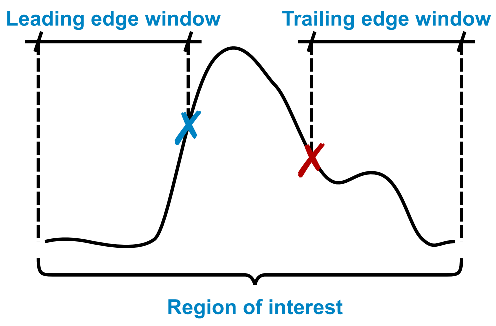


Figure 14: The end of the pulse is the reset event.

2.3.4 When pulse edges are of interest

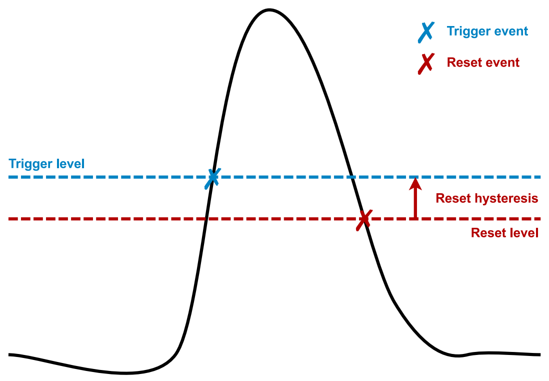
For some applications, the time regions just before and/or just after the pulse are of interest. By including the pulse edges as illustrated in the alongside figure, the region of interest is extended outward from the pulse which makes it possible to study the rising and falling edges of the pulses. The windows are named Leading Edge Window (LEW) and Trailing Edge Window (TEW), respectively.



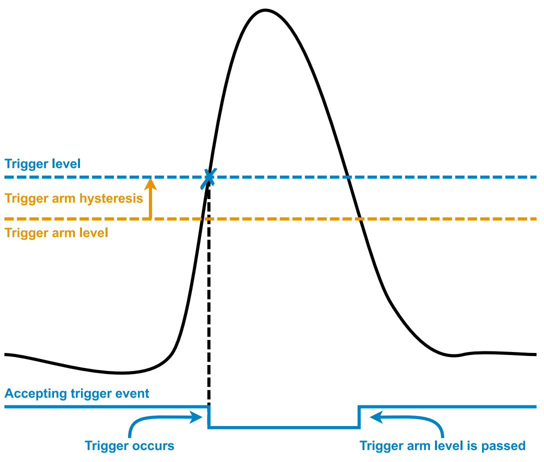
2.3.5 False events

In an automated system, there is an inevitable risk of detecting false events. There is always a trade-off between accepting false events and missing out on real pulses. A first sanity check of the system setup is of course to ensure that the trigger level $T(k)$ is above the noise floor.

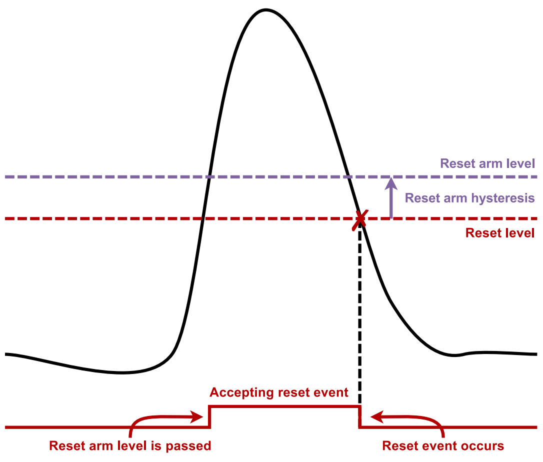
Another protection mechanism that can be used is the *arming hysteresis*. It is available for both the trigger and the reset event and is defined as shown in Figs 15b and 15c. The purpose of the arming hysteresis parameters `trigger_arm_hysteresis` and `reset_arm_hysteresis` is to avoid trigger and reset on noise, as shown in the example in Figs. 15d and 15e.



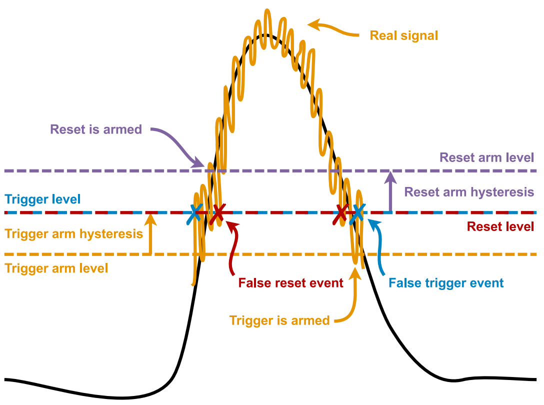
(a) The reset level is determined from the trigger level *and the* reset hysteresis.



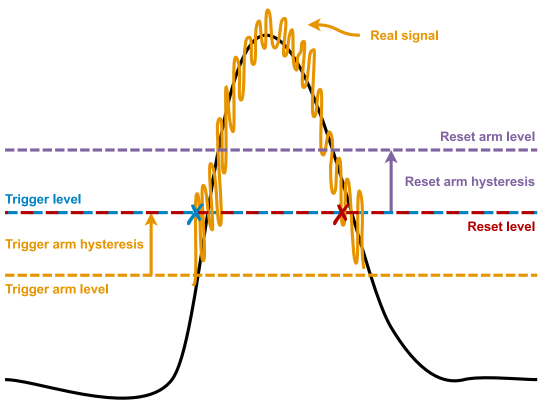
(b) The trigger arm level is determined from the trigger level *and the* trigger arm hysteresis. *The level is used to arm the mechanism generating trigger events.*



(c) The reset arm level is determined from the reset level *and the* reset arm hysteresis. *The level is used to arm the mechanism generating reset events.*



(d) Badly configured arm levels may yield false events from the incorrectly detecting pulses within the signal noise.



(e) Setting the arm levels greater than the noise variation protects against false events.

Figure 15: Figs. 15a–15c demonstrates the pulse definition in the context of a positive pulse. Figs. 15d and 15e highlights the purpose of the arm levels.

⚠ Warning

Bad setting of the arming hysteresis can make things worse, see example in Fig 16.

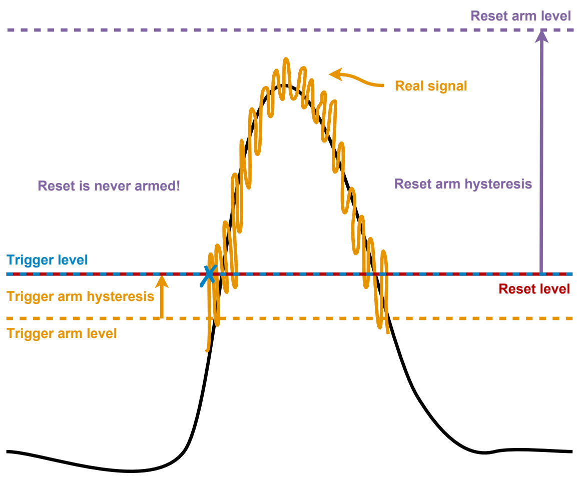


Figure 16: Example of a bad choice of reset arm hysteresis. The reset event will be missed.

3 Timing rules for when to accept pulses

In some applications, everything is not of interest all the time. In Section 2, it was assumed that all pulses where of interest, independently on what happened in the overall system or what was detected on the other digitizer channels. This chapter will focus on situations where such external/global signalling or dependencies between channels exist.

The external or global signal can open up a *Detection window*. If an entire pulse train is of interest, the first pulse in the train can be used to open the window. Only pulses within the window will be considered to be regions of interest.

Channel dependencies can be controlled with *Coincidence*. Pulses that do not fulfill the conditions defined by the *coincidence mask* and the *coincidence windows* will be rejected. Both the Detection window and the Coincidence feature can further limit the regions of interest found using the theory in Section 2, but in different ways. Thus rules on when to accept pulses can be added.

Example

With detection window tied high and the coincidence mask set to default, all pulses are of interest all the time.

3.1 Detection window

The detection window defines a period in time in which pulses are accepted. The detection window has a *fixed length* and an associated trigger event, the *Detection window trigger*, see Fig. 17. There are different ways to open up a detection window.

Software trigger A software command opens the detection window. Useful mostly during system setup and debugging. All channels will have the same detection window.

Internal trigger The internal trigger can open a common detection window for all channels. In a multi-unit system, this internally generated trigger can be used to control triggering of external equipment (Sec. 7.2).

External trigger An external signal on the TRIG connector opens a common detection window for all channels.

Level trigger The per-channel level trigger can open up a data-driven individual detection window. Thus, the first pulse detected on a channel will open the fixed length detection window. See also Fig. 17

The global trigger sources are also treated in Sec. 7.1.

Note

The detection window can be triggered by different sources but the length of the window is fixed.

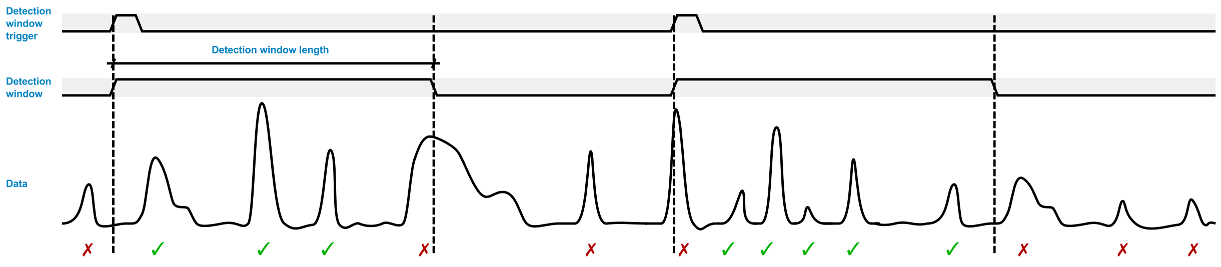


Figure 17: An example of how a detection window is used to mask undesired pulses. Pulses marked with a green check mark are accepted. Pulses marked with a red x*-mark are rejected.*

3.2 Channel dependencies

In some applications there are dependencies between different channels. For example, events on channel A is only of interest just after an event on channel B. Turning on the coincidence trigger function enables detection of simultaneous events on different channels. *Combinations* of pulses will create a trigger. The coincidence function is controlled by a window to define a time for a coincidence, Section 3.2.1, and a mask to select combinations of channels, Section 3.2.2. Figure 18 illustrates a trigger on the first channel activates a coincidence window. Since the trigger on the second channel falls within the coincidence window, it is accepted. If coincidence is turned off (default), each channel is triggered independently of all other channels.

Note

By default, all channels are triggered independently on each other.

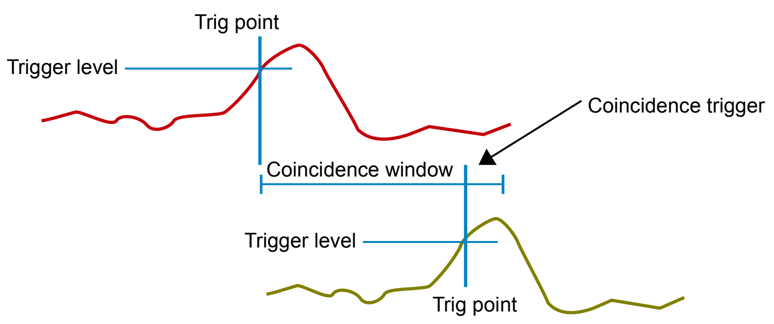


Figure 18: Coincidence trigger overview. The green channel is dependent on the red channel. Green channel triggers are only accepted if the red channel has triggered. Coincidence triggers are only accepted within the Coincidence window.

3.2.1 Coincidence window

The coincidence window is a window that is opened after a channel has triggered. E.g. a trigger event on channel A opens the coincidence window Win_A, as illustrated in Fig. 19. The window defines a space in time where trigger events on other channels should be accepted, *if* these channels are dependent on channel A. That is, given that other channels are dependent on activity on channel A to be of interest, their trigger events will be accepted within Win_A only. The pulses of two dependent channels coincide if the time between their trigger events is smaller than the coincidence window. One can see it as $Trigger_A$ is stretched in time with the length of the coincidence window Win_A.

Note

The pulses of two dependent channels coincide if the time between their trigger events is smaller than the coincidence window.

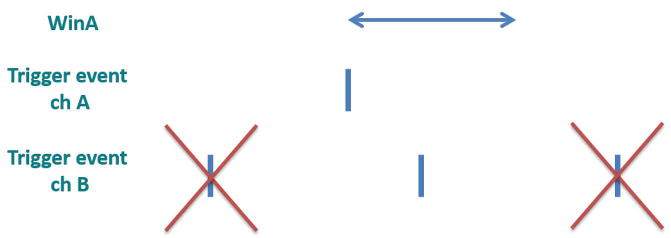


Figure 19: Coincidence window for channel A and exemplifying trigger events on channel B with $Mask_{BA}$ = 1, that is channel B is dependent on channel A.

3.2.2 Coincidence mask

The coincidence mask defines what combinations of trigger events that should be accepted, that is how channels depend on each other. There is one mask per channel $Mask_X$ with 4 bits as shown in (8).

$$Mask_X = [Mask_{XD}, Mask_{XC}, Mask_{XB}, Mask_{XA}], \quad \text{for } X = A, B, C, D \tag{8}$$

The equation for a coincidence event on channel A is

$$Coincidence_A = (Mask_{AA} \cdot Win_A + Mask_{AB} \cdot Win_B + Mask_{AC} \cdot Win_C + Mask_{AD} \cdot Win_D) \cdot Trigger_A \tag{9}$$

where the bits $Mask_{AX}$ are explained in (8) with $X = A$, and Win_X is the trigger event stretched in time as shown in Figure 19. The default coincidence mask, without channel dependencies is shown in (10).

$$\begin{bmatrix} Mask_A \\ Mask_B \\ Mask_C \\ Mask_D \end{bmatrix} = \begin{bmatrix} Mask_{AD} & Mask_{AC} & Mask_{AB} & Mask_{AA} \\ Mask_{BD} & Mask_{BC} & Mask_{BB} & Mask_{BA} \\ Mask_{CD} & Mask_{CC} & Mask_{CB} & Mask_{CA} \\ Mask_{DD} & Mask_{DC} & Mask_{DB} & Mask_{DA} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{10}$$

Example

Setting up the mask to trigger each channel individually, (10), is the same as turning coincidence off.

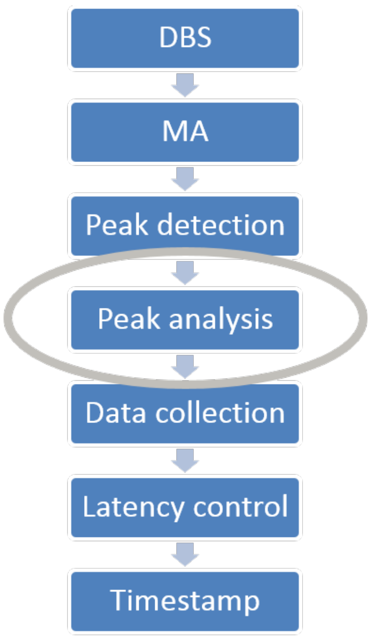
Various examples of coincidence masks are given below.

1. Accept trigger events on channel A only if channel B has triggered: Set $Mask_A$ = [0010].
2. Accept trigger events on channel A only if channel B or D has triggered: Set $Mask_A$ = [1010].
3. Accept trigger events on channel C independently of other channels: Set $Mask_C$ = [1100] or [0100] or [0110] or [0101] or any other combination where $Mask_{CC}$ = 1.
4. Accept trigger events on channel C if any other channel has triggered: Set $Mask_C$ = [1011].

Note

Note that setting the bit $Mask_{XX}$ in the mask for channel X overrides all coincidence and the channel is individual as in (10).

4 Pulse analysis



In several applications, the pulse activity is so high that it is not possible to send raw data of all the accepted pulses without over-loading the host PC interface and storing capacity. Information will then be lost. To overcome this problem, the analysis and characterization of the pulses can be made in real-time in the FPGA with the result that only metadata of the pulses need to be communicated to the host. This will reduce the load on the host interface significantly. Also, CPU processing will decrease considerably. Three key pulse metrics are available; namely *peak value*, *pulse width*, and *peak timestamp*. This chapter will describe how they are defined and calculated. It will also discuss how to add other, more sophisticated or application specific, metrics if needed.

4.1 Overview of the standard metrics

The pulses detected according to Sec. 2 can be analyzed in real-time in the FPGA. The peak value, Section 4.1.1, and the pulse width (as time-over-threshold), Section 4.1.2, of each pulse is calculated. Each pulse is also given a peak timestamp, Section 4.1.3, to enable chronological sorting. The peak value metrics and the time-over-threshold metrics are additionally summarized in histograms for statistical analysis. As stated earlier, building up the histograms directly in the firmware greatly reduces the data rate to the PC and the processing load of the CPU.

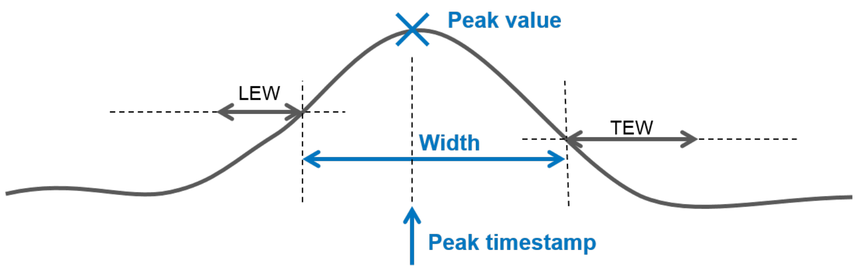


Figure 20: Illustration of the standard analysis metrics peak value, pulse width, and peak timestamp. LEW and TEW is not included in the pulse width.

4.1.1 Peak value

The peak value defines the maximum or the minimum value of a the pulse. For positive pulses the peak is defined as the maximum value, and for negative pulses the peak is defined as the minimum value. The peak is represented by a 16-bit signed value and an illustration is given in Fig. 21.

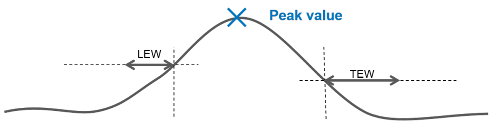


Figure 21: Peak value illustration.

4.1.2 Pulse width – time-over-threshold (TOT)

The width of a pulse can be defined in different ways. A straight-forward way, which is also implementation-efficient, is to compute it as the number of samples between the start and the end of a pulse[6]. With the definitions in Sections 2.3.1 and 2.3.3, that would be the same as the number of samples between the pulse trigger and the sample where the pulse passes the reset level. This definition of pulse width is called time-over-threshold (TOT) and is always ≥ 1. The width is represented by a 16-bit word (two bytes) and will thus wrap if the pulse width is larger than 2^{16} - 1 samples. The pulse widths of detected pulses are summarized in a histogram.

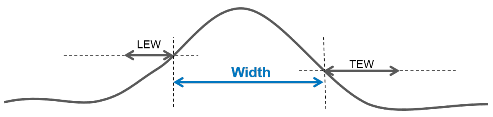


Figure 22: Pulse width illustration.

4.1.3 Peak timestamp

A timestamp is useful when the exact time of the detection of the pulse is of interest; both relative to other pulses and as an absolute measurement from the system reset. The available metric *peak timestamp* is defined as the time (in samples) between the start of the detection and the peak value.

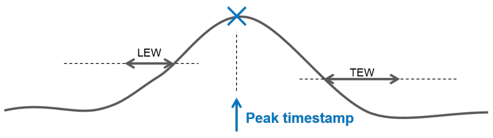


Figure 23: Peak timestamp illustration.

The peak timestamp is (in other words, and more exact) the sample index at which the extreme value occurred, relative to the *record header timestamp*. For this sentence to be helpful, one need to be acquainted with the concept of *records*. The reader can here jump to Chapter 5 and learn about records, timestamps, and *collection modes*, or just settle with the information that a record is the format in which data is collected and sent to the end user. A record has a header and a data part, and the header includes timing information in form of a record header timestamp. The content of the data part of the record depends on the collection mode. The record [header] timestamp and the peak timestamp together enable relative timing analysis between pulses as well as absolute timing analysis.

For applications with repetitive processes it is convenient to be able to reset the record timestamp. A *timestamp reset* engine is therefore available to enable both single and periodic resets of the record timestamp, see Sec. 7.3.

Note

The peak timestamp is the sample index at which the extreme value occurred, relative to the record header timestamp.

[6]Another more sophisticated pulse width definition is full-width-half-max (FWHM).

For wider or obtuse peaks, the peak value might be reached several times. In this case, the peak timestamp value is defined as the time of the *last* extreme value. Finally, it is of interest to know how large a peak timestamp can be. The peak timestamp is a 32-bit unsigned value and will thus wrap if the distance between the record header timestamp and the pulse peak is larger than 2^{32} - 1 samples.

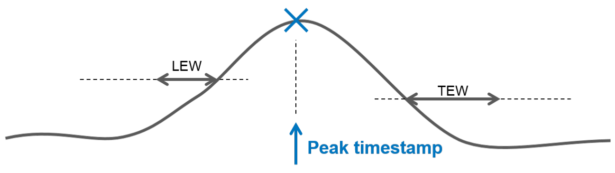


Figure 24: Peak timestamp illustration.

4.2 Intense bursts of pulses

If the application at hand generates intense bursts of pulses, it is important to ensure that the throughput is high enough. To ease the requirements on momentary throughput under such circumstances, a buffering FIFO is implemented. If the pulse rate still exceeds the processing bandwidth, overflow flags are available, allowing for user surveillance.

The pulse characterization module can on average analyze 500 million pulses/second at 2 GSPS and 250 million pulses/second at 1 GSPS. The module will buffer up to 2048 pulses at 2 GSPS (1024 pulses at 1 GSPS) after which the characterization module overflows. The histogram processing is slightly more sophisticated and will thus limit the throughput. The maximum average processing rate of the histograms is 33 million pulses/second. For higher rates, the histogram overflow flag will be set.

4.3 Custom-made analysis

The three available metrics (peak value, pulse width and peak timestamp) are straight forward and simple to use, but specific applications might require custom-made pulse analysis. The FPGA is therefore open for custom designs in the User Logic 1 IP block with the firmware development kit *ADQ DevKit* for FWPD. TSPD's design service is available to support the implementation work. Examples of custom analysis is given below and is illustrated in Fig. 25.

Note

The FPGA is open for custom designs of application specific pulse analysis.

- Area
- Power
- Custom peak definition
- Custom width definition
- Gaussian curve fit

- FWHM – full width at half maximum
- FWTM – full width at tenth of maximum
- Spline interpolation
- Qualify/disqualify pulse

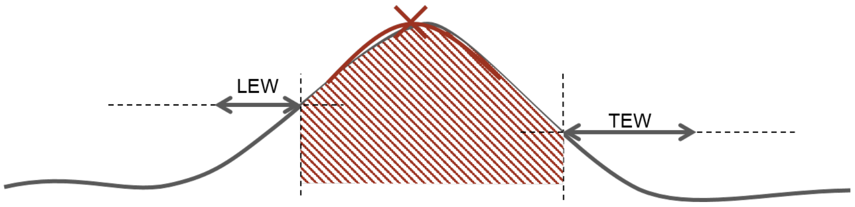


Figure 25: The user can implement application-specific pulse analysis such as area or gaussian curve fit.

4.4 Histograms

The firmware histogram module provides histograms for the metrics *peak width* and *peak value* of the pulses. Note that the histogram operation is independent of the data collection mode[7]. The number of histogram bins are listed in Table 1. Each histogram bin is represented by a 20-bit unsigned number and will saturate at the maximum value (2^{20} - 1 = 1 048 575). The data is mapped to a bin number by

$$\text{Bin}[n] = \left\lfloor \frac{(x + \delta) \cdot \alpha}{1024} \right\rfloor , \tag{11}$$

where x is the histogram metadata value, δ is an offset and α is a scale factor.

Table 1: The number of bins for the two different histogram.

Type	**Number of bins**
Pulse peak value	2^{14} = 16384
Pulse width	2^{12} = 4096

[7]Data collection modes are described in Sec. 5.3.

Example

To span the range of the digitizer, set δ = 32 768 and α = 256 since:

1. The data from the 14-bit digitizer is represented as 16 bits MSB aligned words. The range in two's complement representation is thus $[-2^{15}, 2^{15} - 1]$.
2. The offset 2^{15} = 32 768 converts the two's complement values to all positive values.
3. The effective scale is 256/1024 = 1/4. The result is that the 16 bits MSB aligned code is converted to a 14 bits representation and each bin is one digitizer code.

Values falling outside the range of the histogram are collected in overflow and underflow bins. That is, if the resulting bin number from (11) is outside of the allowed range. More precisely, if the bin number is negative, the underflow bin will be incremented by one. Conversely, if the bin number is larger than the total number of bins (Table 1) the overflow bin will be incremented by one. A histogram example is shown in Fig. 26.

Example

Let a certain pulse have the peak value -5000 and the peak value histogram be configured with scale factor α = 1024 and the offset δ = 4000. The resulting bin number would be -1000, from (11). Since this is less than 0, the underflow bin would be incremented by one.

The total number of peak values, that is the sum of the histogram, is also available for read-out.

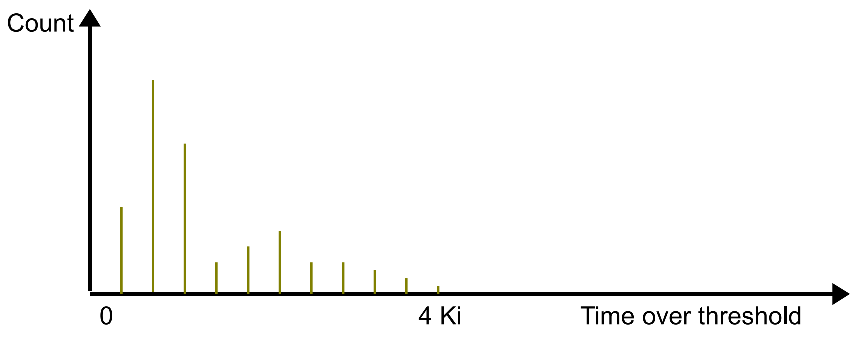
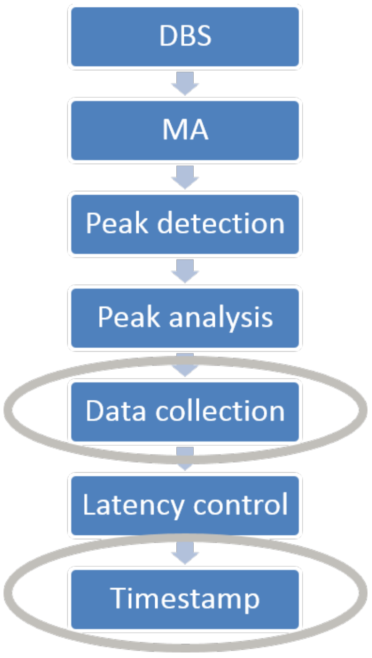


Figure 26: Pulse width histogram example, where Ki = kibi = 1024^1.

5 Data collection



Chapters 5 and 6 are a bit of a catch-22. In order to understand data collection it is favorable to know a bit about latency control (by padding) and vice versa. The reader is recommended to jump back and forth between the chapters, when needed. This chapter will look into how the detected data and metadata can be collected and sent to the user. A number of collection modes are available, each of them being suitable for different situations. Some to use during operation and others to facilitate set-up and debugging of the system. Once the reader is a little bit acquainted with the different collection modes, a section on timestamp is given (Sec. 5.4). The modes can also be combined, to further increase the collection flexibility (Sec. 5.5). Before digging into details on the different collection modes, the concepts of *records* and *frame length* are introduced.

5.1 Records

A record is the format in which data is collected and sent to the user. A record has a header and a data part. The header includes information about the length and a timestamp (named trigger timestamp or record header timestamp). For more information on timestamps, see Sec. 5.4. The contents of the data part will depend on the *data collection mode*, see Section 5.3.

Note
A record is the format in which data is collected and sent to the user.

Records can either have fixed or variable length as illustrated in Fig. 27.

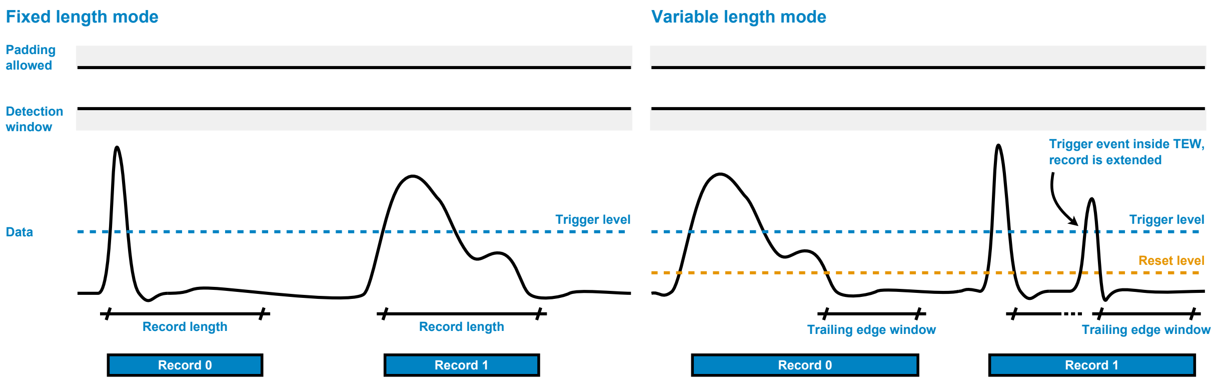


Figure 27: Fixed length vs variable length records. With fixed length records, the data does not affect the record length, only the start of the record. TEW does not exist. With variable length records, both the record start and the record length is data-driven. If a new trigger event is detected before TEW has ended, the current record will continue until the TEW of the second pulse has ended. Thus the record will include two pulses.

Fixed length record A fixed length record is defined by the trigger event and the record length. The length of a fixed length record is specified by the user, and is independent of the data collected. A fixed length record can be triggered by any of the trigger sources[8].

Variable length record The length of a variable length record will depend on the data. A variable length record is defined as the data between the trigger event and reset event. If trailing or leading edge windows are used, these will also be included in the record. If two pulses have overlapping trailing and leading edge windows, they will be concatenated into the same record. The variable length mode can only be used with the level trigger.

5.2 Frame length

The frame length is the length of all data collected within one *frame*. The frame has different names depending on collection mode[9]. A frame is the constant distance between two frame triggers. In Mode 1 and 3, the frame is called Detection frame and in Mode 4 the name is Padding frame. The trigger is named Detection window trigger[10] or Padding trigger[11]. The reason for the different names in the different modes is simply that the detection window does not exist in Mode 4, but still a trigger and a frame is needed to define regions where it is OK to pad. Thereby the Mode 4 specific names *Padding trigger* and *Padding frame*.

The frame length is the length of one or more records. In Mode 1, there is always exactly one record per frame, thus (12) where N_P is the number of metatada packages generated during the current frame. In Mode 3–4, the frame length is the sum of the lengths of the records collected since the last frame trigger, including eventual padding records, see (13), where N_R in (13) is the number for records generated during a frame.

$$\text{frame length} = \mathit{length}(\mathit{record}) = N_P * 8\mathit{bytes} \tag{12}$$

$$\text{frame length} = \sum_{n=1}^{N_R} \mathit{length}(\mathit{record}(n)) + \mathit{length}(\mathit{padding\_record}) \tag{13}$$

Note

The frame length is the length of all data collected within one frame, from one external trigger to the next.

5.3 Data collection modes

There are five data collection modes available for different purposes, with the names *Mode 0 – Mode 4*. Each of them are discussed in detail below.

[8]Available trigger sources are SW trigger, internal trigger, external trigger, and level trigger.
[9]The different collection modes are described in detail in Sec. 5.3
[10]Mode 1 and 3.
[11]Mode 4.

5.3.1 Mode 0 – Raw data

Mode 0 is the raw data mode. It is also the default mode where most of the features, such as padding and detection window, are turned off. When a trigger comes, data is simply collected. With fixed length records any of the four available trigger sources can be used, but variable record length goes with the per-channel data-driven level trigger only. Data does not necessarily include pulses. With a fixed length record and trigger source other than the level trigger, the likelihood of catching a pulse is in most applications miniscule. Mode 0 is used mostly for debugging and during system set-up. It can also be used to achieve a variant of Mode 1, with the metadata calculations moved to the host. The level trigger can trigger a detection window equivalent and open up a fixed length frame. The raw data is collected in a record and sent to the host for offline analysis and characterization, and finally comparison with the metadata collected in parallel in Mode 1.

Note

Mode 0:

- The default mode.
- Fixed length records with any trigger source.
- Variable length records with level trigger.
- No padding.
- No detection window.

5.3.2 Mode 1 – Pulse metadata

In Mode 1, the detection window opens up a record which is stuffed with metadata packages shown in Fig. 28. A metadata package is eight bytes long and has three fields carrying information on one detected pulse. The three fields are the pulse peak timestamp, the peak value, and the pulse width (as time-over-threshold). Half of the package (four bytes) is dedicated for the timestamp and the other half is split between the other two.

pulse peak timestamp (4 bytes)	peak value (2 bytes)	pulse width (2 bytes)

Figure 28: A metadata package consists of four bytes timestamp and two bytes of pulse peak value and pulse width each.

Mode 1 requires a detection window and a detection window trigger, telling the system when a new frame starts and how long it is. Each time a new frame starts, a new record is opened up and the frame length is reset. The metadata mode can be used both with and without padding. With padding, it is suitable for mass spectrometry applications. It is the most powerful mode in terms of data reduction and is therefore superior for applications with frequent pulsing and insufficient data transfer bandwidth. An example of data collection in Mode 1 is illustrated in fig. 29. Note that the padding offset coincides with the Detection window and that the minimum frame length is controlled via the parameter `minimum_frame_length`.

Note

Mode 1:

- Major data reduction (zero suppression).
- Packages of metadata are structured in records.
- Package size is 8 bytes.
- Detection window is required.
- Pulses are detected using the level trigger.
- Detection window opens up a record and stuff it with metadata packages.
- The frame length coincides with the record length, as shown in (12).
- With or without padding.
- With padding, the record length is guaranteed to be $\geq$ minimum frame length (defined by the user).
- Padding offset coincides with the detection window.

5.3.3 Mode 2 – Every N^{th} record

In applications with high frequency pulsing, the bandwidth of data transfer to host might not be enough to send raw pulse data. A way to circumvent this problem is to not send all raw data records to the user. Mode 2 is the same as Mode 0 with the exception that only every N^{th} record is sent. The rest is rejected. Depending on the choice of N, Mode 2 is a powerful data reduction tool. One example on how to use Mode 2 for system analysis is the following: If the external trigger is used to trigger a detection window, Mode 2 can be used to verify the analysis made in Mode 1.

Example

The external trigger can be used to trigger a detection window in Mode 2 with offline pulse analysis in the PC. The result can be used to verify that the real-time analysis made in Mode 1 is correct.

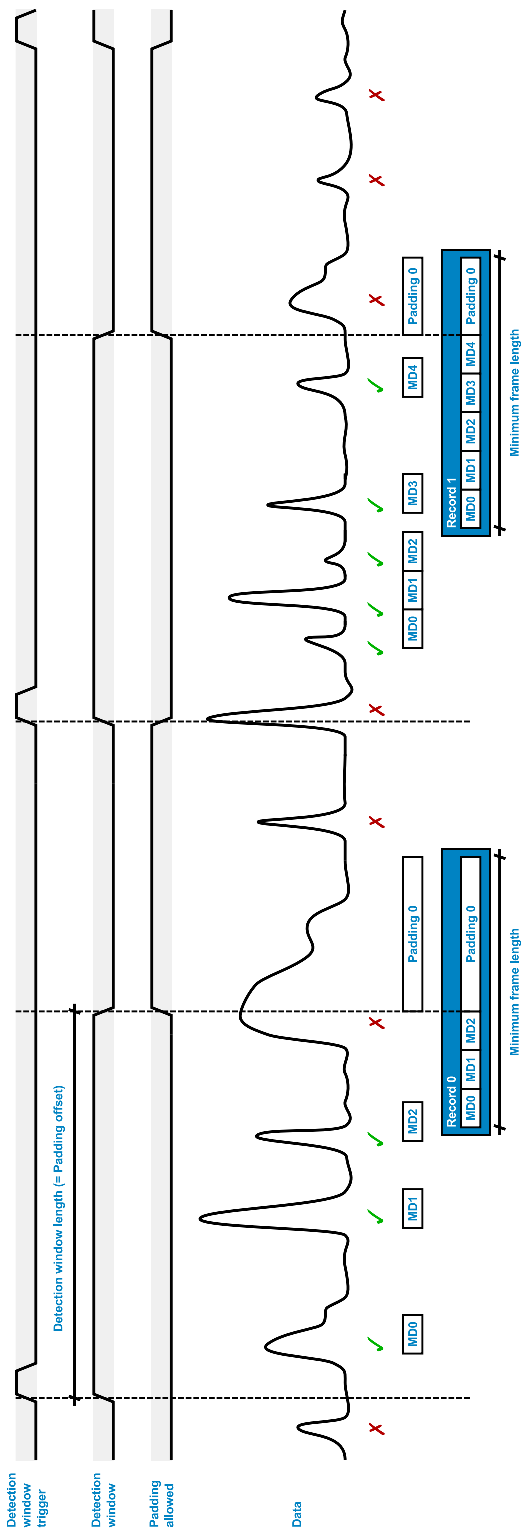


Figure 29: In the metadata mode (Mode 1) with padding, the records are populated with metadata packages. If the record from a frame is too short, a padding packaged with zeros will be included to ensure a minimum record length. Pulses marked with a green check mark are accepted. Pulses marked with a red x-mark are rejected.

Note

Mode 2:

- Same as Mode 0 with one exception.
- Only every N^{th} record is transferred to the host.
- Depending on the choice of N, significant data reduction compared to Mode 0.
- Of interest during system setup for applications with frequent pulsing.

5.3.4 Mode 3 – Raw pulse data all-inclusive

Mode 3 is – as opposed to Mode 0 – the most flexible mode with all features enabled. A flow graph of the data collection is given in Fig. 30 and an example is illustrated in Fig. 32a. When the frame trigger comes (Detection window trigger), pulses are detected with the per-channel data-driven level trigger and collected in records. If the pulses' regions of interest overlap, they will end up in the same record (Fig. 31). If they do not overlap, one record per pulse will be generated. The record generation will continue until the Padding offset[12] is reached. If the total length of the collected data in the current frame falls short of the required *minimum frame length*, a padding record will be added. The total length of the records sent to the user during the current frame is thus $\geq$ the user-controlled parameter `minimum_frame_length`. Any pulses coming in after the Padding offset but before the next frame trigger[13] will be rejected. Only when the next detection window trigger (frame trigger) comes, new pulses are collected.

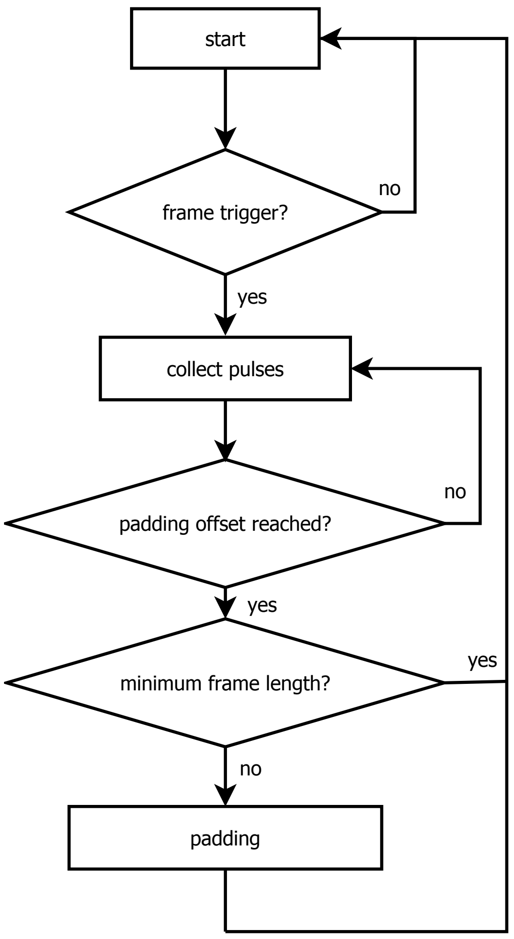


Figure 30: Flow graph of data collection in Mode 3 and Mode 4.

[12] The Padding offset is in Mode 3 the same as the length of the Detection window.
[13] In Mode 3, the frame trigger is the Detection window trigger.

Note

Mode 3:

- The most flexible Mode.
- Detection window exists.
- Padding is enabled.
- Pulses are detected with the per-channel data-driven level trigger.
- The frame length is the sum of the lengths of the data records plus eventually a padding record.
- Padding offset coincides with the detection window.
- After the padding offset, if not the minimum frame length is reached, a padding record will be generated.

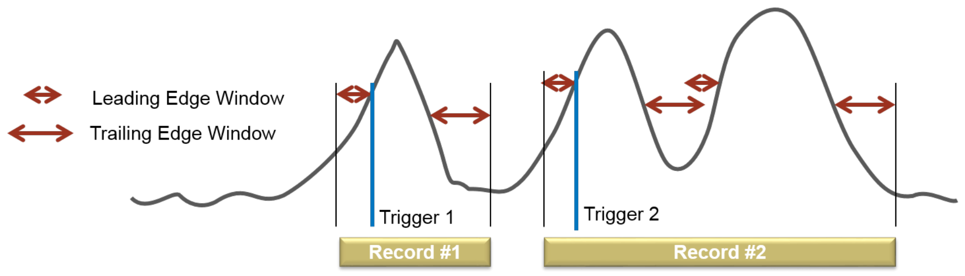


Figure 31: If trailing and leading edges of two pulses overlap, the will end up in the same record as shown for Record #2.

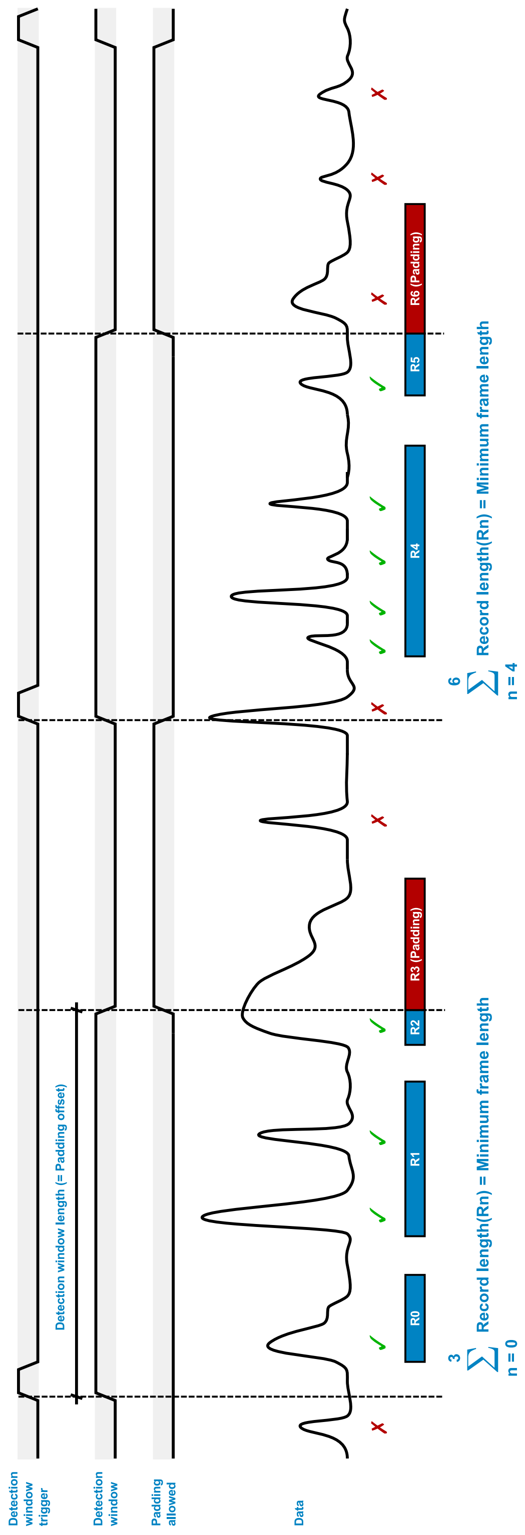
5.3.5 Mode 4 – Raw pulse data with padding without detection window

Mode 4 is similar to Mode 3, except for the Detection window. An example of data collection in Mode 4 is given in Fig. 32b and a flow graph (shared with Mode 3) is given in Fig. 30. When the application at hand cannot offer blind regions where it is OK to reject eventual pulses, the Detection window is tied high (detection all the time). In the absence of Detection window trigger, the *Padding* trigger defines the start of the frame (Padding frame). A consequence of not having any blind regions is that if pulses are detected after the padding offset, they will cut short any ongoing padding record generations. This will in turn, allow for more than one padding record per frame.

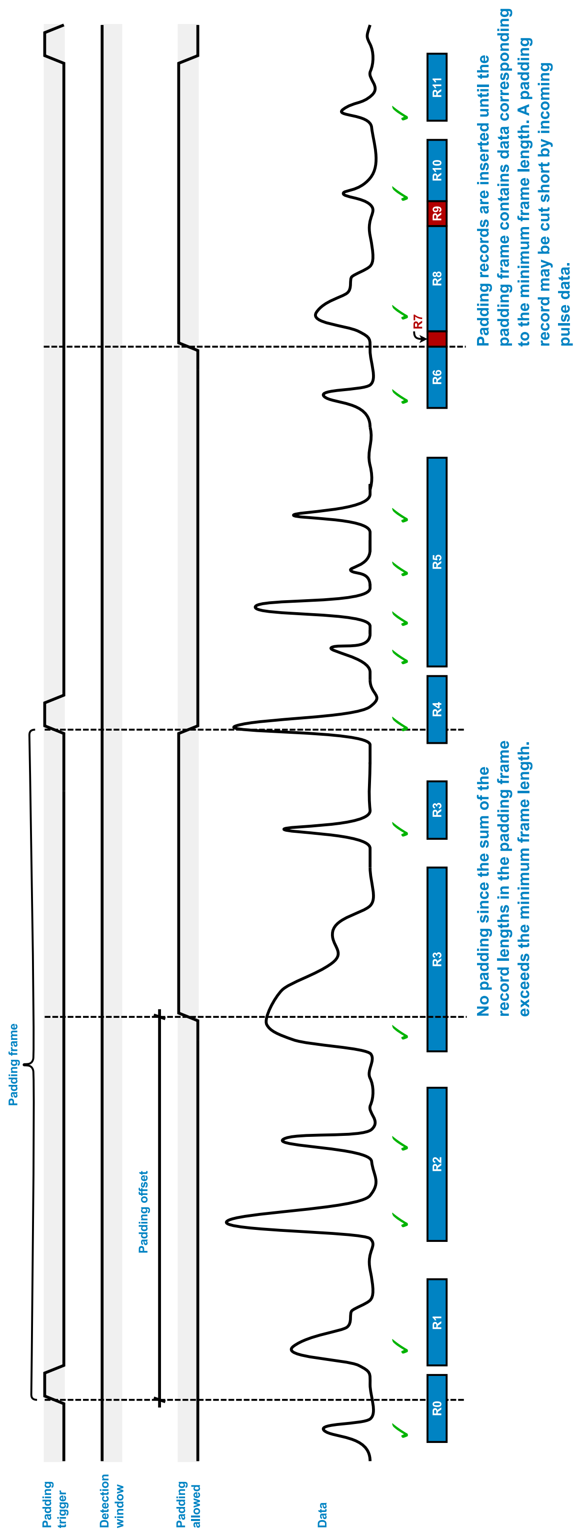
Note

Mode 4:

- Detection all the time (no detection window).
- Padding trigger (instead of Detection window trigger) indicates the start of a frame.
- The pulses are found with the data-driven level trigger.
- When Padding offset is passed, Padding record(s) are inserted until the padding frame contains data corresponding to the minimum frame length.
- Since pulses are accepted also after the padding offset (as distinct from Mode 3), a padding record may be cut short by incoming pulse data.


Detection window trigger
Detection window
Padding allowed
Data
Detection window length (= Padding offset)
R0
R1
R2
R3 (Padding)
R4
R5
R6 (Padding)
$\sum_{n=0}^{3}$ Record length(Rn) = Minimum frame length
$\sum_{n=4}^{6}$ Record length(Rn) = Minimum frame length

(a) Data collection in Mode 3 (all-inclusive).


Padding trigger
Detection window
Padding allowed
Data
Padding frame
Padding offset
R0
R1
R2
R3
R3
R4
R5
R6
R7
R8
R9
R10
R11
No padding since the sum of the record lengths in the padding frame exceeds the minimum frame length.
Padding records are inserted until the padding frame contains data corresponding to the minimum frame length. A padding record may be cut short by incoming pulse data.

(b) Data collection in Mode 4 (no detection window).

Figure 32: Raw pulse data collection with padding.

5.4 Timestamp

This section treats the subject of record header timestamps. It should not be mixed up with the peak timestamp discussed in Sec. 4.1.3. Each record has a header with timing information related to the data in the data part. Such a record header timestamp (or just timestamp) is a real-time timing measure of the event in the data part of the record. The timing information is relative to a user-defined time zero, that can be set an reset in several ways:

- Power up of the digitizer (happens automatically).
- Software reset of the time stamp (feasible but not recommended).
- Reset on external trigger signal.
- Reset on external sync signal.

The timestamp measures the number of clock cycles from time zero. To relate the internal clock to an external time measure, the clock reference of the digitizer can be locked to an external clock through the clock reference input CLK. The timestamp resolution has sub-sample precision of 125 ps and 25 ps for ADQ14 and ADQ7, respectively.

There is always one timestamp per record, even if it includes several pulses, and depending on the collection mode it either defines the start of a detection window, or the start of a pulse. The two cases are illustrated in Figs. 33 and 34. Fig. 33 typically illustrates Mode 0 and Mode 1, and Fig 34 Mode 3 and Mode 4. Note that the dedicated metadata channel (Pulse characteristics channel) is only available on ADQ7.

Note

The timestamp either defines the start of a detection window or the start of a pulse relative a user-defined time-zero.

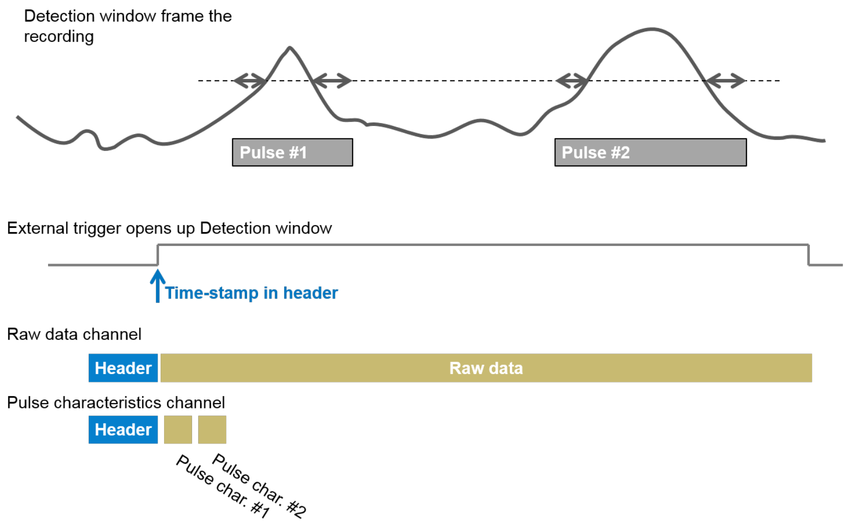

Detection window frame the recording
Pulse #1
Pulse #2
External trigger opens up Detection window
Time-stamp in header
Raw data channel
Header
Raw data
Pulse characteristics channel
Header
Pulse char. #1
Pulse char. #2

Figure 33: The external trigger can open up a detection window and set the record header timestamp.

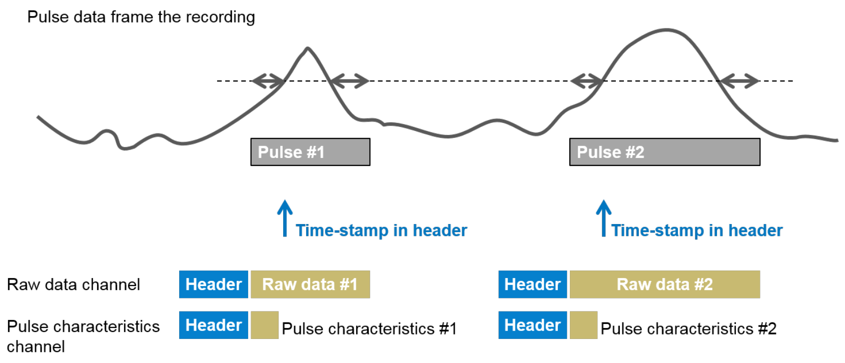


Figure 34: The data-driven level trigger can set the record header timestamp.

A example of collection in Mode 3 and timestamp reset by the external detection window trigger is shown in Fig. 35. Each record has its unique timestamp, even though they belong to the same frame. The pulse outside the window is rejected.

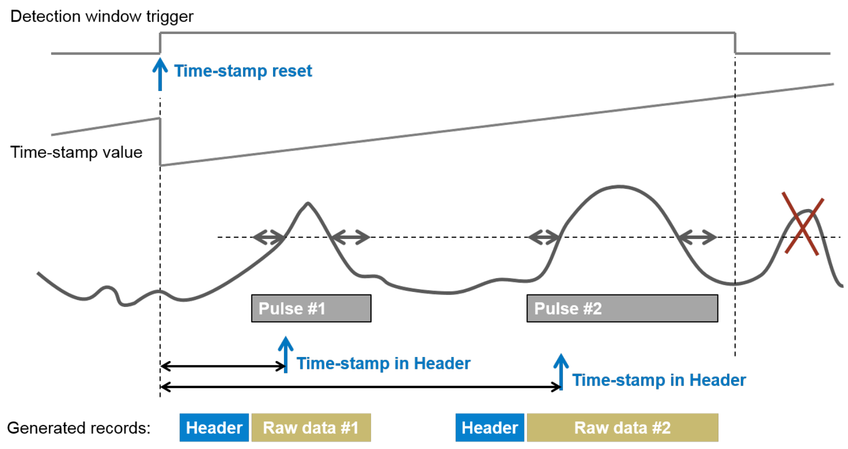


Figure 35: Timestamp example with timestamp reset at start of detection window.

Now with some knowledge about the header timestamp, it is time to study the relation between the peak timestamp (Sec. 4.1.3) and the header timestamp. Again there are two cases, depending on if the record content is triggered by data or by a detection window. Both cases are illustrated in Fig. 36.

5.5 Combination of collection modes with ADQ14

The ADQ14-FWPD includes a *data multiplexer* allowing combinations of the above described collection modes to further increase the flexibility and facilitate system setup and verification. For ADQ7, separate metadata channels offer full flexibility to collect both raw data and metadata all the time, thus this Section is ADQ14-exclusive.

There is one characterization engine available for each channel for individual and simultaneous pulse analysis and characterization. The multiplexer allows the user to connect *any* of the input channels to *any* of the characterization engines, as shown in Fig. 37. For example, by connecting the same input

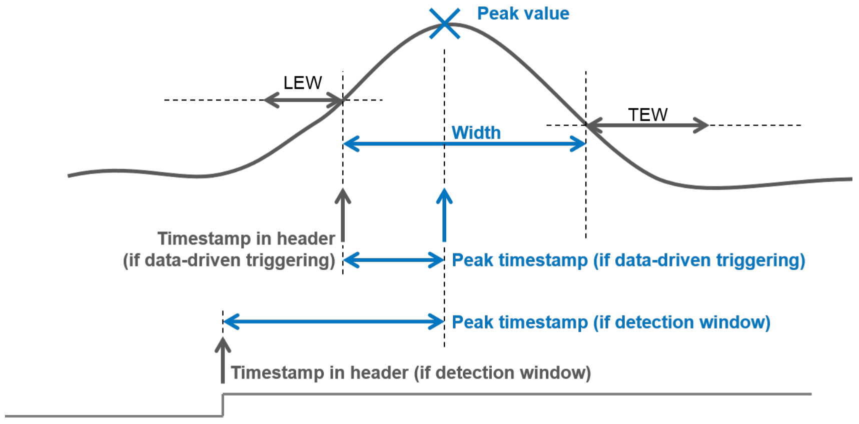

Peak value
LEW
TEW
Width
Timestamp in header
(if data-driven triggering)
Peak timestamp (if data-driven triggering)
Peak timestamp (if detection window)
Timestamp in header (if detection window)

Figure 36: The two cases of how the peak timestamp is related to the header timestamp illustrated in the same picture. The detection window (if it exists) is typically triggered by the internal or external global trigger source.

data channel to two different analyzing engines, both metadata and raw pulse data can be output on two different output channels. The data multiplexer is a helpful and powerful tool when verifying, for example, parameter settings.

Note

ADQ7 has separate dedicated metadata channels. ADQ14 offers data multiplexing as a powerful tool during system setup.

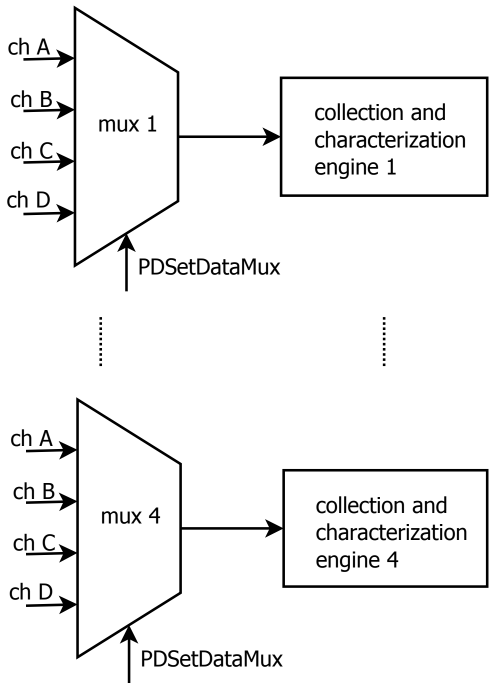

ch A
ch B
ch C
ch D
mux 1
collection and characterization engine 1
PDSetDataMux
ch A
ch B
ch C
ch D
mux 4
collection and characterization engine 4
PDSetDataMux

Figure 37: Illustration of collection mode flexibility with the data multiplexers on ADQ14. Any of the four input channels can be connected to any of the characterization engines.

6 Latency control

Imagine a bus at a bus stop. It's quite crowded already and on top of that, new people show up all the time. It would neither be popular nor cost-efficient to depart without filling up the bus as much as possible. Imagine now that the bus stop is almost empty. The surroundings feels desolated and lonely. The bus arrives and people enter. Now, waiting until the bus is cram-full before leaving is no longer a good idea. It might take the whole day or perhaps a week to fill the bus. Kids will be late for school...

Figure 38: *Latency control: Sometimes it is crowded at the bus stop, sometimes it is not. If not, it's a good idea to depart without waiting for the bus to be full.*

The example above including Fig. 38 illustrates what can happen without the possibility to control latency. After a reasonable time, the bus must take off, irrespective of how full it is. Latency in the pulse detection context can be controlled by adding the feature of *padding*. It will ensure a constant data rate from the digitizer. The padding block will append zeros to the data to ensure that the total amount of data measured over one *frame* (Sec 5.2) is at least equal to the *minimum frame length*. Setting the minimum frame length to zero disables the padding. The terms used to define the padding functionality is described below.

Padding package A padding package is a metadata package with all fields set to zero. The padding packages will be structured together with metadata packages in a record and sent to the user, to ensure that the minimum frame length is reached. Thus, that the record is long enough. Padding packages only exist in Mode 1 (the metadata collection mode), where each package represent metadata from one detected pulse. Data collection with padding packages is illustrated in Fig. 29.

Padding record A padding record is an entire record of padding. They are used with raw pulse data collection in Mode 3 and Mode 4, as illustrated in Figs. 32a and 32b. In general, each record contains one pulse, but if the region of interest of two pulses overlap, they will be structured in the same record. The data part of a padding record is set to zero and the record header contains information to the user to disregard the record. Mode 3 generates at most one padding record per frame, whereas Mode 4 might generate more than one (Sec. 5.3.5).

Padding offset Regardless of how sparsely the pulses are, there will be no padding before the time defined by *padding offset* has elapsed. In Mode 1 and Mode 3, the padding offset is the same as the length of the detection window. In Mode 4 (no detection window), it is a self-contained parameter. Padding offset is illustrated in Figs. 29, 32a, and 32b.

Padding trigger In Mode 4, the detection window does not exist (all pulses are of interest). Consequently, no trigger indicates the start of the detection. Yet, a trigger is needed to say where the padding offset region should start. This is the function of the padding trigger and it has the same available sources as the detection window trigger (see 3.1). The padding trigger exists in Mode 4 only and is illustrated in Fig.32b.

Padding frame In Mode 4 (no detection window), the padding frame defines how often a new padding offset region should start. One padding frame has the length of the padding offset *plus* a region in time where it is OK to add padding records (that is, if the pulses do not come often enough to fill up the minimum frame length, defined by the user). The padding frame exists in Mode 4 only, and is illustrated in Fig. 32b.

Note

Setting the `minimum_frame_length` to zero disables the padding.

7 Global trigger features

In addition to the per-channel data-driven level trigger, thoroughly discussed in Section 2.3, there are some *global* triggering functionality. There are three global trigger sources (internal, external and SW). Also features for trigger distribution to other equipments as well as functionality for trigger blocking is available. Note that these feature are general digitizer features and not specific for the pulse detection firmware option (–FWPD). However they might be useful in pulse detection applications, and that is why they are mentioned here.

7.1 Global trigger sources

Three global trigger sources are available and they all work with a fixed record length. Which source to choose (if any) is controlled via a global trigger mux. Each sources is described in more detail below. The trigger sources can be used to control e.g. the start of a detection window (Section 3.1) or a padding frame (in Mode 4). The global trigger mux is shown in Fig. 39.

Internal trigger source A trigger signal is generated internally in the digitizer using the internal reference clock. Can be used for example in trigger distribution (Section 7.2).

External trigger source The device is triggered by an external signal connected to the connector TRIG on the front panel.

Software trigger source The SW trigger functionality is of limited practical use during operation. However it is quite useful for debugging purpose and during system setup.

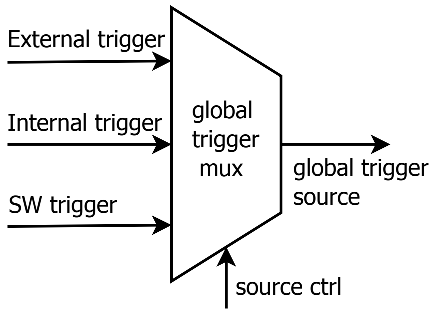


Figure 39: Control of global trigger source.

7.2 Trigger distribution

In some applications, it is useful and convenient to be able to control when and how often the equipment or system that generates the data to be detected is fired. The digitizer provides such a feature using the two connectors TRIG and SYNC on the front panel and a frame generator. The frame generator is controlled via three parameters:

Frame length Frame length sets the duty cycle of the sync signal. Please, note the unfortunate name collision of *frame length*. This parameter does not have anything to do with the frame length discussed in Section 5.2.

Frame factor Frame factor sets the period of the sync signal.

Edge Edge decides on which edge (rising of falling) the frame generator should look for when counting the internal triggers.

Note
The digitizer can be configured to control triggering of external equipment.

The bullet list below describes the process of trigger distribution and an example is given in Fig. 40. The output SYNC provides a frame for the free-running internal trigger signal.

- The internal trigger is set as trigger source (Fig. 39).
- The internal trigger is output on the connector TRIG.
- The frame generator listens to the internal trigger signal and counts the number of edges matching the edge type specified by the user.
- A wave form defined by the configuration parameters and the internal trigger is output on connector SYNC.

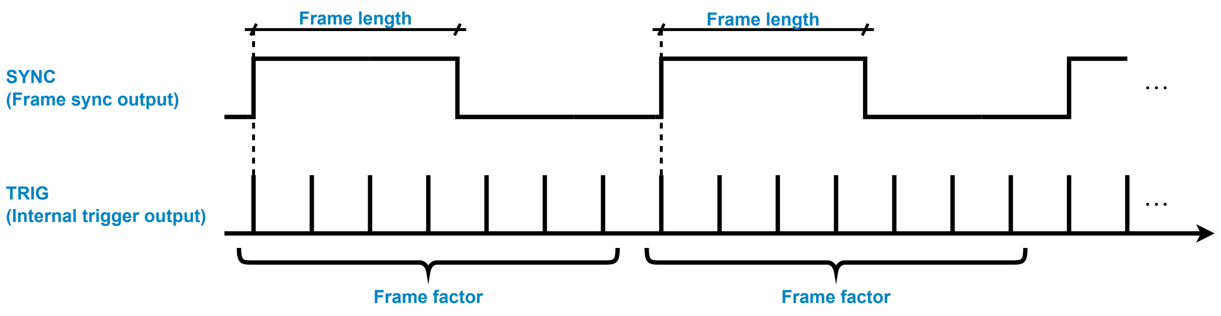


Figure 40: A trigger distribution example with frame factor 7.

7.3 Timebase synchronization

In application requiring more than one digitizer there is a need for timebase synchronization between the boards. The Timestamp Reset feature allows the user to input an external signal that resets the timebase of the digitizer. Reset of timebase means that the counter for the record header timestamp is reset (see also Sec. 5.4). The pulse peak timestamp counter is not affected. The function can be used to synchronize the timebase of multiple digitizers or to relate the timebase to some external event. This is useful for applications involving some repeated process.

Note

The Timestamp reset feature affects the record header timestamps and allows for timebase synchronization of several boards. It does not affect the pulse peak timestamp.

The timestamp reset engine normally listens for external events on either the external trigger connector (TRIG) or the synchronization connector (SYNC). Once armed, the first (Mode 0) or every (Mode 1) event on the external signal will reset the timestamp counter.

7.4 Trigger blocking

Trigger blocking is a layer on top of all the trigger and detection window functionality discussed earlier. If armed it inhibits the creation of records according to the chosen trigger blocking mode (there are four of them). The trigger blocking feature can be used together with the Timestamp Reset feature (Section 7.3) to ensure that the digitizer does not create records with unsynchronized timestamps.

The trigger blocking engine listens for external events on either the external trigger connector (TRIG) or the synchronization connector (SYNC). The four trigger blocking mode are described below.

Once *Mode index 0*

Once the trigger blocking engine is armed, this mode inhibits the creation of records until the first event on the trigger blocking source has been observed. At this point, the blocking is disengaged and all subsequent triggers are allowed to propagate.

Window *Mode index 1*

Once the trigger blocking engine is armed, this mode inhibits the creation of records until an event on the trigger blocking source has been observed. Following an event, the blocking is disengaged for a fixed duration (the *window length*) and triggers are allowed to propagate. Any additional events on the trigger blocking source that occurs during the window are ignored, i.e. they do not reset the window.

Gate *Mode index 2*

Once the trigger blocking engine is armed, this mode inhibits the creation of records until an event on the trigger blocking source has been observed. Following an event, the blocking is disengaged for as long as the trigger blocking source signal remains 'logic high' or 'logic low' (depending on the source's edge specification) and is reengaged once an event of the opposite type is detected.

Inverse window *Mode index 3*

Once the trigger blocking engine is armed, this mode allows triggers to propagate until an event on the trigger blocking source has been observed. Following an event, the blocking is engaged for a fixed duration (the *window length*), preventing triggers from propagating and creating records during this time.

The trigger blocking feature can be queried by the user for the number of blocking events Additionally, the information is included in the record header and represent the gate counter's value at the time the record was triggered. Combining the timestamp reset and trigger blocking features allow the user to keep track of the number of times the timestamp has been reset. In this way, the data from two batches separated by one or several resets of the timestamp are able to be related to each other in time.

Note

The timestamp reset and trigger blocking features may be combined to make the gate counter count the number of timestamp resets.

8 Application examples

The final chapter in this application note is a brief on how pulse detection is used in different applications. It is made in humbleness, keeping in mind that the authors' area of expertise is none of the applications mentioned. The examples are listed below.

Example	**Sec**
Flight time coincidence	8.1.1
Fusion plasma neutron emission spectrometry	8.1.2
Time-of-flight mass spectrometry (TOF MS)	8.2.1
LiDAR	8.2.2
System verification	8.3
Monitoring the experiment	8.4
Detecting faliure in an experiment	8.5

8.1 Examples of data-driven detection of random processes

In data-driven pulse detection it is the data itself that defines when the data collection starts. The end of the collection varies with the different collection modes (Sec. 5.3) and the record length can be either variable or fixed (Sec. 5.1).

8.1.1 Flight time coincidence

To measure flight time of particles from one array of detectors to another, only particles hitting both arrays are of interest. The coincidence trigger is then the first step of discrimination (Sec. 3.2). Set up the coincidence trigger such that if the first array triggers (channel A or B), then accept triggers the second array (channel C or D), see (14) and Figure 41. The symbol T represents a detected pulse. This would result in that pulses are stored from channel C or D only if a pulse was also detected on channel A or B. The timestamp of each pulse detected on channel C and D represent one particle flying between the detector arrays. To find the flight time, look up the corresponding pulse from channel A or B, which has a timestamp within the coincidence window and pair it with the first pulse.

$$\begin{bmatrix} Mask_A \\ Mask_B \\ Mask_C \\ Mask_D \end{bmatrix} = \begin{bmatrix} Mask_{AD} & Mask_{AC} & Mask_{AB} & Mask_{AA} \\ Mask_{BD} & Mask_{BC} & Mask_{BB} & Mask_{BA} \\ Mask_{CD} & Mask_{CC} & Mask_{CB} & Mask_{CA} \\ Mask_{DD} & Mask_{DC} & Mask_{DB} & Mask_{DA} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{14}$$

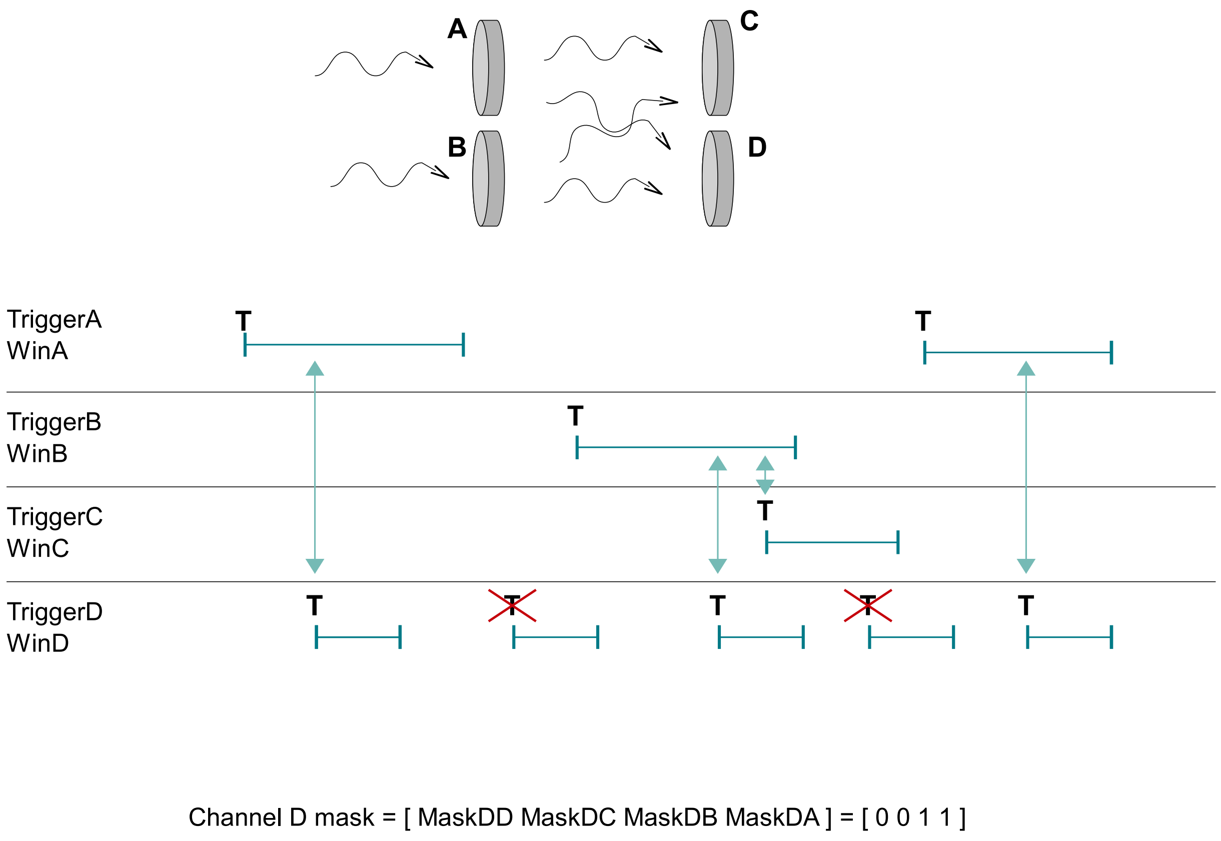

A
B
C
D
TriggerA
WinA
TriggerB
WinB
TriggerC
WinC
TriggerD
WinD
T
Channel D mask = [MaskDD MaskDC MaskDB MaskDA] = [0 0 1 1]

Figure 41: Coincidence mask example.

8.1.2 Fusion plasma neutron emission spectrometry

In fusion research, time-of-flight spectrometry (TOF) is a common technique for study of energy distribution of the released neutrons. The neutrons' energy distributions carry important information on what is going on inside the reactor. The TOF technique measures the time it takes for a neutron to travel between two detector regions (S1 and S2). Some of the neutrons detected in S1 (located to be reached by a direct fusion neutron stream trough a tiny 'hole' in the reactor), will also reach S2 a little later. The start detectors (S1) has typically ten times more activity compared to the stop detectors.

Using the Coincidence feature described in Sec. 8.1.1, where channel A and B act as start detectors and channel C and D as stop detectors, this will give rise to pairs of neutron travel times (coincidences). From the pairs, the flight times are measured. Further, from the timing measurements, spectra can be constructed and energy distributions be derived. Multi-parameter analysis (Sec. 4) on the correlated time and energy data is possible.

The measurements can be aligned in time with the reactor discharges using a common trigger distribution as described in Section 7.2. Further, since it is a multi-channel system with a number of both S1 and S2 detectors, a multi-board system might be needed. The boards can be synchronized in time as described in Section 7.3. The zero suppression and real-time analysis allows for a hight pulse detection rate without overloading the host interface or its storing capacity.

8.2 Scheduled detection examples

Scheduled pulse detection is pulse detection with a detection window that is *not* triggerd by data but with a global trigger source as described in Section 7. The duty cycle of a typical detection window in scheduled pulse detection is 90% or more, which implies data generation at about 8-20 GByte/s. Such high data rates are for most systems too much for the host interface to handle. Zero suppression and real-time pulse analysis (Sec. 4) are therefore fundamental prerequisites in scheduled pulse detection.

As opposed to data-driven detection, scheduled detection cannot guarantee that pulses are detected during the detection window. Especially not during system setup, before parameters are properly tuned. Lateny control (Sec. 6) is therefore of great help, enuring a response from the system irrespective of the degree of pulse activity. Examples of scheduled pulse detection are TOF MS and LiDAR.

8.2.1 Time-of-flight mass spectrometry

Time-of-flight mass spectrometry (TOF MS) is a method of mass spectrometry in which an ions mass-to-charge ratio is determined via a time of flight measurement. This is an example where a scheduled pulse analysis scheme is suitable according to:

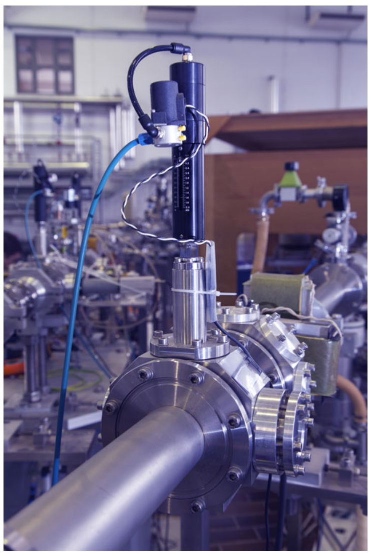


Figure 42: A time-of-flight mass spectrometer.

- Each time the TOF is triggered, a detection window is opened.
- The detection window lasts for the maximum flight time in the flight tube.
- By resetting the header timestamp (Sec. 5.4) at each TOF trigger, the gate counter in the Trigger blocking engine (Sec. 7.4 will keep track of the number of detection windows. The Trigger blocking is run in mode 'Window' and the gate counter number (which is the same for all pulses within one detection window) is found in the record header. The gate counter can also be used to organize the TOF triggers chronologically.

  The record header timestamps and the peak timestamps hold information on timing *within* a detection window:

  Collection Mode 1 (metadata channel) In data collection Mode 1 (Sec. 5.3.2), a single record is generated per detection window, with one metadata package per detected pulse. The record header timestamp will always be zero, but the peak timestamps relate to the start of the detection window and can be used to organize pulses within the detection window. See the metadata channel (Pulse charachteristics channel) in Fig. 33

  Collection Mode 3 (raw data) In data collection Mode 3 (Sec. 5.3.4), there is typically one record per pulse, thus more than one record per detection window. The record header timestamps relate to the start of the detection window and can be used for chronological sorting. A relevant example is shown in Fig. 35.

- Instead of resetting the header timstamp each detection window, the user could choose to reset it just once, at the first TOF trigger. The header timestamps can then be used to sort detection windows and the peak timestamps will (as above) hold the timing information within the window. This works well for metadata channels (Mode 1), but not in Mode 3, since the reference to each detection window trigger will be lost.

- The real-time pulse analysis (Sec. 4) will minimize the data processing in the PC.
- The raw data channel is available during development for verifying the pulse analysis.

The pulses in a mass spec application can be short-lived with minimal energy. To capture these pulses, the –FWPD is tailored as:

- DBS stabilizes the baseline and minimize pattern noise
- Because of DBS, the threshold can be set tight to the noise floor for high sensitivity
- Set the reset hysteresis to 1 to capture the smallest pulses.
- Since the edges of the pulses are very sharp, set the reset-arm-hysteresis trigger-arm-hysteresis to 1 for maximum sensitivity.

This set-up makes it possible to detect pulses with amplitude 1 LSB and width 1 sample, as illustrated in Fig. 43.

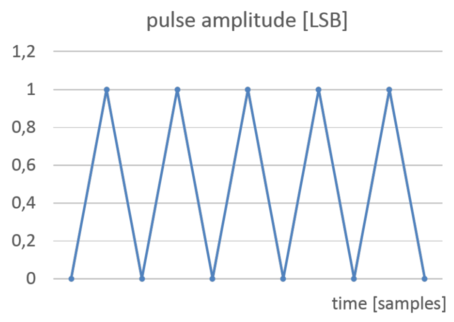


Figure 43: Detectable pulse train with settings for TOF MS.

8.2.2 LiDAR

Lidar (also called LIDAR, LiDAR, and LADAR) is a surveying method that measures distance to a target by illuminating the target with pulsed laser light and measuring the reflected pulses with a sensor (photon detection).

For this applocation, the detection window is of great use (collection Mode 3 (Sec. 5.3.4)). At each laser pulse, a detection window can be triggered, and the (fixed) length of the window is set after the maximum measurement distance. A timestamp reset (Sec. 5.4) at each detection window trigger, allows for intuitive chronological sorting of events originating from the same laser pulse. Further, with Trigger blocking in 'window' mode (Sec. 7.4) that shares the same external source as the detection window trigger, the gate counter enables detection windows to be related in time. The gate counter, found in the record header, then keeps track of the number of detection windows (since it coincides with the number of timstamp resets) and in which order they came. Finally, the shape of each detected pulse can be studied off-line with application specific analyze.

8.3 System verification using the GUI

It is challenging to get started with a new application and to find suitable parameters for the automated pulse detection algorithm. To support the integration of the firmware, there is example code and a GUI with basic functionality. The purpose of the GUI is to support the user in finding a first set of parameters. The GUI can do an automatic set-up of the pulse detection with a proposed set of parameters. This requires clearly visible pulses. The GUI also plots the result. When hitting the `Run selected` (or `Run all`) button in the GUI a python example code is executed. The command line for calling this example code is available as text. Use the command line together with the example to transfer your setting into the final application software.

The data multiplexer (Section 5.5) on ADQ14 allows the user to select which data stream to input to the four individual analysis and characterization engines. Combining different data collection modes to operate on the input same channel makes it possible to verify the system by observing the raw pules data together with its corresponding metadata. On ADQ7, this functionality is available without the multiplexer, due to the dedicated metadata channels.

For demonstration purpose, a simple verification example is given. Channel A on an ADQ14AC-2X unit with a sample rate of 2GSPS is connected to a pulse generator[14]. The board is controlled via the FWPD Pulse Characterization GUI, included in the release. The settings on the pulse generator and in the GUI are shown in Table 2 and the hierarchy of the files used are given in Fig. 47.

Figure 44 shows the four data multiplexers of which two are active. Both of them forward data from input channel A, to output channels A and B. Output channel A is in Mode 0 (the fixed record lengths works as a detection window), and output channel B operates in Mode 1 (metadata mode).

A screen shot of tab `Channel A` of the GUI, is shown in Fig. 45. The data multiplexing is controlled in the `Input multiplexer` tab up to the right. A data collection is triggered by pressing the buttom `Run selected`. The 64 000 samples of raw data are collected in one record and sent to the host and is shown in the lower left corner. Note that since the board has a AC-coupled front-end, the input DC level will not be preserved. The start of the pulse is after 1 1000 samples, as a result of the LEW parameter setting. Collecting 64 000 samples at 2 GSPS gives a sequence in time for

$$\frac{64 \cdot 10^3}{2 \cdot 10^{-9}} = 32 \cdot 10^{-6} = 32\mu s,$$

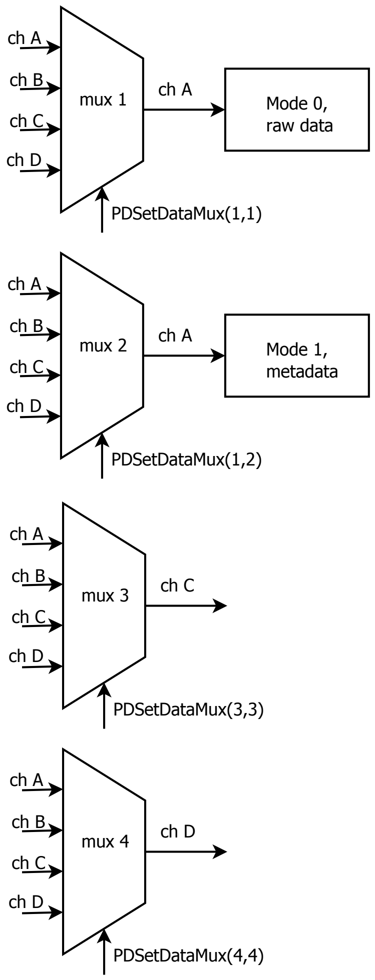


Figure 44: Control of ADQ14 data multiplexer for system verification.

thus four pulses with a period of 10μs fits within the window.

A screen shot of tab `Channel B` is shown in Fig. 46. Here, the waveform plot down to the left is not of much help, it is the metadata vaules in binary form for the four pulses and is not straight-forward to

[14]HEWLETT PACKAGE 8131A 500 MHz Pulse Generator.

interpret. Instead a python script, `parse_metadata.py`, is used to parse the data to decimal numbers. The python script is included in the release and the output from the script is shown in Fig. 48.

Table 2: Verification example settings.

parameter	Value
FWPD Pulse Charachterization GUI	
trigger level	1 000
record length (fixed)	64 000
LEW (pre-trigger length)	1 000
Pulse generator	
pulse width	500ns
period	10μs
high level	0.10 V
low level	-0.10 V

The data from output channel A is plotted with `plot.py` and then it is straight-forward to verify the peak values from the metadata as well as the pulse widths. A pulse width of 994 samples at 2GSPS gives a pulse with in time of

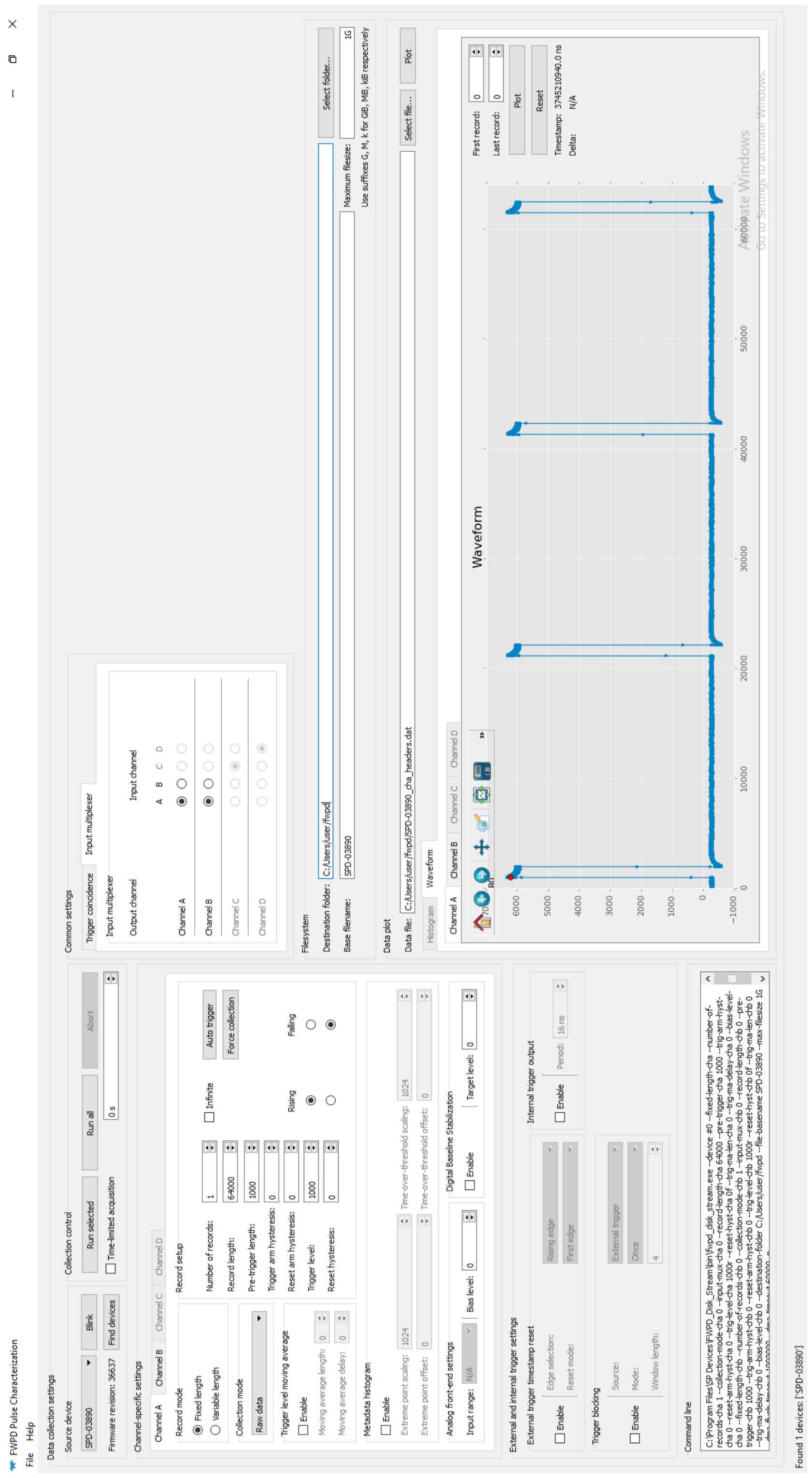
$$\frac{994}{2 \cdot 10^9} = 497 \cdot 10^{-9} \approx 500ns$$

.

which is well consistent with the signal generator setting in Table 2. As seen in the print-out from the `parse_metadata.py`, six metadata packages are found, instead of four. In addition to the metadata packages there are two padding packages. The reson for that is *not* that padding was enabled and the minimum framelength (Sec. 5.2) was not reached. No, these two packages, althout identical to padding packages from the padding engine, have another origin than the padding engine. It is a tiny detail in firmware, and here follows a brief explanation: Firstly, while creating a metadata record, *two* packages are packed on each data-valid pulse. Secondly, the collection engine continue until it gets a stop signal, and on the stop signal, it will always pack the two last packages. If there are no more packages to pack, it will pack two packages with zero-data. These zero-data packages are identical to the padding packages and will be treaded as such (that is rejected) by the user.

8.4 Monitoring the experiment

The pulse record contains a header with information about the record and the pulse(s). The header also contains information about the degree of filling in the digitizer FIFO. If the degree of filling peaks at high



Data collection settings

Source device: SPD-03890
Firmware revision: 36637

Collection control: [ ] Time-limited acquisition 0 s

Channel-specific settings — Channel A | Channel B | Channel C | Channel D

Record mode: [✓ Fixed length] Variable length

Collection mode: Raw data

Trigger level moving average: [ ] Enable
Moving average length: 0
Moving average delay: 0

Record setup:
Number of records: 1 [ ] Infinite
Record length: 64000
Pre-trigger length: 1000
Trigger arm hysteresis: 0
Reset arm hysteresis: 0
Trigger level: 1000 [✓ Rising] Falling
Reset hysteresis: 0 Rising [✓ Falling]

Metadata histogram: [ ] Enable
Extreme point scaling: 1024 Time-over-threshold scaling: 1024
Extreme point offset: 0 Time-over-threshold offset: 0

Analog front-end settings: Input range: N/A Bias level: 0
Digital Baseline Stabilization: [ ] Enable Target level: 0

External and internal trigger settings

External trigger timestamp reset: [ ] Enable Edge selection: Rising edge Reset mode: First edge

Internal trigger output: [ ] Enable Period: 16 ns

Trigger blocking: [ ] Enable Source: External trigger Mode: Once Window length: 4

Common settings — Trigger coincidence | Input multiplexer

Output channel	A	B	C	D
Channel A	●	○	○	○
Channel B	●	○	○	○
Channel C	○	○	●	○
Channel D	○	○	○	●

Filesystem
Destination folder: C:/Users/user/fwpd
Base filename: SPD-03890
Maximum filesize: 1G
Use suffixes G, M, k for GiB, MiB, kiB respectively

Data plot
Data file: C:/Users/user/fwpd/SPD-03890_cha_headers.dat

First record: 0
Last record: 0
Timestamp: 3745210940.0 ns
Delta: N/A

Command line

```
C:\Program Files\SP Devices\FWPD_Disk_Stream\bin\fwpd_disk_stream.exe --device #0 --fixed-length-cha --number-of-
records-cha 1 --collection-mode-cha 0 --input-mux-cha 0 --record-length-cha 64000 --pre-trigger-cha 1000 --trig-arm-hyst-
cha 0 --reset-arm-hyst-cha 0 --trig-level-cha 1000r --reset-hyst-cha 0f --trig-ma-len-cha 0 --trig-ma-delay-cha 0 --bias-level-
cha 0 --fixed-length-chb --number-of-records-chb 0 --collection-mode-chb 1 --input-mux-chb 0 --record-length-chb 0 --pre-
trigger-chb 1000 --trig-arm-hyst-chb 0 --reset-arm-hyst-chb 0 --trig-level-chb 1000r --reset-hyst-chb 0f --trig-ma-len-chb 0
--trig-ma-delay-chb 0 --bias-level-chb 0 --destination-folder C:/Users/user/fwpd --file-basename SPD-03890 --max-filesize 1G
```

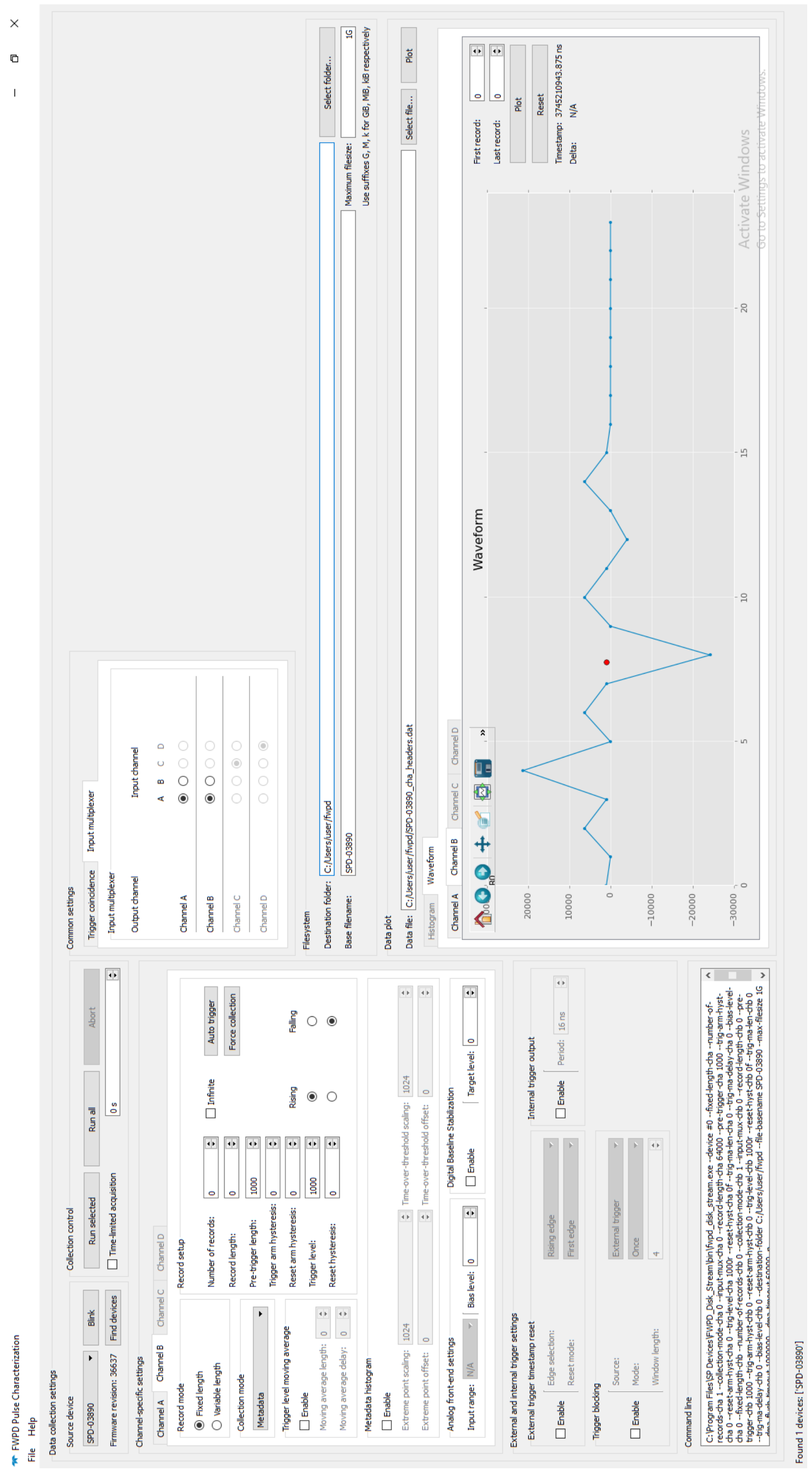
Found 1 devices: ['SPD-03890']

Figure 45: Screen shot of GUI, tab Channel A. A fixed length record with four pulses is collected.

levels, there is a risk of overflow and lost data. By keeping an eye on the FIFO fill factor, the experiment can be controlled, for example by reducing the system trigger frequency.

8.5 Detecting failure in an experiment

If the pulses suddenly are too long or intense, these failures can directly be detected in the user's software. For example, when the record length increase above a certain limit, a stop command can be sent to the digitizer and the system can be restarted. Since this stop is defined in user software, the conditions for fault detection are fully under the user's control.


FWPD Pulse Characterization

File Help

Data collection settings

Source device: SPD-03890 — Blink
Firmware revision: 36637 — Find devices

Collection control: Run selected — Run all — Abort
☐ Time-limited acquisition: 0 s

Channel-specific settings: Channel A | Channel B | Channel C | Channel D

Record mode: (●) Fixed length ( ) Variable length

Collection mode: Metadata

Trigger level moving average: ☐ Enable
Moving average length: 0
Moving average delay: 0

Record setup:
Number of records: 0 ☐ Infinite — Auto trigger
Record length: 0 — Force collection
Pre-trigger length: 1000
Trigger arm hysteresis: 0
Reset arm hysteresis: 0

		Rising	Falling
Trigger level:	1000	●	○
Reset hysteresis:	0	○	●

Metadata histogram: ☐ Enable
Extreme point scaling: 1024 — Time-over-threshold scaling: 1024
Extreme point offset: 0 — Time-over-threshold offset: 0

Analog front-end settings: Input range: N/A — Bias level: 0
Digital Baseline Stabilization: ☐ Enable — Target level: 0

External and internal trigger settings
External trigger timestamp reset: ☐ Enable — Edge selection: Rising edge — Reset mode: First edge
Internal trigger output: ☐ Enable — Period: 16 ns
Trigger blocking: ☐ Enable — Source: External trigger — Mode: Once — Window length: 4

Command line

```
C:\Program Files\SP Devices\FWPD_Disk_Stream\bin\fwpd_disk_stream.exe --device #0 --fixed-length-cha --number-of-
records-cha 1 --collection-mode-cha 0 --input-mux-cha 0 --record-length-cha 64000 --pre-trigger-cha 1000 --trig-arm-hyst-
cha 0 --reset-arm-hyst-cha 0 --trig-level-cha 1000r --reset-hyst-cha 0f --trig-ma-len-cha 0 --trig-ma-delay-cha 0 --bias-level-
cha 0 --fixed-length-chb --number-of-records-chb 0 --collection-mode-chb 1 --input-mux-chb 0 --record-length-chb 0 --pre-
trigger-chb 1000 --trig-arm-hyst-chb 0 --reset-arm-hyst-chb 0 --trig-level-chb 1000r --reset-hyst-chb 0f --trig-ma-len-chb 0
--trig-ma-delay-chb 0 --bias-level-chb 0 --destination-folder C:/Users/user/fwpd --file-basename SPD-03890 --max-filesize 1G
```

Found 1 devices: ['SPD-03890']

Common settings: Trigger coincidence | Input multiplexer

Input multiplexer

Output channel	A	B	C	D
Channel A	●	○	○	○
Channel B	●	○	○	○
Channel C	○	○	●	○
Channel D	○	○	○	●

Filesystem
Destination folder: C:/Users/user/fwpd — Select folder...
Base filename: SPD-03890 — Maximum filesize: 1G
Use suffixes G, M, k for GiB, MiB, kiB respectively

Data plot
Data file: C:/Users/user/fwpd/SPD-03890_cha_headers.dat — Select file... — Plot
Histogram | Waveform
Channel A | Channel B | Channel C | Channel D

First record: 0
Last record: 0
Plot
Reset
Timestamp: 3745210943.875 ns
Delta: N/A

Figure 46: Screen shot of GUI, tab Channel B. Metadata of the raw pulse data on channel A is calculated in real-time and output on channel B.

```
<Path to installation directory>/
  └ SP Devices/
      ├ C_examples/
      │   └ ADQAPI_FWPD_example/
      │       ├ parse_metadata.py        Parse and print metadata.
      │       └ plot.py                  Plot raw data.
      └ FWPD_Pulse_Char/                 GUI location, from a separate installer.
          └ fwpd_pulse_char.exe
```

Figure 47: Hierarchy of files used in Example 8.3.

```
C://Users/user/fwpd>python parse_metadata.py
Parsing data file ./SPD-03890_chb_00000.dat
Found 1 record headers
Found 6 metadata packets
Record timestamp 0x6f9db11ff
Record number 0
**** Parsing packet 0, record 0 ****
Extreme value timestamp: 1024
Extreme value: 6320
Time over threshold: 994
**************************
**** Parsing packet 1, record 0 ****
Extreme value timestamp: 21212
Extreme value: 6316
Time over threshold: 994
**************************
**** Parsing packet 2, record 0 ****
Extreme value timestamp: 41364
Extreme value: 6312
Time over threshold: 994
**************************
**** Parsing packet 3, record 0 ****
Extreme value timestamp: 61554
Extreme value: 6324
Time over threshold: 994
**************************
2 padding packets
```

Figure 48: Output from `parse_metadata.py`.

Worldwide Sales and Technical Support
www.teledyne-spdevices.com

Teledyne SP Devices Corporate Headquarters
Teknikringen 6
SE-583 30 Linköping
Sweden
Phone: +46 (0)13 645 0600
Fax: +46 (0)13 991 3044
Email: info@teledyne-spdevices.com