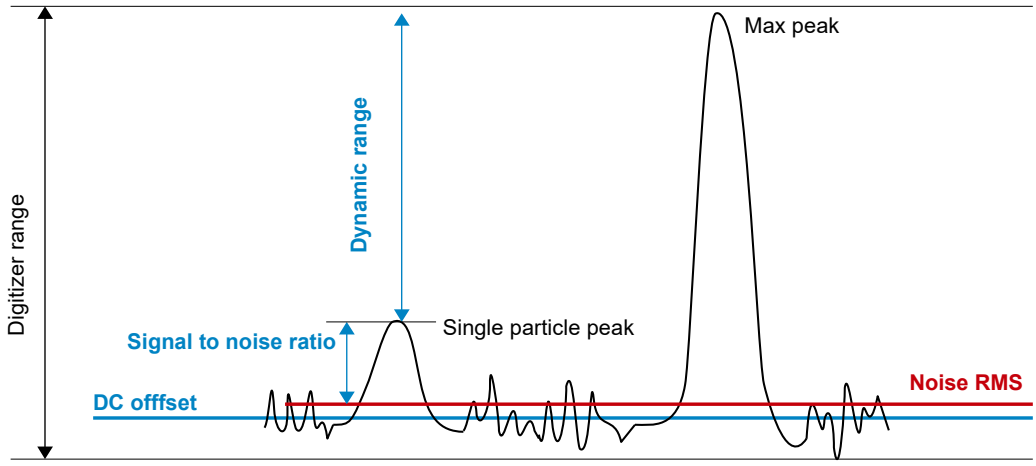
TELEDYNE SP DEVICES
Everywhereyoulook™

Advanced time-domain measurements

Application note



Author(s): Teledyne SP Devices
Document ID: 18-2104
Classification: Public
Revision: C
Print date: 2018-12-20

Contents

1 Definitions & Abbreviations **2**

2 Introduction **3**
2.1 Block diagram . . . 3

3 Triggering a waveform **5**

4 Tailor input signal capturing **6**
4.1 Overview . . . 7
4.2 Unipolar signal conditioning . . . 8
4.3 Pulse input data . . . 8
4.3.1 Track slow baseline variations . . . 9
4.3.2 Suppression of interleaving noise . . . 9
4.4 Limited signal bandwidth . . . 10

5 Waveform averaging **11**
5.1 Suppression of random noise with WFA . . . 12
5.1.1 Some formulas . . . 12
5.2 The accumulation engine . . . 13

6 Suppression of systematic noise **14**
6.1 Three methods . . . 14

7 Finding rare pulses **18**
7.1 Threshold operation . . . 18
7.2 Advanced threshold operation with filter . . . 18
7.3 Threshold example . . . 21

8 Data read-out to user application – ADQ7 only **22**
8.1 The ATD data struct . . . 22
8.2 Seamless streaming with queue interface . . . 23
8.3 Dead-time free acquisition and duty cycle trade-off . . . 25
8.4 System robustness and overflow . . . 26
8.4.1 Overflow in the physical interface . . . 26
8.4.2 Overflow in the queue interface . . . 26
8.4.3 Overflow behavior . . . 27

9 Application examples **28**
9.1 Measuring a changing process . . . 28
9.2 Live updates of subtotals . . . 28
9.3 Semiconductor Automated Test Equipment (ATE) . . . 30

1 Definitions & Abbreviations

Table 1 lists the definitions and abbreviations used in this document and provides an explanation for each entry. See also Glossary on the website, under the Support tab.

Table 1: Definitions and abbreviations used in this document.

Term	Explanation
ADC	Analog-to-digital converter
ADQ	Common name for ADQ7 and ADQ14
ADQ14	TSPD's 14-bit, 2 GSPS digitizer platform, 1-4 channels
ADQ7	TSPD's 14-bit, 10 GSPS digitizer platform, 1-2 channels
ADQAPI	Application programming interface for ADQ
AFE	Analog front-end
ATE	Automated test equipment
DC offset	Analog DC level which is added to the input signal inside the digitizer (to 'move it vertically')
DBS	TSPD's proprietary IP Digital Baseline Stabilizer for stable and smooth DC level
DRAM	Dynamic random access memory
FIR	Finite impulse response
FW	Firmware (digitizer feature set)
FWATD	Advanced time-domain firmware option
GSPS	Giga (10^9) samples per second
N_A	Number of accumulations
N_R	Number of accumulation repetitions (number of N_A)
Q.E.D.	Quod erat demonstrandum" meaning "what was to be demonstrated"
Record	A batch of consecutive samples acquired at a trigger event
R_L	Record length
SDK	Software development kit
SNR	Signal-to-noise ratio
Trigger	A real-time event starting the acquisition of a record
User space record	Record presented to end user, carrying N_A accumulated waveforms
Waveform	Analog signal with a distribution in time, digitized into a record
WFA	Waveform accumulation/accumulator

2 Introduction

The firmware option FWATD (advanced time domain) for ADQ is designed for *extreme dynamic range*. With a high dynamic range, a rarely occurring weak signal can coexist with a strong (in the meaning high amplitude) one. Several steps of noise suppression are offered and can be combined to reach the required performance. Typical applications that includes advanced time-domain measurements and benefit from extreme dynamic ranges are

- Scientific instruments
- Time-of-flight
- Mass spectrometry
- Electron paramagnetic resonance
- Particle physics experiments
- Test & measurements
- Distributed fiber sensing

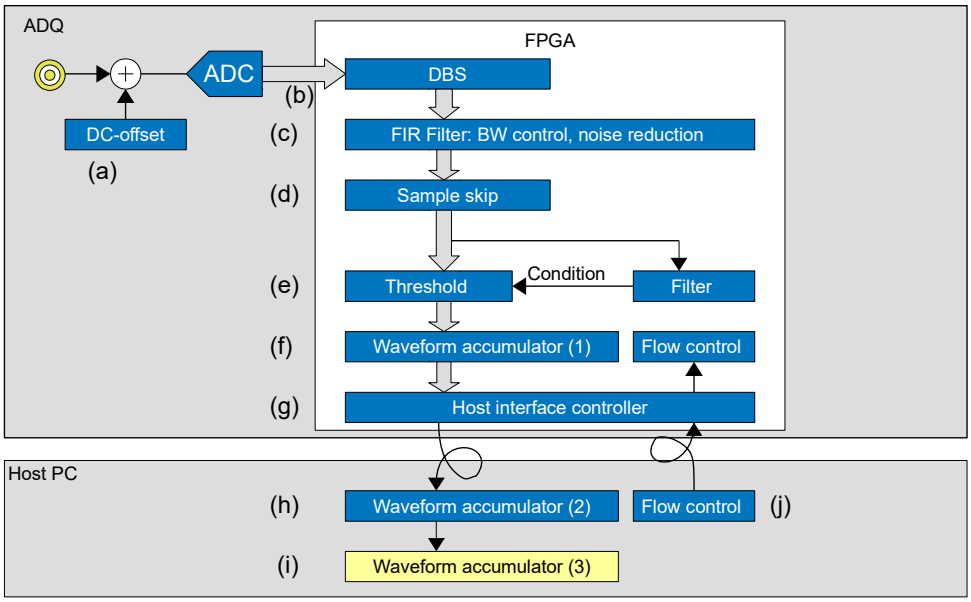
2.1 Block diagram

To achieve extreme dynamic range, several parameters need to be taken into account. The firmware contains four levels of noise suppression; baseline stabilization, linear filtering, non-linear threshold, and waveform averaging. Three methods that focus specifically on systematic noise (pattern noise) are also offered. FWATD can be seen as a toolbox with the above mentioned noise suppression methods. From there, the application at hand can choose suitable tools to obtain the required dynamic range. Some application examples are provided in Section 9.

Note

FWATD is a toolbox, where different applications will require different tools.

In a multi-channel ADQ, there is one FWATD unit per channel. They operate synchronously with the same settings on waveform size and number of accumulations. The filter and threshold parameters are however set individually for each channel. The block diagram of the ADQ-FWATD is shown in Fig. 1.



a. To fully use the range of the ADC, an analog DC offset can be added to the signal. This effectively doubles the dynamic range for unipolar pulses (Section 4.2).

b. The Digital Baseline Stabilizer, DBS, is a Teledyne SP Devices' proprietary algorithm for baseline stabilization in pulse data systems. The DBS output is a controlled baseline level set at a user-defined value with 22 bits precision (Section 4.3.1).

c. General data path FIR filter for limiting the bandwidth and reducing wideband noise (Section 4.4).

d. Sample skip for data reduction or full-fledged decimation with (c) as anti-aliasing filter (Section 4.4).

e. The advanced threshold operation is a non-linear noise suppression which discriminates samples below a user-defined level. A filter allows for sophisticated promotion of sample sequences with a certain shape, even if the pulse amplitude has the same order of magnitude as the noise (Section 7).

f. Waveform averaging accumulates a large number of waveforms for random noise suppression by repeated measurements (Section 5). Note that the accumulator adds the waveforms; division is left to be performed by the user in the PC.

g. Seamless streaming to host gives near dead-time free acquisition (Section 8).

h. To enable longer waveforms, R_L, and longer accumulation, N_A, a second part of the accumulation can be done in the ADQAPI. A partitioning algorithm decides if accumulation in the ADQAPI is needed for the current use case (Section 5.2).

i. If needed, the user can continue the averaging in the user applications using a longer format, e.g `int64`.

j. Data transfer to host PC over USB3.0 or PCIe.

Figure 1: ADQ-FWATD architecture with descriptions of each block.

3 Triggering a waveform

A waveform is a sequential set of data following a trigger event. This data set is also called record throughout this document. The available trigger sources for triggering a waveform are described below.

Note

A waveform is a record which is a batch of consecutive samples acquired at a trigger event.

Internal trigger source A trigger signal is generated internally in the digitizer using the internal reference clock. By default, the internal trigger is free running (see also Section 6). The internal trigger can be output on connector TRIG, as master external equipment.

External trigger source The device is triggered by an external signal connected to the connector TRIG on the front panel. The pattern noise suppression trigger module in Section 6.1 is an example of an external trigger source.

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.

Level trigger The amplitude of the input signal decides when the system should trigger. The triggering is thus data-driven.

A system using waveform averaging is preferably a repeated measurement. Then the trigger is either the *external trigger* starting the acquisition from an external equipment, or the *internal trigger* where the ADQ controls the external equipment via the connector TRIG. There are also applications where certain pulse shapes are of interest. A data-driven *level trigger* can be used to trigger several times on the repeated pulse.

The waveform is always of fixed (user-defined) length, R_L, but the user has some freedom to choose when the waveform should start. The start of the waveform can be moved forth and back in time, relative to the trigger event. The parameter that sets the number of *pre-trigger* samples is available for applications where the samples just *before* the trigger event are of interest. For applications where the trigger comes too early, another parameter is available to control the *trigger delay*. Pre-trigger and trigger delay are illustrated in Fig. 2. Note that neither pre-trigger nor trigger delay affect the record length.

Note

Pre-trigger is data before the trigger event. Trigger delay is waiting before starting acquisition. Neither of them affects R_L.

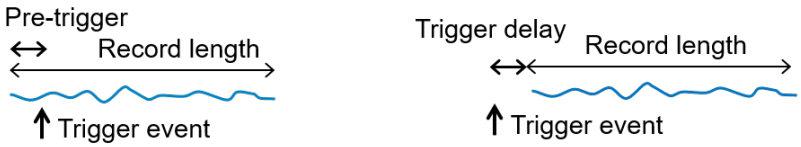


Figure 2: Illustration of pre-trigger and trigger delay. The user has some freedom to choose the start sample of the waveform.

The time between two records is called *dead time*. The dead time is limited by the rearm time (the time period from the last sample of a waveform until the system is ready to accept a new trigger), but also by the data transfer speed to the host PC. The dead time and rearm time is illustrated in Fig. 3. The minimum dead time has an order of magnitude of tens of nanoseconds; exact values are found in the ADQ-FWATD s ([1] and [2]).

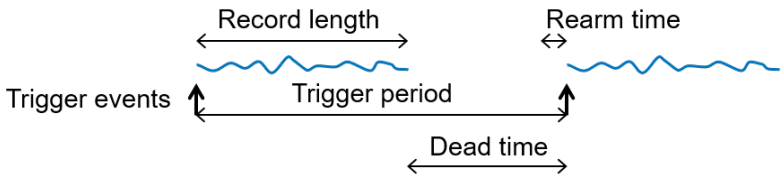


Figure 3: Illustration of trigger period, rearm time, and dead time.

In Fig. 3, the trigger period is also shown. If Trigger period $< R_L$, triggers occurring during recording will be ignored. The duty cycle can thereby be defined as in (1). The duty cycle is upper limited by the fixed rearm time.

$$\text{Duty cycle} = \frac{\text{Record length}}{\text{Trigger period}} \tag{1}$$

4 Tailor input signal capturing

In many applications some general characteristics of the input signal are known beforehand. This information can be used to configure the digitizer and thereby optimize the data measurement performance. In this chapter, three types of information that can be used to tailor the signal capturing to the application at hand will be discussed. The characteristics come from both the time-domain and the frequency domain. Some basic skills in transform theory is therefore recommended.

Note

Advance information about the input signal can be used to optimize the measurement performance.

The three information types are listed below and will be described in detail in the subsequent sections. But before that (Section 4.1), an overview of the signal capturing to clarify the context.

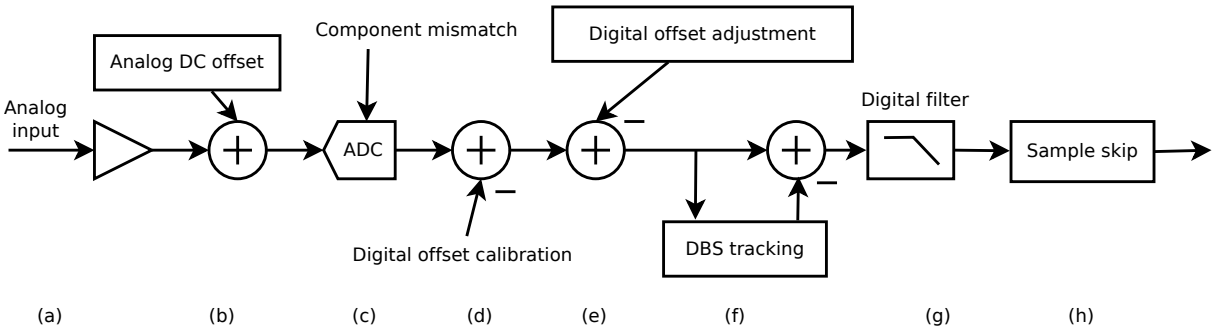
Unipolar signal The signal level is constrained to values either between approximately zero volts and some arbitrary positive voltage, or between (approximately) zero volts and some negative voltage (Section 4.2).

Pulse data signal The signal is approximately a reference-level (baseline) with time-domain pulses (Section 4.3).

Limited signal bandwidth The signal's frequency content is limited (Section 4.4).

4.1 Overview

Analog DC offset adjustment is just one of several DC level contributors in the digitizer. An overview of the different contributions is shown in Fig. 4. Note that the contributions can be both positive and negative, resulting in a increase or decrease of the DC level.



a. The analog signal from the sensor can contain a DC level.

b. A user controlled DC offset can be added to the analog signal at the analog input of the ADQ (Section 4.2).

c. The analog circuits add unwanted per-ADC-core offsets due to component mismatch in the interleaved ADCs.

d. The factory calibration procedure adds static per-core digital offset correction.

e. The user may add an additional digital offset to the factory calibrated values.

f. The baseline is stabilized with DBS and set to a user-defined value (Section 4.3).

g. Bandwidth-limiting digital filter for noise attenuation (Section 4.4). Named `UserLogicFilter` in the API [3].

h. Sample skip option for data reduction or part of a decimation (Section 4.4).

Figure 4: Overview of the signal capturing including AFE and digital signal conditioning with all contributions to the DC level and the position of the noise reducing digital filter, as well as sample skip. Sample skip does not affect the offset level, but is shown here since it is part of the signal capturing.

4.2 Unipolar signal conditioning

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. 6. 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. 6 (a)[1]. To fully use the symmetrical input range of the ADC, an analog DC offset is added to the signal, Fig. 6 (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. 6 (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. 6 (d) and (e).

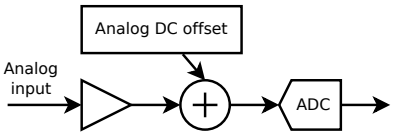


Figure 5: Zoom in Fig. 4 on the analog DC offset control.

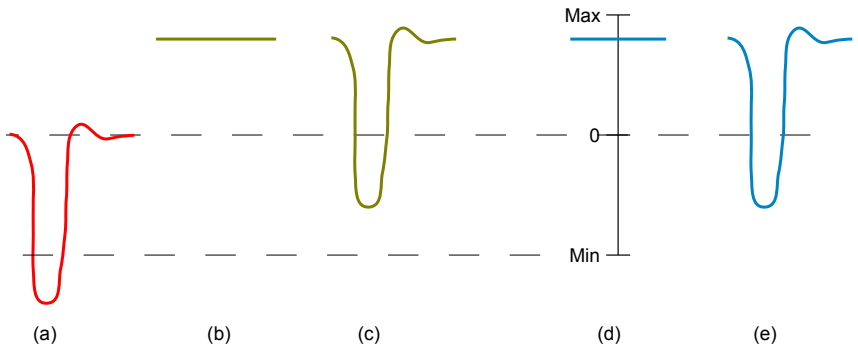


Figure 6: Analog input DC offset adjustment.

4.3 Pulse input data

The digitizer can be tailored for *pulse* input data to improve the measurement performance. For pulse data applications, a stable baseline is as critical as the pulses themselves. With large baseline variations, it becomes problematic to define what is a pulse and what is not. If not compensated for, aging, temperature variations, pattern noise from electrical components, etc, will reduce the possibilities to distinguish a weak signal from noise. This section describes how the fact that the signal has a *baseline* can be used to improve the digital signal quality.

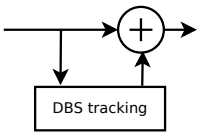


Figure 7: Zoom in Fig. 4 on DBS.

Note

A stable baseline can improve signal quality.

[1] 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.

4.3.1 Track slow baseline variations

Temperature and aging slowly change the behavior of electronics. Tracking and removing this kind of variations from the signal facilitate pulse focus. 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 user-controlled target value, `dc_target`, with 22 bits precision. The principle of operation is shown in Fig. 8. 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. 9. Fig. 9 (a) shows the baseline fluctuations and Fig. 9 (c) the stabilized baseline after compensation by DBS (b). Note that DBS act on the entire signal chain. If the system contains a temperature sensitive amplifier, these variations will also be corrected for. DBS can be disabled by the user if it is not needed.

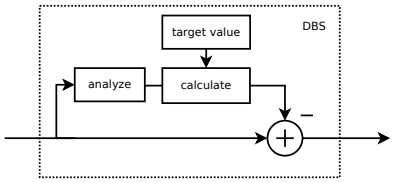


Figure 8: DBS principle of operation. The target value is set by the user via the variable `dc_target` [3].

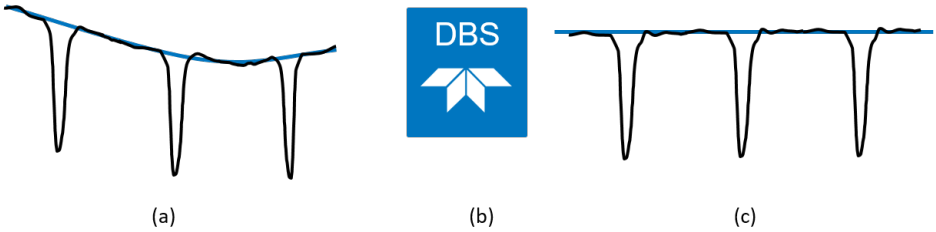


Figure 9: Baseline stabilizing with DBS.

4.3.2 Suppression of interleaving noise

As for most high-speed ADC components, one channel on the digitizer consists of several interleaved ADC cores. Interleaving is a commonly used technique for increasing sample rate of ADCs and digitizers. The interleaving takes place inside the die, as well as on the board level[2]. These cores are not 100% identical and therefore mismatch errors will show up as zig-zag pattern noise in the time-domain, and disturb the baseline. The order of magnitude of the offset mismatch is typically around 100 codes, thus it is a significant noise source (in particular when accumulating/averaging records), perhaps the limiting one, if not compensated for. To suppress these mismatches, the DBS IP takes into account the number of ADC cores. Fig. 10 (a) shows the analog input signal and Fig. 10 (b) an ADC with four interleaved cores. Fig. 10 (c) gives the output from each ADC core, illustrating the effects of component mismatch. The digital signal is restored (e) by letting DBS (d) estimate and correct for the baseline differences.

[2] In one-channel mode only.

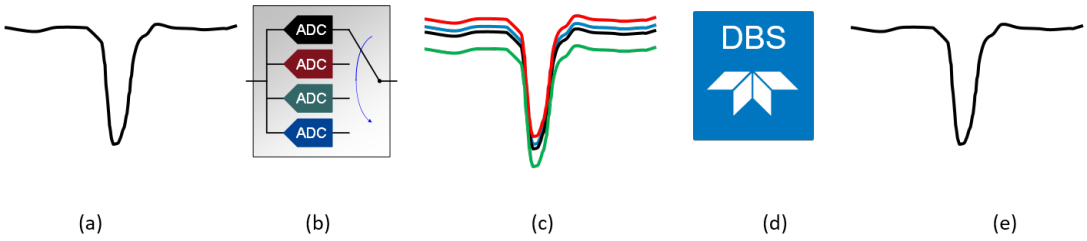


Figure 10: Correction for offset interleaving mismatch with DBS.

4.4 Limited signal bandwidth

The third useful input signal information is *limited signal bandwidth*. Input signals to an ADC need a limited bandwidth in order to reduce aliasing. The ADC on ADQ is therefore preceded by an anti-aliasing filter with about 3 GHz (ADQ7) bandwidth [4][3]. An example of an input signal in the frequency domain together with the analog anti-aliasing filter is given in Fig. 12 (a). The analog anti-aliasing filter attenuates all frequencies above 3 GHz, and thereby allows for noise reduction by means of a frequency selective digital filter.

Fig. 12 (b) shows the signal (also in the frequency domain) after sampling. The sampling maps the analog frequency range $[0, f_s/2]$ to $[0, \pi]$. Apart from the signal there is an approximately flat noise floor, covering the whole spectrum ($[0, \pi]$), originating mainly from the analog-to-digital conversion process.

Fig. 12 (c) illustrates the beauty of noise reduction with digital filtering. Although signal and noise are mixed in the time domain, they can (partly) be *separated* in the frequency domain. All noise above the signal bandwidth limit can be attenuated with the user-controlled digital filter, without affecting the wanted signal. The more limited in frequency the wanted signal is, the more can the digital filter improve SNR. The digital filter is implemented in User Logic 1 and is therefore referred to as the `UserLogicFilter` in [3].

An additional advantage that comes with bandwidth reduction is the possibility of sample skip without information loss (decimation). If the bandwidth is limited to $1/M$ of the digital frequency range $[0, \pi]$[4], the signal can be downsampled with a factor M. Keeping only every M^{th} sample reduces the computational complexity and relieves the requirements on the host transfer speed. Fig. 12 (d) illustrates sample skip with a factor $M = 2$. All star samples are kept and the circle samples are discarded. As a result, the amount of data is halved.

Sample skip can also be used independently of the FIR filter to

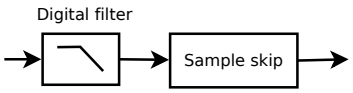


Figure 11: Zoom in Fig. 4 on bandwidth limiting filter and sample skip.

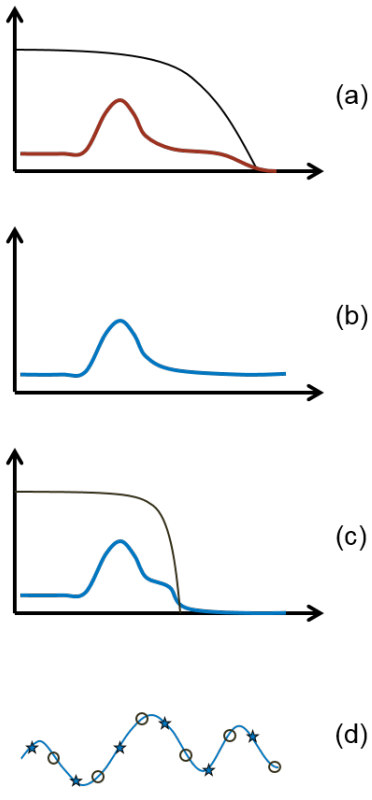


Figure 12: If the input signal has a limited bandwidth $< f_s/2$, a digital filter can be used to reduce white noise.

[3] For other ADQ families, refer their respective s (e.g. [5]).
[4] $[0, \pi/M]$ corresponds to the analog frequencies $[0, f_s/2/M]$

increase the record time duration. Without sample skip, the maximum waveform length corresponds to 200 μs, but by skipping for example every second sample, this time is doubled[5].

Example

The analog bandwidth of one-channel ADQ7 is limited by the analog anti-aliasing filter to 3 GHz. The noise from the ADC has a bandwidth of $f_s/2 = 5$ *GHz*. So the noise has higher bandwidth than the wanted signal. The digital filter can thus directly (no additional signal bandwidth restrictions) reduce the noise with $10\log(5/3) \approx 2.2$ [dB] without signal information loss.

Example

A bandwidth-limited input signal with frequency content up to 1.25 GHz is sampled with ADQ7-FWATD at 10 GSPS. Without information loss, it can be filtered through a digital filter with passband edge $\pi/4$ and decimated with a sample skip factor $M = 4$, resulting in a SNR improvement of 6 dB and a data reduction of a factor 4.

5 Waveform averaging

One common way of reducing random noise and increase sensitivity is to repeat a measurement several times and study the averaged signal. This method reduces the non-correlated noise and increases the correlated wanted signal. The method of waveform averaging results in large amount of measurement data. In order to reduce the load on the host computer, FWATD includes support for real-time waveform accumulation in the on-board FPGA.

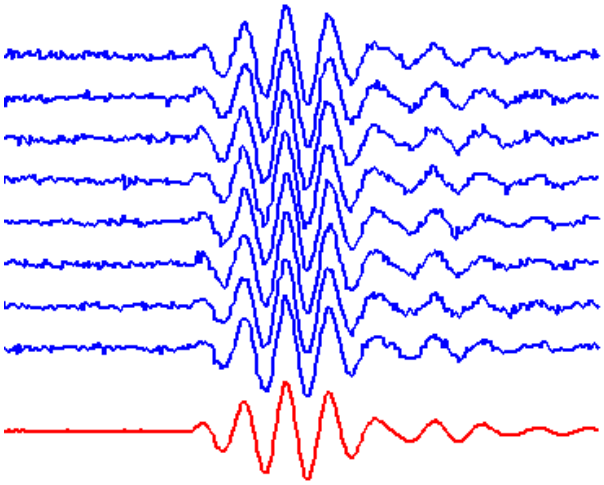


Figure 13: Illustration of waveform accumulation.

The waveform averaging (WFA) is an accumulator that adds waveforms to each other. The first sample in each waveform is added to the first sample of all other waveforms. An averaging function should by definition contain a division with the number of waveforms, N_A. However, this division is not suitable for FPGA implementation and is therefore left for the user to perform in software. The WFA concept is illustrated in Fig. 13. Generous waveform lengths with up to 2^{21} ($>$ 2 million) samples[6] are enabled.

[5] The price to pay for the increased time duration is reduced time resolution.
[6] ADQ7 in one-channel mode.

5.1 Suppression of random noise with WFA

The beauty of WFA is that when accumulating, the power of the signal grows faster than the random noise power. More precisely, reduction of random noise through averaging is proportional to the square root of the number of accumulations. The SNR is improved by $10\log_{10} N_A$, where N_A is the number of accumulations. For the interested reader, some theory of these statements is given in Section 5.1.1.

Note

Reduction of random noise through averaging is proportional to the square root of the number of accumulations.

5.1.1 Some formulas

The input signal can be seen as a sum of the wanted signal $s(n)$ and some random noise $e(n)$ as in (2).

$$x(n) = s(n) + e(n) \tag{2}$$

The wanted signal $s(n)$ is assumed to be the same in each waveform with an average power denoted as $P[s(n)]$. Accumulating (2) N_A times will then give (3).

$$N_A \cdot x(n) = N_A \cdot s(n) + e_1(n) + e_2(n) + \ldots e_N(n) \tag{3}$$

The random noise $e(n)$ varies stochastically and has a zero mean value. Further, each pair of samples of $e(n)$ is uncorrelated. The average power of such white noise is the same as its *variance*, σ^2, and thereby the power of the accumulated wanted signal and the accumulated noise can be written as (4) and (5), respectively.

$$P[\text{accumulated wanted signal}] = P[N_A \cdot s(n)] = N_A^2 \cdot P[s(n)] \tag{4}$$

$$P[\text{accumulated noise}] = \sigma_{\text{ACC}} = \sum_{i=1}^{N_A} \sigma_i^2 = N_A \cdot \sigma_{\text{noise}}^2 \tag{5}$$

Combining (4) and (5) gives

$$RMS_{ACC} = \sqrt{\frac{N_A^2 \cdot P[s(n)]}{N_A \cdot \sigma_{noise}^2}} = \sqrt{N_A} \cdot RMS_{\text{one waveform}} \tag{6}$$

Q.E.D.

Correspondingly, the SNR after accumulation is thus given by (8), Q.E.D.

$$SNR_{ACC} = 10\log_{10}\left(\frac{N_A^2 \cdot P[s(n)]}{N_A \cdot \sigma_{\text{ACC}}^2}\right) \tag{7}$$

$$= 10\log_{10}\left(N_A \cdot \frac{P[s(n)]}{\sigma_{\text{noise}}^2}\right) = 10\log_{10}(N_A) + SNR_{\text{one waveform}} \tag{8}$$

5.2 The accumulation engine

Each sample in a waveform entering the WFA is represented in 2's complement format with 14 bits (from the 14-bit ADC). The accumulated result has the same length as a single waveform but the size of each data word is 32 bits instead of the original 14. There is therefore 18 bits available headroom for the additions, as illustrated in Fig. 14. Consequently, $2^{18} = 262\,144$ waveforms may be added before there is a risk of overflow.

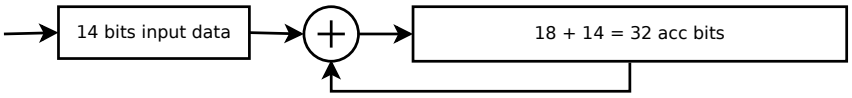


Figure 14: The 14-bit input data is accumulated in a 32-bit word. The accumulation head-room is thereby 18 bits.

The WFA on ADQ-FWATD is *partitioned*, meaning that the work load is split between the digitizer and the host computer (inside the ADQAPI). Fig. 15 presents a block diagram of the ADQ-FWATD structure, starting from the digitizer and ending in user space. The 32-bit accumulator running in the digitizer's FPGA works tandem with a second 32-bit accumulator, implemented in software and running in a separate thread from the user application. This structure offers flexibility and capabilities beyond what could be achieved by using the accumulators separately. The partition algorithm attempts to divide the work load so as to perform as many accumulations as possible in hardware, to reduce the required transfer speed to host.

Note

If possible, with the given parameter set (N_A, R_L, etc.), all accumulations will be performed in the firmware. The data rate over the physical interface will be $1/N_A$ times the input data rate.

If the current parameter set does not allow for all accumulations to be done in firmware, a partitioning will be made between the firmware and the API layer. The partitioning result is available to the user via the ADQAPI command `ATDGetDeviceNofAccumulations`. In this case, the output data rate is *not* $1/N_A$ times the input data rate, but something larger.

Data is delivered to user space through a queue interface where the user is responsible for allocating memory for the accumulated records and keeping the queue of available memory from running out. See also Section 8.4.

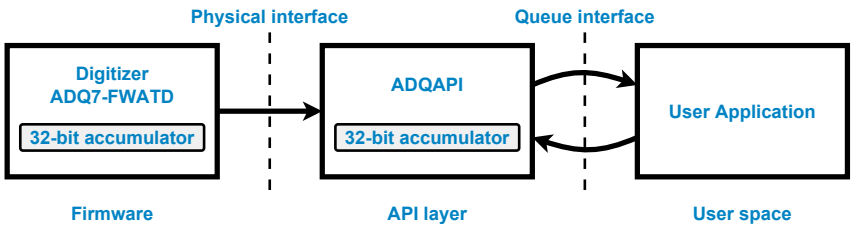


Figure 15: A simple block diagram of the partitioned WFA.

6 Suppression of systematic noise

When accumulating a large number of waveforms, any systematic signal will be favoured and appear more clearly (Section 5). Unfortunately, this is true not only for the wanted signal, but also for correlated noise. Such correlated noise is also called *pattern noise* or *systematic noise*. The major source of pattern noise is the interleaving of the ADC cores (Section 4.3.2) illustrated in Fig. 16. Other sources of pattern noise are clocks, correlated to the trigger frequency. Normally, the ADC's built-in noise suppression can be enough, but because of the WFA, the remaining variations might still be a problem (since they are subject to the same gain as the wanted signal, as illustrated Fig. 17. Pattern noise will thereby reduce the visibility of weak signals so that a signal in the same range as the pattern noise will be hidden also after accumulation.

Important

Random noise will decrease with accumulation but pattern noise is subject to the same gain as the wanted signal.

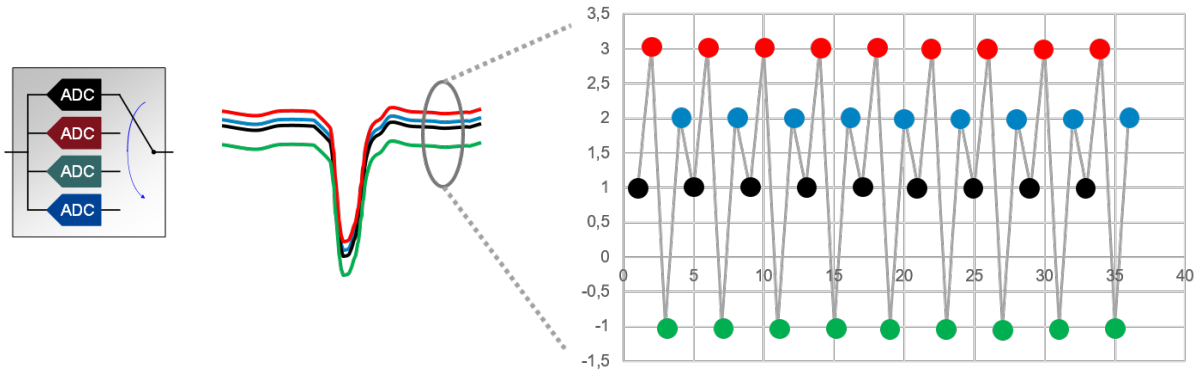


Figure 16: Pattern noise from an interleaved ADC with four cores.

6.1 Three methods

Three methods for dealing with systematic noise are offered. One of them is DBS which is already known from Section 4.3.2. The other two methods use the triggering of the waveform to *randomize the systematic noise*. Although the noise is still systematic, it will differ from waveform to waveform, thus the accumulation of a number of waveforms will randomize and suppress the noise, as illustrated in Fig. 18. Fig. 18 shows the ideal case, where the trigger position is distributed equally over the four ADC cores.

The three methods have different qualities so the method of choice depends on the application at hand. The properties of the methods described below are summarized in Table 2.

DBS The DBS removes the pattern noise by estimating the offset from the data and subtract the per-core estimated offset. Fig. 19 shows background noise from an ADQ7 with and without DBS. This method has high precision and works for low activity systems, that is, systems with a clear baseline. Its limitation is thus that it needs a baseline to be able to estimate the offsets correctly. High activity

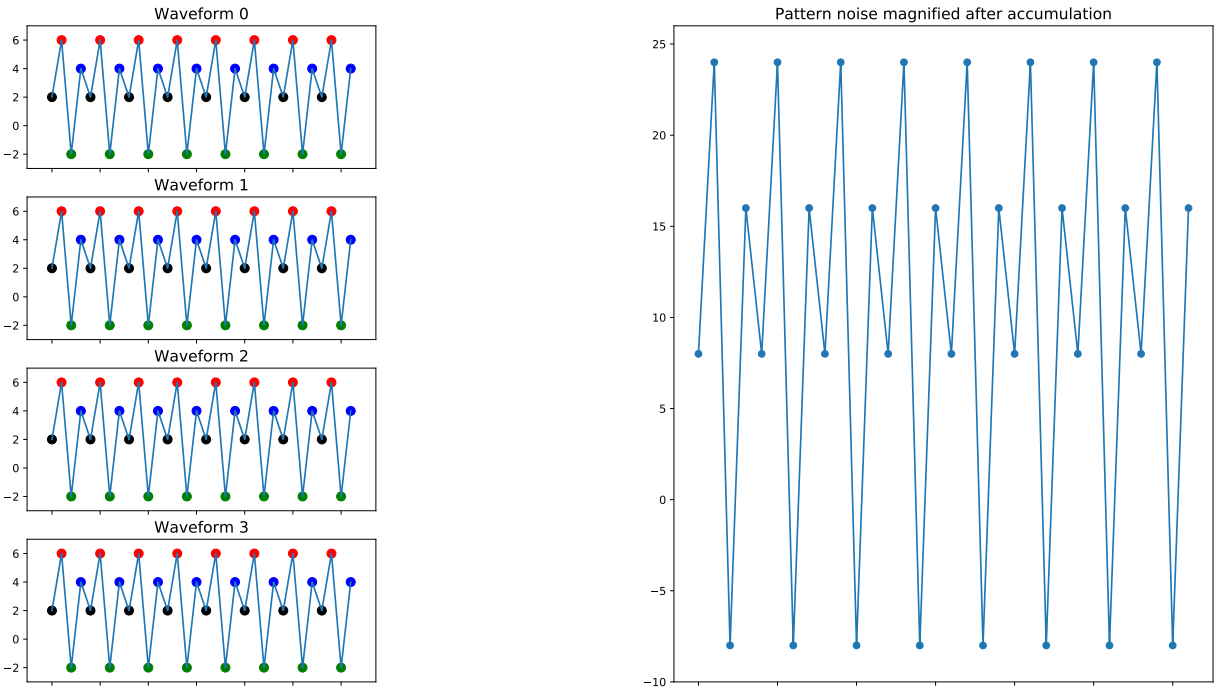


Figure 17: Patter noise is subject to the same gain as the wanted signal. The repetitive patterns is four samples long.

input signals (such as sine waves) yield a poor result. An advantage of DBS is that it works on a single record; large numbers of accumulations are not needed to get a good result, as shown in Fig. 20.

Free running trigger Having a free running trigger means that it is not phase-locked to the sampling clock of the ADQ. The internal trigger source (see Section 3) is for example by default free running. Suppression of pattern noise with a free running trigger is typically good for large numbers of accumulations. Even though the level of suppression is not guaranteed (as the frequency of the trigger is undetermined), the probability of achieving good noise reduction increases with the number of accumulations, N_A. A drawback with a free running trigger is that the unlocked trigger may smear short pulses[7].

Trigger module solution (additional HW option) The trigger module is an external pattern noise suppression trigger source that can be connected to TRIG (physical connector). The trigger frequency is determined to be a number which is not correlated to any sequences or clocks of the ADQ. As a result, the pattern noise is suppressed with accumulation. As opposed to a free running trigger, the trigger frequency is not arbitrary but fixed, to a value that randomizes the pattern noise and thereby suppresses it with accumulation. The drawback with smearing out short pulses is the same as for the free running trigger[8].

[7] That is, the flanks of the accumulated pulses will be smeared due to the variations in the start (and end) of each pulse. Short pulses are more sensitive to such variations.

[8] Except if the external trigger module also triggers the pulse generation.

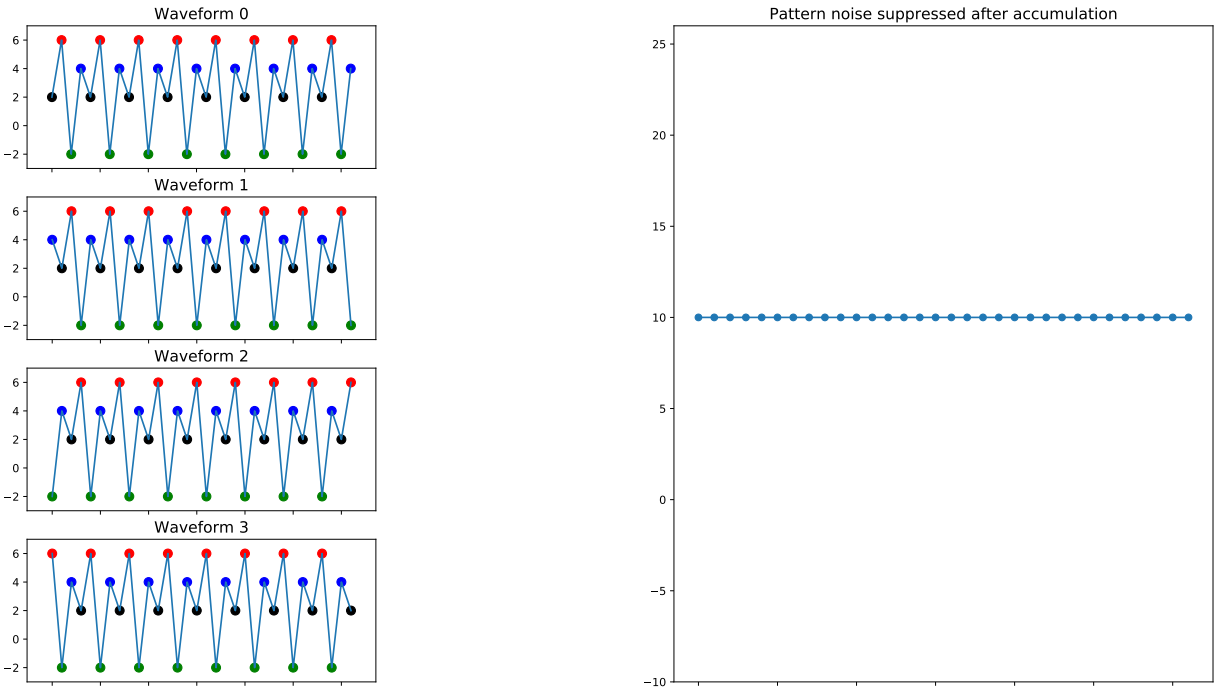


Figure 18: With a free running trigger or the trigger module solution, the start of the systematic noise varies. Here, the ideal case is shown, where the four trigger events are distributed equally over the four ADC cores.

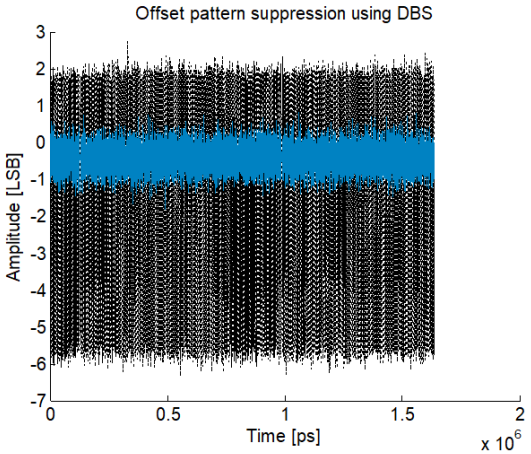


Figure 19: Background noise from ADQ7 with (blue) and without (black) DBS.

Note

If the application at hand offers a baseline, in the meaning that it is a low activity system, DBS can well be combined with one of the two triggering methods.

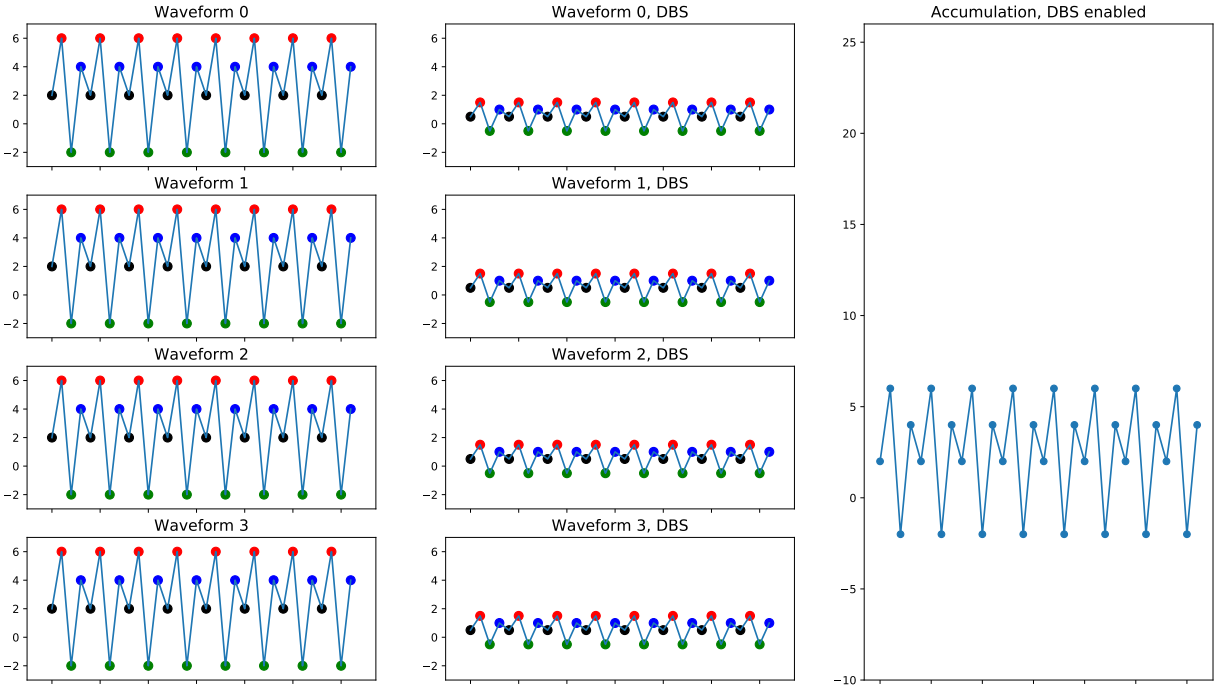


Figure 20: DBS suppresses pattern noise on a single waveform, accumulation is not needed to get a good result.

Table 2: Pattern noise suppression methods work best under different conditions. The method of choice depends on whether the application at hand has high or low activity, and if it includes many or few accumulations. Low activity means here that an isolated baseline dominates the signal constitution over time.

method	low activity	high activity	few acc	many acc
DBS	✓		✓	✓
Trigger module	✓	✓		✓
Free running trigger	✓	✓		✓

7 Finding rare pulses

An ordinary accumulation result in a noise power that grow as N_A at the same time as the signal power grow as N_A^2 (Section 5.1). The result is then an SNR increase of 3 dB per doubled amount of accumulated records. This is however valid only if the signal of interest is present in *every* record. Current chapter treats the case when some of the pulses are so rare that they do not appear in each record.

7.1 Threshold operation

The threshold operation is a non-linear method of suppressing random noise (e.g. thermal). The method is a complement to the succeeding WFA, tailored for rarely occurring weak pulses.

The threshold function in a basic model is one of the following (depending on polarity of the unipolar pulse), Fig. 21:

- For a system with positive pulse, samples below a threshold values is set to the baseline value.
- For a system with negative pulse, samples above a threshold values is set to the baseline value.

The result is that high values are considered wanted signal and kept, and low values are considered noise and are removed. The parameters `threshold` and `baseline` (white boxes in Fig. 21) are set in ADC codes, that is in the range [-32 768,32 767][9]. If DBS (Section 4.3.1) is enabled, `baseline` can preferably be set to the same values as `dc_target` (also in ADC codes).

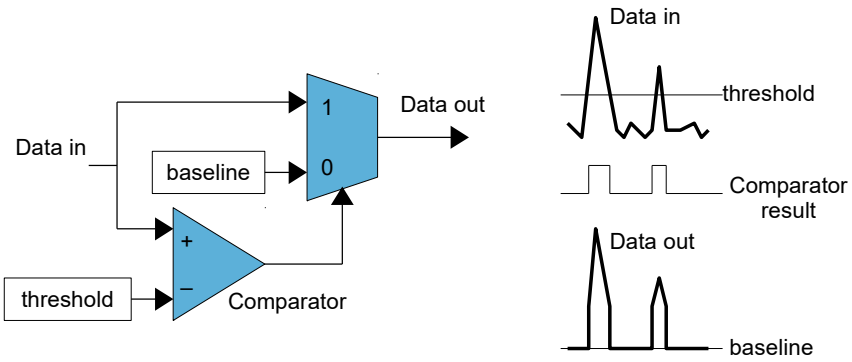


Figure 21: Standard threshold operation on a positive pulse. All samples below threshold is substituted with the baseline value. The white boxes are controlled by the user.

7.2 Advanced threshold operation with filter

An additional available feature is that the user can choose to apply the threshold to a *filtered* version of the signal. A filter is a general form of correlation. If the user has knowledge about the shape of the incoming pulses, that information can be used to favour incoming data of that particular shape. This is done by programing the threshold FIR filter to a similar shape. A block diagram and an example of the function is presented in Fig. 22.

[9] Data in the FPGA is represented with 16 bits, MSB aligned with the 14 bits from the ADC.

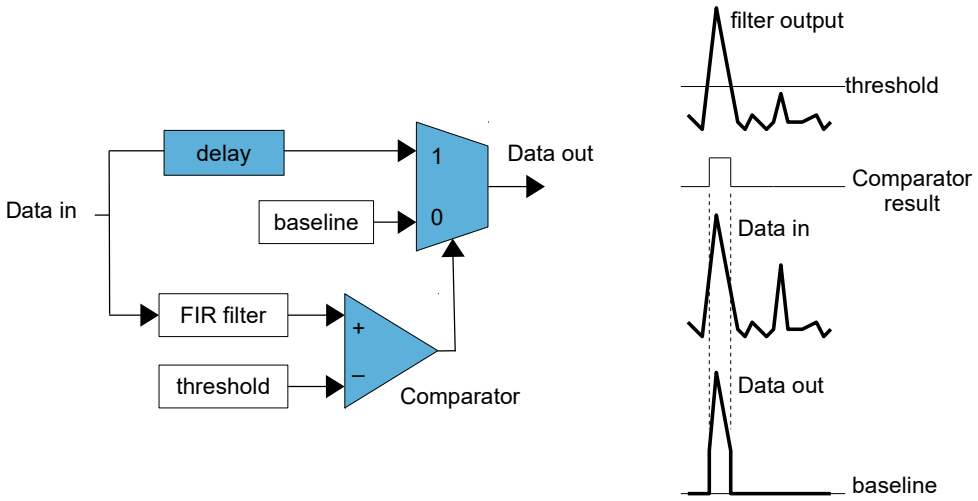


Figure 22: Block diagram (left) and illustration example (right) of the advanced threshold function with filter with the goal to single out weak pulses from the surrounding noise. The FIR filter attenuates the rightmost peak so that it lands under the threshold level. The white boxes are controlled by the user. A delay line (blue box) is added to match the timing of the two branches.

The threshold is applied to the filtered signal to favour pulse-shaped samples before random noise. Samples with pulse information, are saved, and all other samples are set to the baseline value. Note that the threshold is applied to the filtered signal, whereas data from the original non-filtered signal is selected as output, so that the filter does not affect the shape of Data out. To match the timing of the two branches, a delay line is added in the upper branch. Filter coefficients should be input as 16-bit two's complements words with 14 of them being fractional.

Important

Data is not changed by the threshold filter. The filter only affects the comparator result.

Example

The digital value 1_{10} represented as a threshold filter coefficient with 14 fractional bits is $0100\ 0000\ 0000\ 0000_2$ and $-1_{10} = 1100\ 0000\ 0000\ 0000_2$.

An example on advanced threshold with filter is given in Fig. 23. Without filter, a threshold can not distinguish the wanted signal from the noise. However, applying a filter that favours wider pulses, it is possible to find a threshold level that can detect the signal. The filter coefficients used are plotted in Fig. 24.

Note

The threshold filter is not the same filter as the bandwith-limiting filter in Section 4.4, although they have the same structure and length.

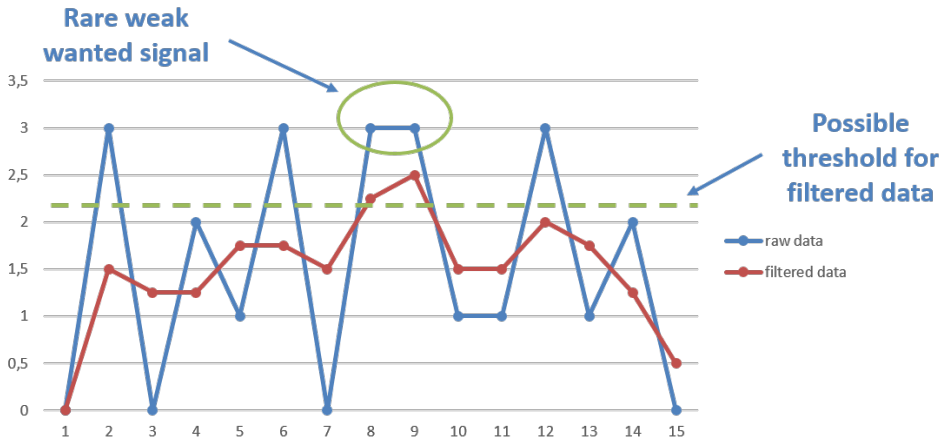


Figure 23: Applying a threshold on the raw data cannot distinguish the signal from the noise, since they have the same magnitude. However, taking into account that the signal width is two samples, and applying a filter with length > 1, the filter output will have a clear peak when the signal comes. The threshold can be lowered to e.g. 2.2 so that the wanted signal is distinguished from the noise and can be detected. Note that the threshold value must be convert to ADC codes before it is applied.

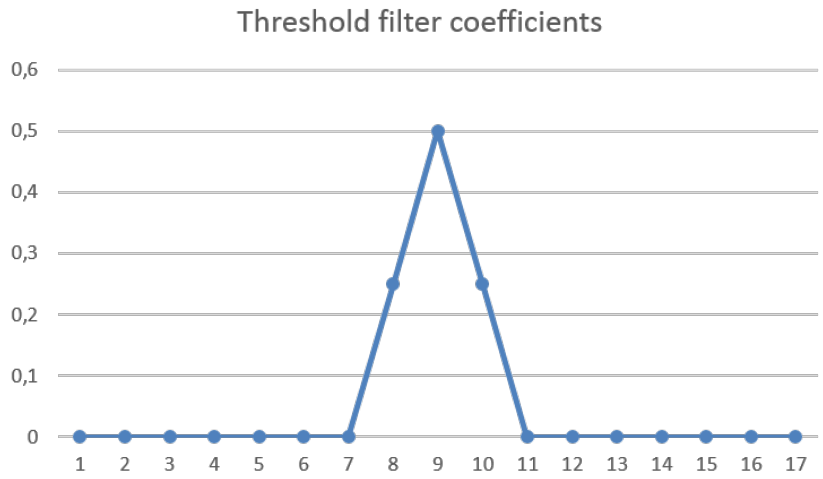
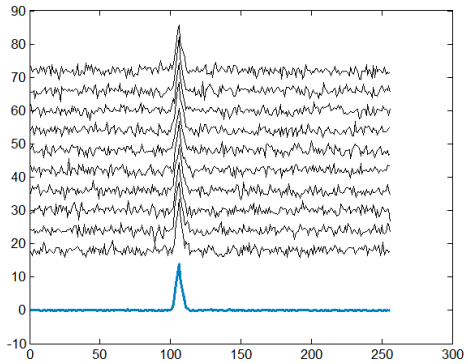


Figure 24: Impulse response in decimal representation of the filter used in the example on advanced filtering in Fig. 23. To be convert from decimal to binary representation, $2 + 14$ bits.

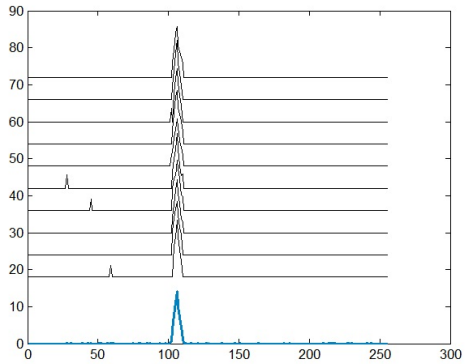
7.3 Threshold example

Fig. 25a illustrates a set of records that are accumulated $N_A = 10$ times, and the SNR is improved with $10 \log_{10} N_A$ according to the theory in Section 5. In Fig. 25b, a threshold function is added to further improve performance. There is a clear effect on each individual record where noise has been suppressed by the threshold operation. But the result *after accumulation* is not significantly improved, compared to Fig. 25a without applying a threshold.

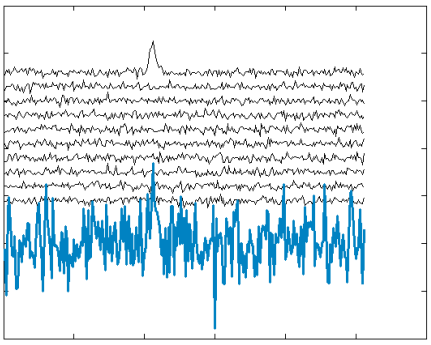
Fig. 25c illustrate an example with a rarely occurring pulse; the pulse is present in only *one* of the ten records. Their accumulated result is also shown (blue). The SNR is not improved by the accumulation, but instead the single pulse is lost in the noise floor. Here, the essence of the threshold operations is illustrated. In Fig. 25d, a threshold function is applied on the records in Fig. 25c and the effect on the accumulated record is clear.



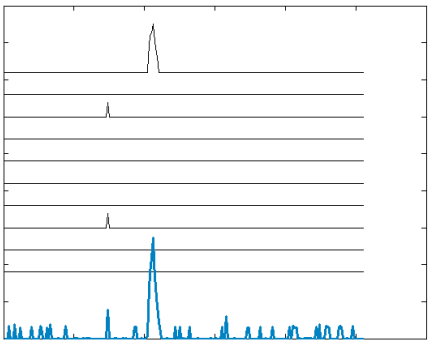
(a) ADQ-FWATD no threshold operation.



(b) ADQ-FWATD using threshold operation. Performance same as in (a).



(c) One pulse no threshold operation. Peak is lost in noise after accumulation.



(d) One pulse with threshold operation. Peak is clear also after accumulation.

Figure 25: Illustration of how the threshold operation is applied prior to accumulation in order to achieve significantly increased performance for the case where pulses are so rare that they do not occur in all records.

8 Data read-out to user application – ADQ7 only

Important

Section 8 is ADQ7-FWATD specific. For information on data read-out on ADQ14-FWATD, please refer to the user guide [6]. Section 8.3 is an exception; it is valid also for ADQ14.

A strength of ADQ7-FWATD is that it provides the user not only with (accumulated) waveforms but also with metadata of each user space[10] record. To this end, metadata and a pointer to the waveform data is organized in an ATD data struct, according to Fig. 26. Each field is described in Section 8.1. Sections 8.2 and 8.3 explain how the streaming is made seamless and dead-time free. Finally in this chapter (Section 8.4) the important subject of overflow is treated. Since all parts of the system is not under the complete control of the ADQ, it is important that if an overflow occurs, the ADQ behaves in a deterministic and well-defined manner.

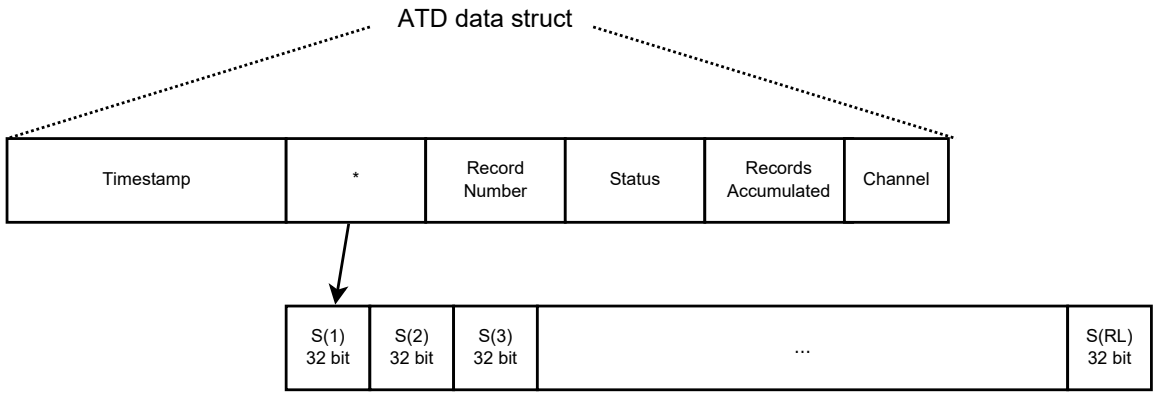


Figure 26: The user space record is presented to the user in the form of an ATD data struct. Both metadata and a pointer to the accumulated waveform data is included. The pointer ($*$) points to a waveform buffer with R_L samples (total size $R_L \times 32$ bits).

8.1 The ATD data struct

The ATD data struct is used to represent a record in user space and is used by all FWATD-specific functions associated with data handling. The struct is an abstraction of a record. It does not only include a pointer to the accumulated waveform data but also metadata with fields described below. For more information on the overflow behavior of FWATD, see Section 8.4.3.

Timestamp The timestamp of the first waveform included in an accumulation.

RecordNumber A chronological counter, telling the user how many accumulation records that has been started so far, each with N_A waveforms. If a number is missing, that record has been discarded due to overflow.

[10] The user space is illustrated in Fig. 15.

RecordsAccumulated Number of accumulations in the user space record. Should be N_A, but if something went wrong (overflow),
RecordsAccumulated < N_A.

Status Four flags with information on the current overall health of the WFA:

Bit 3 Arithmetic overflow. Samples have been saturated.

Bit 2 Data is lost *within* the record (RecordsAccumulated $< N_A$)

Bit 1 The Write FIFO (Fig. 28) was empty which caused a stall of the device-to-host interface.

Bit 0 At least one *whole* preceding record has been discarded by the robustness mechanism (Section 8.4). RecordNumber shows the discrepancy between the number of started accumulations and the actual number received by the user application.

Channel The originating channel.

Example

Examples illustrating how overflow information is communicated to the end user via metadata in the data struct is given in Tables 3 and 4. $N_A = 200$, and the number of times to repeat the accumulation process $N_R = 4$. The case with data loss within a user space record, as well as the case of loss of an entire record is included.

Table 3: Example of overflow where data is lost within a user space record where $N_A = 200$ and $N_R = 4$. Status flag Bit 2 is set and RecordsAccumulated $< N_A$ in the second repetition.

RecordNumber	Status	RecordsAccumulated
1	0000	200
2	0100	193
3	0000	200
4	0000	200

8.2 Seamless streaming with queue interface

To enable seamless streaming, a queue interface is used to transfer data to the user application. The overall structure was introduced earlier in Fig. 15, and here in Fig. 27 and 28 some more details are given. The user is responsible for keeping the queue (the Write FIFO) filled with addresses to vacant data struct buffers.

The flow of the seamless streaming is explained below.

1. The user allocates the needed amount[11] of Waveform memory, where each buffer is $R_L \times 32$ bits.

[11] The needed amount of buffers is highly application dependent.

Table 4: Example of overflow when an entire user space record is missing where $N_A = 200$ and $N_R = 4$. Status flag Bit 0 is set (last row), indicating that at least one record is lost. User space record with RecordNumber = 3 is missing.

RecordNumber	Status	RecordsAccumulated
1	0000	200
2	0000	200
4	0001	200

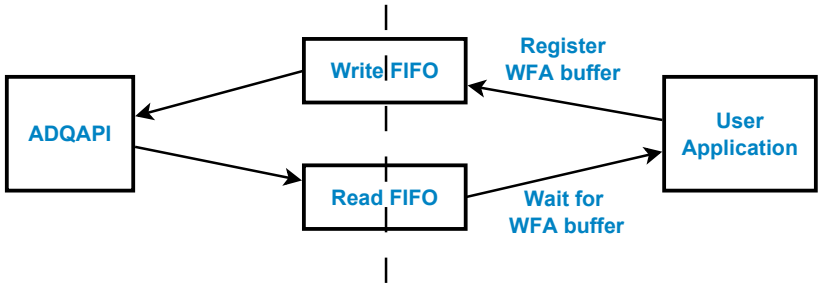


Figure 27: The user provides the API with addresses to available data struct buffers (Write FIFO). As soon as the API has finished an accumulation and the result is available on a valid address, (from the Write FIFO) that address will be written to the Read FIFO. From that point, data at that address is the user's responsibility.

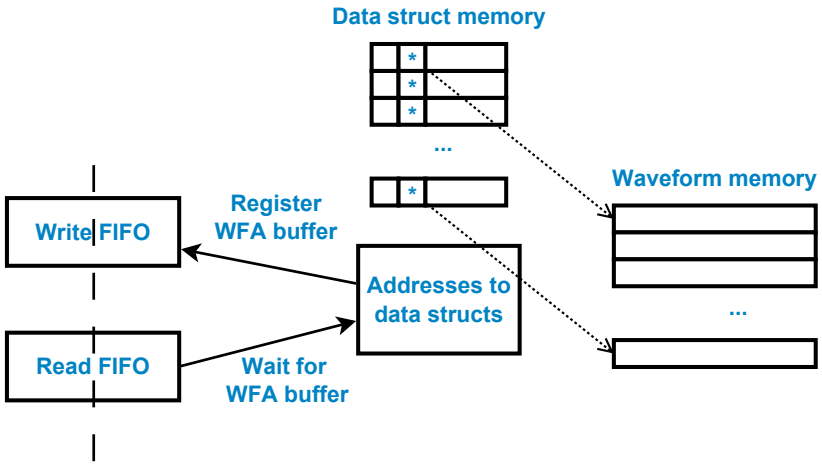


Figure 28: At startup, the user allocates space for Waveform memory and Data struct memory with static sizes. The API will write to these locations, starting with the first address it gets from the Write FIFO. The address goes to a buffer in the Data struct memory. That in turn will point to the corresponding Waveform memory buffer.

2. The user allocates Data struct memory, where each buffer is 32 Bytes [7].
3. The user fills the Write FIFO with the addresses to the ATD data struct buffers.
4. At WFA engine startup, the first entered address will be used by the API to store the accumulated result (in the Waveform memory) and the corresponding metadata (in the Data struct memory).

5. When one accumulation his completed, the API will return the address of that data struct buffer in the Read FIFO.
6. The user waits for a data struct address to show up in the Read FIFO and once it shows up, the data is the user's responsibility.
7. After the data is handled by the user application (e.g. storing or further processing) the address is once again made available to the API by returning it the Write FIFO.

Details on the data collection can be found in [7].

Important

The user is responsible for feeding the queue (Write FIFO) with references to available data struct buffers so that it never drains.

The reuse of memory locations enables transferring an unlimited amount of data using a limited amount of memory. Additionally, due to the continuous nature of the streaming mode, it is well suited for applications which require a feedback mechanism to control devices placed earlier in the signal chain.

8.3 Dead-time free acquisition and duty cycle trade-off

A new accumulation may start only 32 ns after the previous one has finished[12]. This is the time needed for the ADQ to rearm (Fig. 3). Aside from the 320 samples (assuming 10 GHz) that will be missed between two waveforms, the acquisition is dead-time free. Further, the duty cycle is defined as the ratio between the trigger period and the waveform length (1).

Example

The trigger rate is 10 kHz which gives a trigger period of $1/(10\ kHz) = 100\ \mu s$. With a waveform length of 50 μs, the duty cycle is 50 $\mu s/100\ \mu s = 50\%$.

There is a trade-off between the readout speed, duty cycle and N_A. The data rate in to, for example, ADQ7-FWATD is 10 GSPS[13] of data from the ADC. The data rate out from FWATD is set by the link to the PC. For ADQ7-FWATD with PCIe, it is limited to 5 GBytes/s and with the USB3.0 it is about 200 MBytes/s [4]. For corresponding ADQ14 figures refer to [2]. The data rate is calculated from

$$\text{Data-rate\_out} = \frac{\text{Data-rate\_in} * 4 Bytes/Sample * \text{duty-cycle}}{N_A} \tag{9}$$

Given that each samples is represented with 32 bits = 4 Bytes and a duty cycle of almost 100%, the minimum number of waveforms per accumulation for a USB3.0 interface can be calculated from (9) as

$$N_A = \frac{10\ GSPS \times 1\ channels * 4\ Bytes/Sample}{200\ MByte/s} = 200\ waveforms. \tag{10}$$

For a PCIe Gen 3x8 lanes link, the same calculation gives

$$N_A = \frac{10\ GSPS \times 1\ channels * 4\ Bytes/Sample}{5000\ MByte/s} = 8\ waveforms. \tag{11}$$

[12] Given that the host can handle the data rate. In practice there is a lower limit on the number of accumulations, as will be shown in (10) and (11).

[13] Either 1 × 10 or 2 × 5 GSPS.

Below these limits, the duty cycle has to be decreased to avoid overflow (Section 8.4). Note that these numbers are independent of the length of the waveform.

8.4 System robustness and overflow

An important aspect of FWATD is its system robustness with a deterministic behavior at overflow (loss of data). The core concept is to discard data in a well-defined manner such that the *accumulation grid* is preserved. The accumulation grid is equivalent to the trigger grid down-sampled with a factor equal to the number of accumulations, N_A. The accumulation grid period is

$$T_{\text{grid}} = \frac{N_A}{f_{\text{trig}}} \tag{12}$$

in a system using a constant trigger frequency f_{trig} and N_A number of accumulations.

An overflow is caused by a stall of the data transfer for an extended period of time. The firmware is designed such that overflows will not occur due to internal events. However, there are points in the system where overflow can occur and these are beyond the complete control of the digitizer. The first point is the physical device-to-host interface (Section 8.4.1) and the second is the queue interface (Section 8.4.2). These are the two interfaces in Fig. 15.

8.4.1 Overflow in the physical interface

The physical device-to-host interface (Fig. 29) is the most critical point of the system, regarding overflow sensitivity. This is where the digitizer is connected to the host computer. The effective data transfer rate directly translates to boundaries for the digitizer settings. For example, trade-off between duty cycle and minimum number of accumulations for a given data transfer rate were given in (10) and (11).

Note

The most sensitive point for overflow is the physical device-to-host interface. The data transfer rate here sets the boundaries of the digitizer settings.

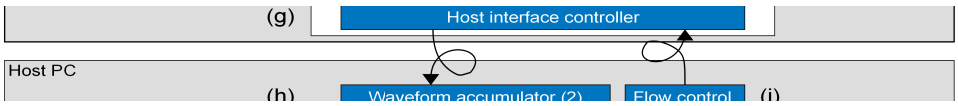


Figure 29: Zoom in Fig. 1 on the physical interface, the weakest link in the chain regarding overflow.

8.4.2 Overflow in the queue interface

The second critical point in the system is the queue interface in the PC, where the responsibility of ATD data structs and accumulated waveforms is shifted from the API to the user application. If there is no free memory to place the new data struct and waveform (empty Write FIFO in Figs. 27 and 28), the data flow will stall and eventually cause incoming data to be discarded.

The number of (data struct) buffers needed to ensure stable operation is dependent on how fast they can circulated from the Read FIFO to the Write FIFO. Thus, the complexity of data processing in the user application.

Note

The number of queued buffers needed to ensure stable operation of the system is highly dependent on buffer handling in the user application.

8.4.3 Overflow behavior

During an overflow, data collected up until that point (waiting to be transferred) remains intact and incoming data is discarded in a well-defined manner, maintaining the accumulation grid defined in (12). The user is informed of the overflow via the Status field in the ATD data struct. For a description of the Status field, see Section 8.1 and Fig. 26). Information on where the overflow ocurred[14] is given in Bit 1. There are two kinds of overflow which are described below. See also the example in Section 8.1.

1. A user space record will contain fewer number of accumulated waveforms than the defined number of accumulations (RecordsAccumulated $< N_A$). Can only occur if

$$\texttt{ATDGetDeviceNofAccumulations} < N_A.$$

   Bit 2 in the Status field in the ATD data struct (Fig. 26) is set.

2. Entire user space records are missing (RecordNumber is incremented with more than one since last user space record). Bit 0 in the Status field (Fig. 26) is set.

[14] Physical interface or queue interface.

9 Application examples

The final chapter in this application note is a brief on how ADQ-FWATD can be used in different applications.

9.1 Measuring a changing process

A seamless accumulation (Section 8.2) is a key feature when measuring a changing process. Seamless accumulation means that practically no data is lost when a new accumulation is started, and thereby no important information will be lost either.

The record length, R_L is the duration of a measurement (e.g. in time-of-flight), and the number of records in the accumulator, N_A affects the accuracy. N_A also sets the tracking speed of the changes. A high number of accumulation is possible for slowly changing processes, whereas a fast changing process allows for only a few accumulations.

The acquisition with accumulation is a way of sampling the changing process with process sampling frequency

$$f_{s,process} = \frac{f_s}{R_L \cdot N_A} \tag{13}$$

where f_s is the sampling frequency of the digitizer, R_L the record length, and N_A the number of accumulations.

Example

With

- $f_s = 10$ GSPS
- $R_L = 1$ MSamples (100 μs)
- $N_A = 100$ accumulated waveforms

a noise suppression of $10 \log_{10}(100) = 20$ dB is achieved and the update rate will be $\frac{10 \times 10^9}{10^6 \cdot 100} = 100$ Hz. Fluctuations with frequencies up to $f_{s,process}/2 = 50$ Hz can be followed.

As regards the choice of method for pattern noise reduction (Section 4.3.2), DBS is recommended for such a low activity system. See Table 2.

9.2 Live updates of subtotals

For user cases with a large N_A, not only the final result, but also subtotals can be of interest. Current section gives an example on how to make subtotals available for display, so that the progress of the noise suppression can be followed on a screen.

The application is built with a live update of the time spectrum by partitioning the accumulations so that some of them are done by the user application (Fig. 30). A block diagram is given in Fig. 31.

Note

If all accumulatiosn are done by FWATD, partial results are not available to the user.

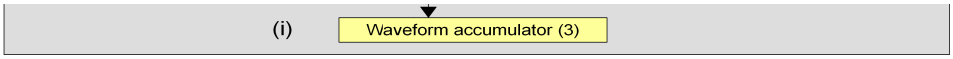


Figure 30: Zoom in Fig. 1 on Waveform accumulator 3. By doing some of the accumulations here, subtotals are available to give the user live updates of the progress.

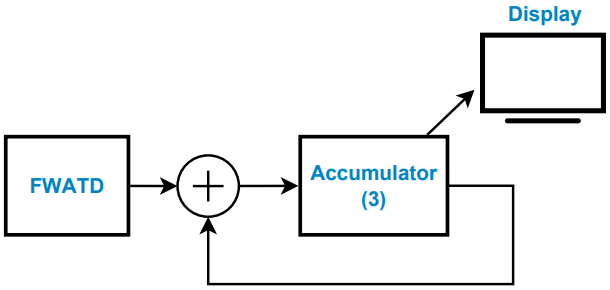


Figure 31: Subtotals are available for display by moving parts of the accumulations to the user application.

Table 5: Specification of live update example.

Parameter	Name	Value
Trigger rate	$f_{trigger}$	10 kHz
Sampling rate	f_s	10 GSPS
Display update rate	$f_{display}$	10 Hz
Waveform length	t	200 μs
Number of accumulations	N_A	800 000

The specification is given in Table 5. The display update rate $f_{display}$ is a key parameter when partitioning the system since it sets the requirements on the host PC link, so first a sanity check of the required data transfer rate is needed. Firstly, R_L is calculated as

$$R_L = t \cdot f_s = 200 \times 10^{-6} \cdot 10 \times 10^9 = 2\ MSample \tag{14}$$

Secondly, since each accumulated sample is represented by 32 $bits = 4\ Bytes$, the size of one waveform will be

$$R_L \cdot 4\ Bytes = 8\ MBytes. \tag{15}$$

With an update rate of $f_{display} = 10$ Hz, ten such records must be sent over the physical interface each second, giving a required data transfer rate of $8 \cdot 10 = 80\ Mbytes/s$. Data rates in this order of magnitude is easily reached with both USB3.0 and PCIe (see also Section 8.3).

After verifying that the physical interface can handle the required data transfer rate, the partitioning of accumulations between FWATD and the user application (waveform accumulator (3) in Fig. 1) is calculated.

To get from a trigger rate of $f_{trigger}$ down to the display rate $f_{display}$,

$$\frac{f_{trigger}}{f_{display}} = \frac{10k}{10} = 1000$$

accumulations must be done by FWATD. The rest of the accumulations (800 000/1 000 = 800) must thus be done in the user application, while plotting the partial results.

9.3 Semiconductor Automated Test Equipment (ATE)

Semiconductor ATE consists of various instruments or cards used for testing memory, digital, mixed-signal, and system-on-a-chip (SoC) components, both at the wafer and packaged stages. Driven by the demand in the consumer, computing, and communication markets, these test systems continue to evolve. To keep pace with innovation in the semiconductor industry, today's ATE products must provide more functionality at higher speeds than ever before.

Using the WFA and accumulating repeated measurements makes it possible to see component characteristics below noise level. These kind of measurements have relatively high signal activity, and therefore the Trigger box is recommended before DBS for pattern noise reduction.

References

[1] Teledyne Signal Processing Devices Sweden AB, *16-1794 ADQ7-FWATD Datasheet*. Technical Specification.

[2] Teledyne Signal Processing Devices Sweden AB, *14-1397 ADQ14-FWATD datasheet*. Technical Specification.

[3] Teledyne Signal Processing Devices Sweden AB, *14-1351 ADQAPI Reference Guide*. Technical Manual.

[4] Teledyne Signal Processing Devices Sweden AB, *16-1692 ADQ7 datasheet*. Technical Specification.

[5] Teledyne Signal Processing Devices Sweden AB, *14-1290 ADQ14 datasheet*. Technical Specification.

[6] Teledyne Signal Processing Devices Sweden AB, *16-1849 ADQ14-FWATD User Guide*. Technical Manual.

[7] Teledyne Signal Processing Devices Sweden AB, *17-1957 ADQ7-FWATD User Guide*. Technical Manual.

TELEDYNE SP DEVICES
Everywhereyoulook™

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: spd_info@teledyne.com