Manual ADQ7WB

This manual describes how to get the full potential out of Teledyne SP Devices' digitizer ADQ7WB. The manual includes these steps:

- *Set up the analog front-end*
- *Master the triggers*
- *Control the acquisition*
- *Manage the sampling clock*
- *Understanding data transfer to host PC*
- *Using GPIO*

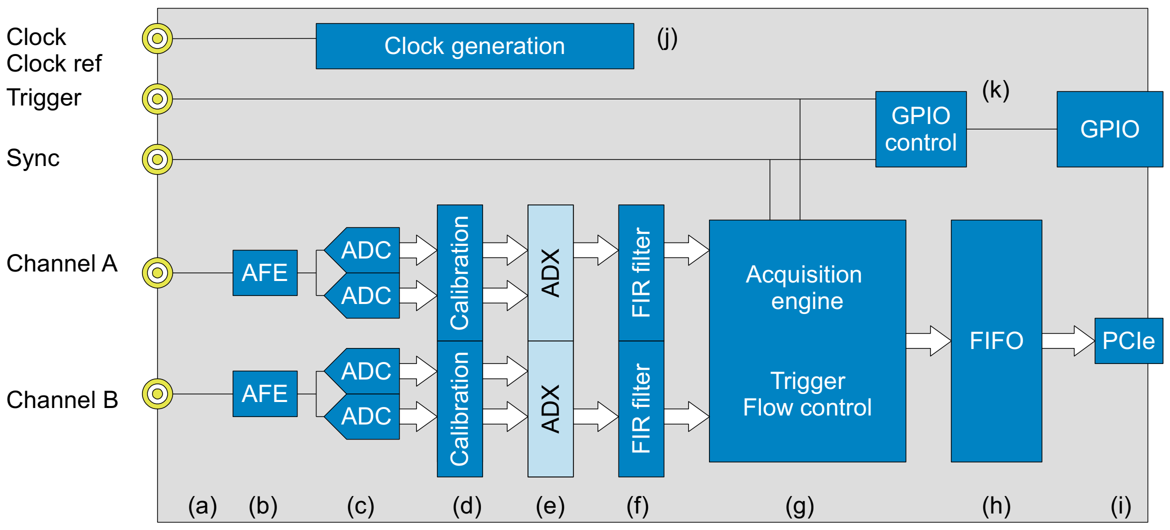
Table of content

1 INTRODUCTION

The purpose of this manual is to explain how the digitizer is operated. The datasheet [1] contain parameters for the specific versions of digitizer. References to software commands are made. In some places, pseudo code is used for description. See [2] for details on how to use the software commands and see [3] for general guidelines on programming the digitizer.

1.1 ADQ7WB Architecture

The ADQ7WB architecture is shown in Figure 1. References to the corresponding sections with further information are also included.



#	DESCRIPTION	REFERENCE
a	AC-coupled analog inputs	2
b	Signal conditioning analog front-end	2
c	High speed and high resolution A/D converter. The A/D converter operate interleaved at 5 GSPS per channel.	3
d	Digital calibration of gain and offset.	2
e	Teledyne SP Devices' proprietary technologies for signal quality enhancement of interleaved ADCs.	3
f	User-programmable FIR filter. Symmetrical 17 taps.	3
g	Acquisition engine that handles triggers and controls the data flow	4, 7
h	Data FIFO to buffer data before transmission to the host PC.	7
i	The data transfer to the host PC is through a PCIe link.	7
j	Flexible clock generator	5
k	General Purpose digital Input and Output control.	6

Figure 1: ADQ7WB architecture.

1.2 Fundamental design properties

There are some fundamental design properties that are necessary to understand before continuing.

1.2.1 Data format

The ADC components of ADQ7WB has 12 bits resolution, while the data format inside the ADQ7WB and out to the host PC is 16 bits. The 12 bits from the ADCs are MSB aligned in this 16 bit data word. Thus initially the 4 LSBs are zero.

The number representation is 2's complement. The full scale maximum code is then 32 767 and the full scale minimum code is –32 768. Overflow or underflow at any position in the signal path will saturate the data and turn on an overflow flag. See **Section 7.7** for more information on over- and under-flow.

The 4 LSBs may not be zero in the data output from the ADQ7WB. Calibration and other computations in the FPGA may result in fractional result. This is not rounded to 12 bits in order to avoid adding computational noise.

Example 1: A 12 bits sequence of data is subject to a gain calibration parameter of 1063. This means that the digital word is corrected by 1063 / 1024, **Section 2.2**. **Table 1** *illustrate how the lowest bits contain computation results. The analog signal level is calculated from* **Section 1.2.6**.

Table 1: Example of how computation results sets the lowest two LSBs.

ADC RAW CODES[1]	GAIN CORRECTION	DIGITAL CODE LEVEL[2]	ACTUAL ANALOG RANGE	ANALOG LEVEL[3]
0x000*0*	1063 / 1024	0x000*0*	0.9 Vpp	0 mV
0x001*0*	1063 / 1024	0x001*1*	0.9 Vpp	0.,23 mV
0x002*0*	1063 / 1024	0x002*1*	0.9 Vpp	0.45 mV
0x003*0*	1063 / 1024	0x003*2*	0.9 Vpp	0.69 mV
0x004*0*	1063 / 1024	0x004*2*	0.9 Vpp	0.91 mV
0x005*0*	1063 / 1024	0x005*3*	0.9 Vpp	1.1 mV

1. This is the raw codes from the ADC. It is 12 bits MSB aligned in 16 bit word. The 4 LSBs are thus 0.
2. This is the result from the gain compensation. The 4 LSBs now contain a fractional result from the computation.
3. This is the corresponding analog signal that was present at the input at the time of measurement. See **Section 1.2.6** for details on how this is calculated.

1.2.2 Calibration

During the factory calibration procedure the analog properties are measured and parameters for a digital compensation are computed. An analog deviation in the front-end is thus compensated for by the inverse function in the digital signal processing part.

Example 2: The full scale signal range of the ADQ is measured in production and the **SetGainAndOffset** *function is used for adjusting to the correct signal range.*

1.2.3 Data acquisition nomenclature

Table 2 defines some key data acquisition terms.

Table 2: Data recording nomenclature.

PARAMETER	DESCRIPTION	REF
ADQ	Collective name for digitizers from Teledyne SP Devices.	
Analog	Analog signal is the input to the digitizer. This is the signal to be digitized.	
Waveform	Analog signal with a distribution in time. This is digitized into a record.	
Sample	An analog signal level is digitized into a sample, that is a numerical value.	
SYNC	Physical connector on the front panel.	4.7, 6.1
Record	A set of consecutive samples is called a record. An analog waveform is digitized into a record of samples.	4
TRIG	Physical SMA connector on the front panel.	4.7, 6.1
Trigger	Trigger is a real-time event that starts acquisition of a record.	4.2
Timestamp	Timestamp is a real-time value that identifies when a trigger happened. The timestamp gives timing information for each sample.	4.3
GSPS	Giga-samples per second (10^9). Clock frequency [Hz] and sample rate [SPS] are both used to denote speed.	
MSPS	Mega-sample per second (10^6).	

1.2.4 ADQ7WB sampling clock frequency

ADQ7WB is designed for 5 GHz sampling frequency only. A different sampling rate can be achieved by using the sample skip function, **Section 5.9**.

1.2.5 System clocks

The different parts of the digitizer operate on different clock rates

The sampling of the analog signal is done on the sampling clock of the ADC (see **Section 1.2.4**).

The external trigger input has a trigger clock which is higher than the sample clock for high trigger time precision (20 GHz).

The PCIe host PC connections has its own clock system.

All other interfaces operate on the data processing clock of the FPGA at 312.5 MHz. This clock is referred to as the Data Clock.

See **Section 5.1** for more details on the clock system.

1.2.6 Analog signal range

The analog signal range (**ACTUAL_ANALOG_RANGE**) is 900 mV_{pp} and is symmetrical around zero.

The maximum digital code 2^15 represents an analog signal with a level **ACTUAL_ANALOG_RANGE / 2** at the input. A specific analog signal **ANALOG_LEVEL** will then be represented by the following digital code:

DIGITAL_CODE_LEVEL = ANALOG_LEVEL / (ACTUAL_ANALOG_RANGE / 2) * 2^15 **(1)**

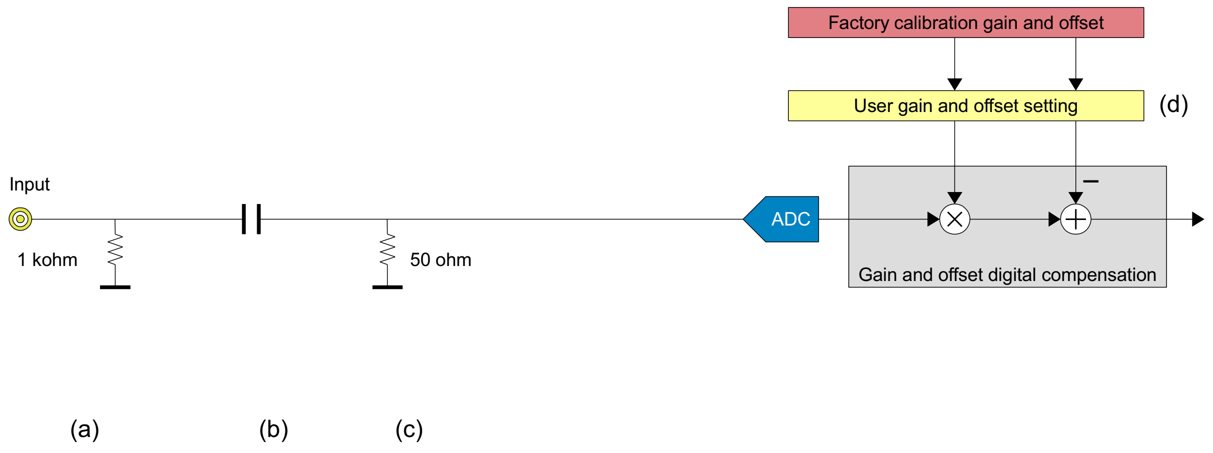
A specific code **DIGITAL_CODE_LEVEL** then represent the analog level as:

ANALOG_LEVEL = (DIGITAL_CODE_LEVEL / 2^15) * (ACTUAL_ANALOG_RANGE / 2) **(2)**

2 SETTING UP THE ANALOG FRONT-END

2.1 AFE block diagram

The analog front-end setup for ADQ7WB is found in **Figure 2**.


Factory calibration gain and offset
User gain and offset setting
(d)
Input
1 kohm
50 ohm
ADC
–
Gain and offset digital compensation
(a)
(b)
(c)

#	DESCRIPTION	USER COMMAND	REF
a	The analog input is DC terminated 10 kΩ to GND to keep control of the DC voltage at the input.		
b	The input is AC-coupled		
c	There is a matched 50 Ω termination.		
d	The gain and offset is calibrated with a sine wave at –1 dB full scale. The digital compensation corrects the offset and the gain at this condition. The user can access this function to set a different gain and offset.	***SetGainAndOffset***	**2.2**

Figure 2: ADQ7 AFE control.

The AC-coupled analog inputs are designed for high bandwidth.

2.2 Adjusting the digital gain and offset

The digital gain and offset block is primarily intended for factory calibration but it may also be accessed by the user, and offers an efficient way of scaling the signal to suit processing in the PC.

The default setting is the calibration parameters ***CAL_GAIN*** and ***CAL_OFFSET***. The raw data from the A/D converter, ***ADC_RAW_CODE***, is corrected with the calibrated values according to:

DIGITAL_OUTPUT_CODE = ADC_RAW_CODE * CAL_GAIN – CAL_OFFSET **(3)**

The user can override these settings by using the software command ***SetGainAndOffset***. The parameter ***USER_GAIN*** and ***USER_OFFSET*** can be applied in two ways; relative to the calibrated value or relative to the raw code.

The normal mode of operation is to apply the gain and offset settings relative to the calibrated data as

DIGITAL_OUTPUT_CODE =
ADC_RAW_CODE * CAL_GAIN * USER_GAIN – CAL_OFFSET – USER_OFFSET. **(4)**

By setting bit 7 in the channels parameter, the calibration data is overridden as:

DIGITAL_OUTPUT_CODE = ADC_RAW_CODE * USER_GAIN – USER_OFFSET **(5)**

To get the raw code, ***ADC_RAW_CODE***, use ***SetGainAndOffset(128+CHANNEL,1024,0)***.

3 SIGNAL QUALITY ENHANCEMENT

3.1 Interleaving correction ADX

The Interleaving correction ADX is available on the 5 GSPS, which is internally interleaved.

The ADX automatically corrects for interleaving mismatch in gain, offset, and timing in the ADC cores. The ADX also compensates for variation over frequency.

At start-up, ADX is loaded with factory calibrated settings but the correction is by-passed. Control ADX by the commands ***SetInterleavingIPEstimationMode*** and ***SetInterleavingIPBypassMode***.

Note that ADX is intended for systems with high energy in concentrated frequency bands, like radio channels.

3.2 User-programmable FIR-filter

3.2.1 Use case

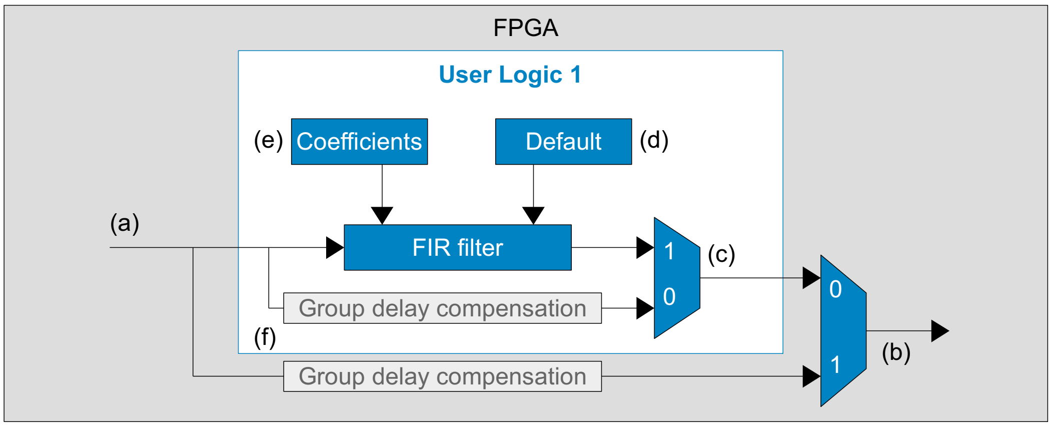
The user-programmable FIR filter is a general purpose symmetrical filter of length 17. This can be used for reducing the noise in unwanted signal bands. Assuming that the wanted signal has a limited band-width and is present in only a part of the digitized frequency band. Then this filter can be used for removing noise in the unwanted signal band and thus increase system performance.

3.2.2 Activating the filter

The block diagram of the filter is in **Figure 3**. The filter is placed in the block “User Logic 1”, see **Section 3.2.4** and **[6]**. This means that the ***BypassUserLogic*** switch has to be set to 0. The filter can then be enabled or disabled by the command ***EnableUserLogicFilter***.

When the filter is bypassed, delay compensation blocks are activated to preserve the timing of the system.

.



#	DESCRIPTION	USER COMMAND	REF
a	This is the input signal to the filter.		**[6] 3.2.4**
b	Bypass “User Logic 1” is default. Set the switch in the mode for not bypassing “User Logic 1” to make it possible to use the filter.	***BypassUserLogic***	**[6] 3.2.4**
c	Enable the filter	***EnableUserLogicFilter***	**3.2.2**
d	Set default parameters	***SetUserLogicFilter***	**3.2.3**
e	Set the FIR filter coefficients	***ResetUserLogicFilter***	**3.2.3**
f	Group delay compenasation to maintain the same timing for all use cases.		

Figure 3: Filter block diagram

3.2.3 Setting filter coefficients

The filter coefficients has to be set by the command ***SetUserLogicFilter***, There is a default set of coefficients with only a one on the center tap which is activated by the ***ResetUserLogicFilter***.

The order of the filter is 16 and the length is 17. It is a symmetric filter which means that there are 9 unique coefficients. These are given as the vector to the ***SetUserLogicFilter*** and interpreted as h in **Figure 4**.

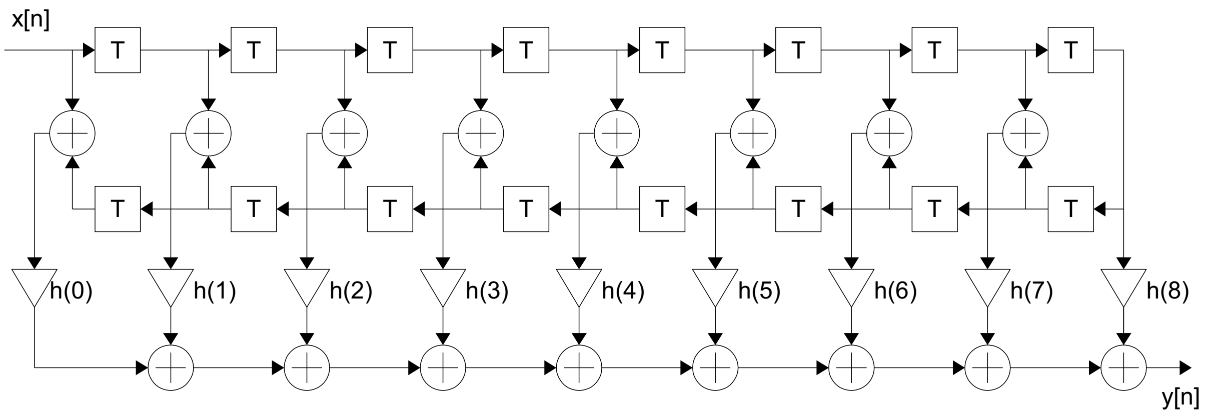


Figure 4: User-programmable FIR filter structure.

Example 3: A low pass filter is Y(z) = B(z) X(z) has the coefficients B = [57 92 -279 21 704 -720 -1163 4127 10784 4127 -1163 -720 704 21 -279 92 57]. This is given as SetUserLogicFilter([57 92 -279 21 704 -720 -1163 4127 10784]). The amplitude response of this filter is plotted in **Figure 5**.

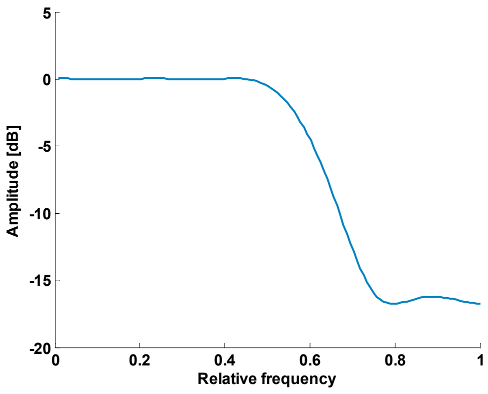


Figure 5: Example FIR filter amplitude response.

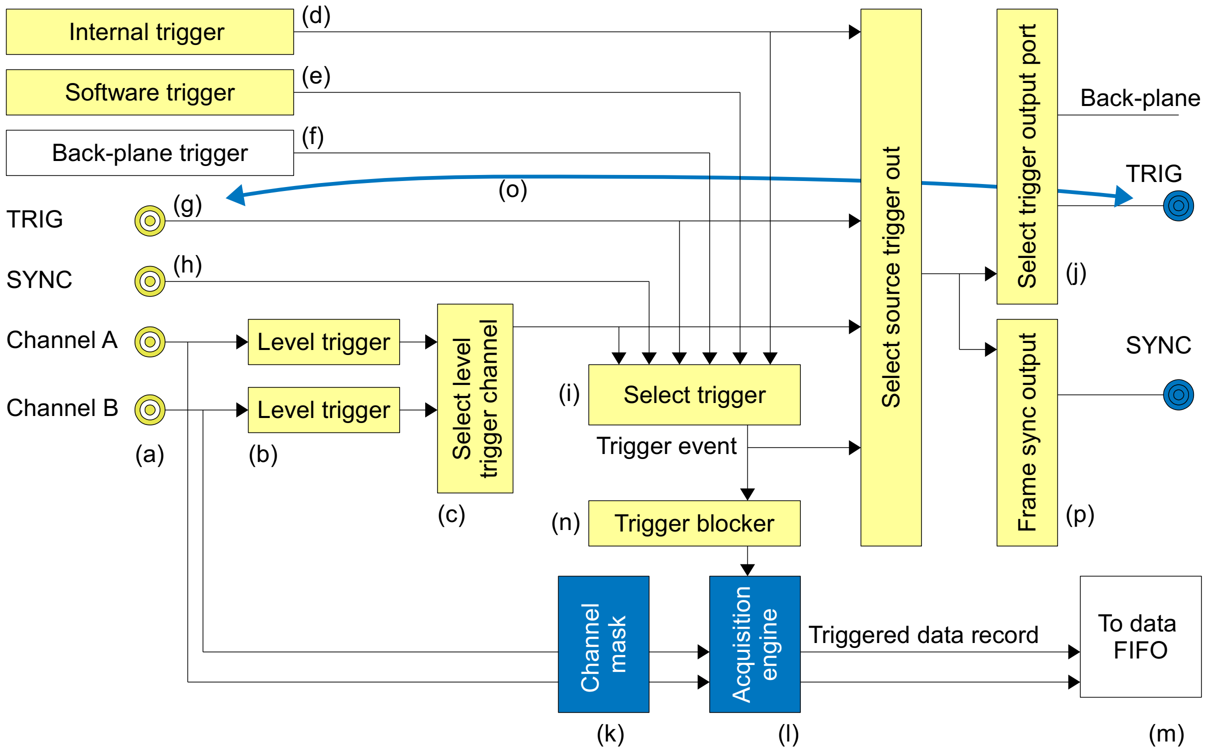
3.2.4 The FIR filter is placed in ADQ Development Kit

The FIR filter is placed in the ADQ7 Development Kit block user logic 1. The ADQ7 Development Kit is an additional tool for building custom firmware and "User Logic 1" is a space for this custom logic. By placing the FIR filter in this block serves the purpose of including it in the standard firmware FWDAQ and also make it available for the ADQ7 Development Kit user for modification. Therefore the switch ***BypassUserLogic*** has to be set correctly for using the filter.

4 TRIGGER

4.1 Trigger block diagram

The digitizer can be triggered in various ways with a number of different internal and external trigger sources. Selected events in the trigger module can also be output to trigger external equipment. The selection of trigger source is illustrated in **Figure 6**.



#	DESCRIPTION	USER COMMAND	REF
a	Connectors for external analog input signals. The number of channel vary on the different models and configurations.		
b	Each analog input is connected to a level trigger block.	*SetupLevelTrigger*	4.9
c	Select on which channel to trigger (when using level trigger).	*SetupLevelTrigger*	4.9
d	Internal trigger generator.	*SetInternalTriggerPeriod*	4.10
e	A software trigger is available for user control.	*SWTrig*	4.6
f	External trigger input from backplane in PXIe.		4.8
g	External trigger input on front panel connector TRIG.		4.7.1
h	External trigger input on front panel connector SYNC.		4.7.2
i	Select which type of trigger to activate.	*SetTriggerMode*	
j	Activate trigger output.	*SetupTriggerOutput*	4.11
k	Select which channels to record data from.	*SetStreamConfig*	
l	Acquisition engine creates a record from streaming data		7
m	Records are sent to data FIFO for transfer to the host PC		7
n	The trigger blocking function controls the flow of triggers to the acquisition engine.	*SetupTriggerBlocking*	4.4
o	Note that the trigger output and the external trigger input are physically the same connector on the front panel: TRIG.		
p	Frame sync is a function that can group triggers.	*SetupFrameSync*	4.11.2

Figure 6: Trigger source selection and setup.

4.2 Position of the trigger in the data

The trigger position relative to the data record is controlled by the parameters pretrigger and trigger delay.

The pretrigger buffer enables capturing data prior to the trigger event, **Figure 7**. Use the command ***SetPreTrigSamples*** to define the pretrigger.

The trigger delay postpone the start of the acquisition of the data record specified number of samples after the trigger event, **Figure 8**. Use the command ***SetTriggerHoldOffSamples*** to define the trigger delay.

The timing of the trigger is read from the record header (**Section 7.6**). The parameters ***TIME_STAMP*** and ***RECORD_START*** are explained in **Section 4.3.1**.

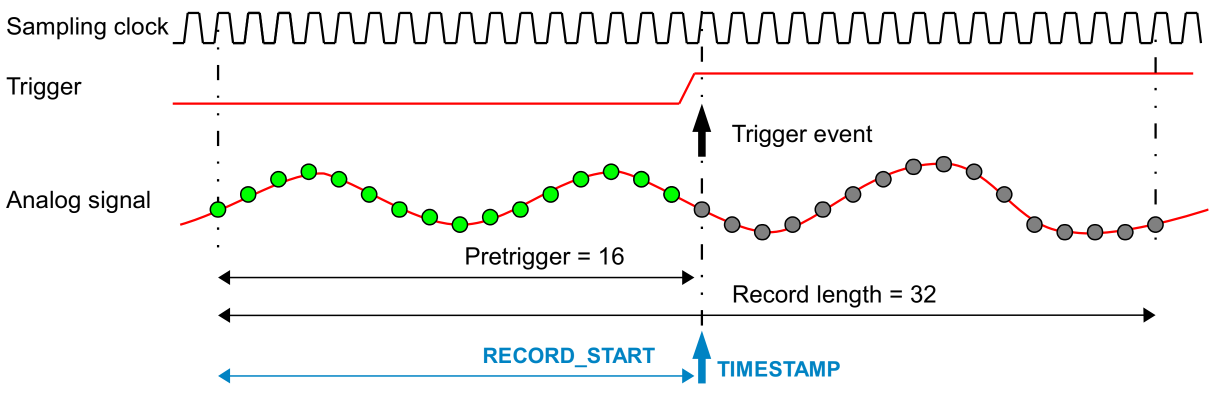


Figure 7: Pretrigger timing.

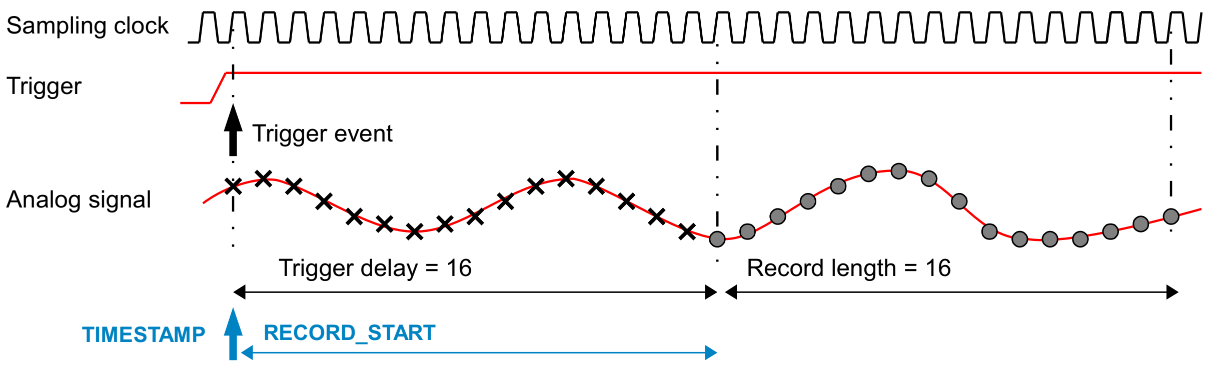


Figure 8: Trigger delay timing.

4.3 Timestamp

4.3.1 Timestamp definitions

The timestamp counter enables real-time measurement of a trigger event. It is used for tagging an event, sorting events in time or comparing timing between events.

The timestamp information consist of three parts, which uniquely defines the timing:

- ***TIME_STAMP*** measures the time of the trigger event relative to other trigger events.
- ***RECORD_START*** is the time between the trigger event and the start of the record. For a pretrigger, this is a negative value. When trigger delay is used, this is a positive value.

- ***SAMPLE_PERIOD*** is the length of a sample period. The sample period may vary with sample skip setting and clock frequency of the digitizer.

The ***TIME_STAMP***, ***RECORD_START***, and ***SAMPLE_PERIOD*** are measured in the unit ***TIME_BASE*** = 25 ps. See **Example 4** on how to use these parameters. These parameters are available in the record header, see **Section 7.6.1**.

The timestamp counter is based on the internal clock of the digitizer. The internal clock is based on the selected clock reference. The timestamp is thus also related to the clock reference. When the clock reference is phase-locked to an external source, the timestamp counter is running synchronized with the external source. On the other hand, if the digitizer is free running, the timestamp counter also free running. (See **Section 5** for all details about the clock system of digitzer.)

The timestamp counter measures the time from a reference time point to the trigger event. The reference time point is when the counter is started or reset. See **Section 4.3.2** for information on how to reset the timestamp counter.

Example 4: Assume an ADQ7WB sampling with a clock frequency at 5 GSPS. The pretrigger is set to 80 samples and the external trigger is used. The following parameters are returned:
TIME_STAMP *= 5005*
RECORD_START *= –645*
SAMPLE_PERIOD *= 8*
TIME_BASE *= 25 ps*

The time for the trigger was then
TRIGGER_TIME *=* ***TIME_STAMP*** *** ***TIME_BASE*** *= 125125 ps = 125.125 ns*

The time for the first sample in the record is
RECORD_TIME *= (****TIME_STAMP*** *+* ***RECORD_START****) ** ***TIME_BASE*** *= 109.000 ps = 109 ns*

The time between two samples are
SAMPLE_TIME *=* ***SAMPLE_PERIOD*** *** ***TIME_BASE*** *= 200 ps*

The time from the record start to the trigger is
RECORD_START *** ***TIME_UNIT*** *= –16.125 ns.*

The number of samples between the record start and the trigger event is
(***TRIGGER_TIME*** *–* ***RECORD_TIME*** *) /* ***SAMPLE_TIME*** *=*
| ***RECORD_START*** *| /* ***SAMPLE_PERIOD*** *= 80.625 samples*
This is the expected 80 samples set in the pretrigger and 5/8 sample in subsample precision in the external trigger.

4.3.2 Timestamp reset

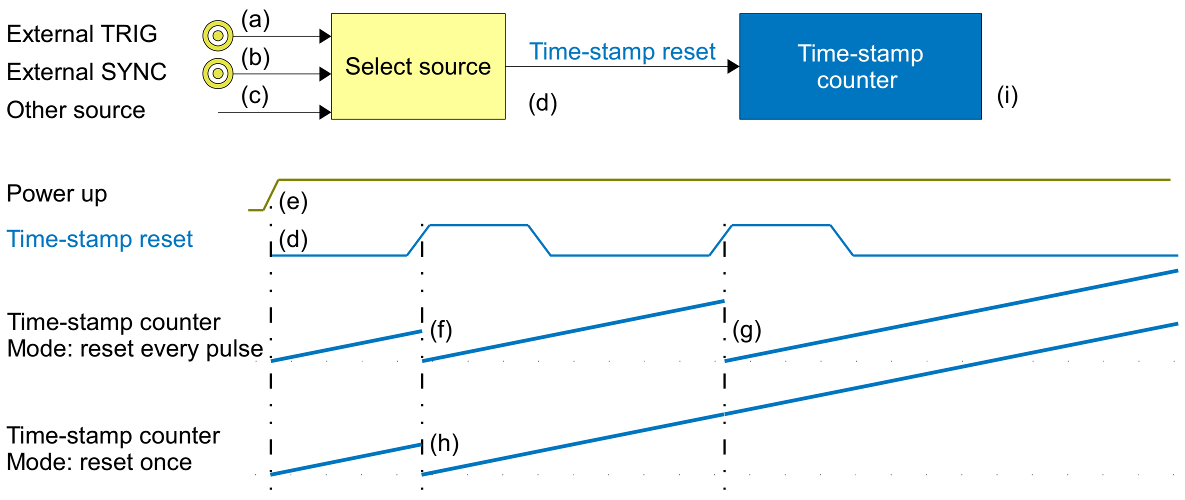
When powering up a system with many boards, the timestamp counter in each board will start. But the counters start at different times in different physical digitizers. There are four methods for resetting the timestamp and get a common time reference in all the digitizers in the system:

1. The timestamp counter is reset at power-up. This methods does not, however, have absolute precision, since the timing of the power up is not defined. In a multi-board system, the timestamp will differ between the boards.
2. With a software reference reset the user has full control of the reset procedure. A reference time point is created in the users application, which is used for aligning time-stamps in different units. After power-up the user runs a custom timestamp reset sequence including:
   - Apply a reference signal to all boards.
   - Trigger a record on the reference signal.
   - Read the time-stamps from the records and call this reference; ***TIME_STAMP_REFERENCE***.

- Start the experiment and subtract the timing reference from each record as *TIME_STAMP* = *TIME_STAMP_OF_RECORD* – *TIME_STAMP_REFERENCE*.

3. The third method is to apply an external trigger to reset the timestamp reset, **Figure 9**. This method has the possibility to synchronize several boards to full precision of the external trigger. See **Section 4.4**. The sequence of operation is:
   - *DisarmTimestampSync*
   - *SetupTimestampSync*
   - *ArmTimestampSync*

   The number of reset pulses are counted and the information is stored in the record header, **Section 7.6**. However, if there are no triggers accepted, there will be no record headers available. To verify that there is activity going on, the number of reset pulses can also be read from a register via *GetTriggerBlockingGateCount*.
4. The fourth method is to reset the timestamp with the sync signal, **Figure 9**. The difference between using the external trigger and the sync is that the external trigger has the a sample resolution while the sync timing resolution is controlled by the Data Clock in the FPGA. Note that the backplane triggers in –PXIe work in the same way as the sync signal.



#	DESCRIPTION	USER COMMAND	REF
a	External trigger input signal on front panel connector.		4.7
b	External sync input signal on front panel connector.		4.7
c	Other available sources (see *SetTriggerMode* for a list)		4.7
d	Select source for reseting timestamp.	*SetupTimestampSync* *DisarmTimeStampSync* *ArmTimeStampSync*	4.3.2
e	Timestamp counter value is reset at power-up of the digitizer.		4.3.2
f	Reset the timestamp counter on each pulse of the selected source. Timestamp is then measuring time relative the previous reset signal.		4.3.2
g			4.3.2
h	Reset the timestamp counter only on the first pulse of the selected signal. The external signal is then a systems synchronization signal.		4.3.2
i	The number of times the time stamp has been reset can be read from a register.	*GetTriggerBlockingGateCount*	4.3.2

Figure 9: Timestamp reset from external trigger.

4.4 Blocking triggers for synchronization

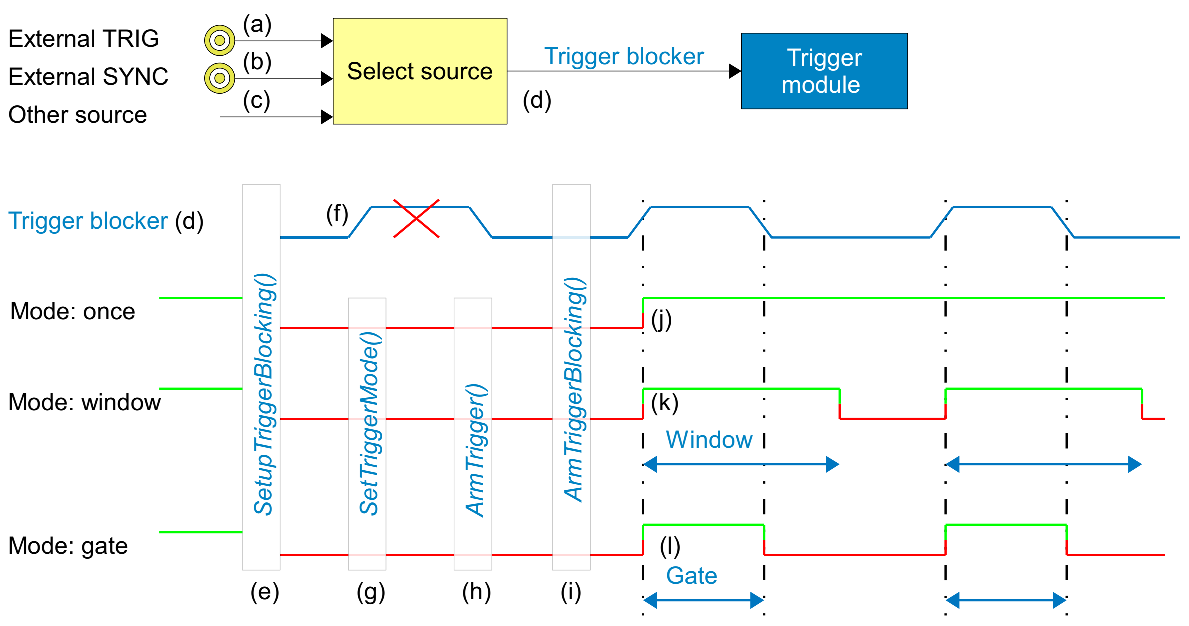
4.4.1 Function overview

In order to synchronize the acquisition to external equipment or to other ADQ digitizers, there is a mechanism for controlling the flow of triggers. The trigger blocking function allows the user to select when to activate incoming triggers, **Figure 10**. The basic function of this block is to use the SYNC signal to frame the trigger signals; for each period of the blocking function, a set of triggers are allowed and and framed by the blocking event. This creates groups of triggers that belong together. The modes of operation for trigger blocking are shown in **Figure 10** (j, k, l).

To avoid that the boards start to produce a large amount of records out of sync, all trigger events can be blocked until the triggers are released by a separate shared signal, **Figure 10** (d). By combining the trigger blocking and the timestamp reset, the timestamp is aligned to the start of the acquisition. The trigger blocker source can be most available trigger sources, **Figure 10** (a, b, c).

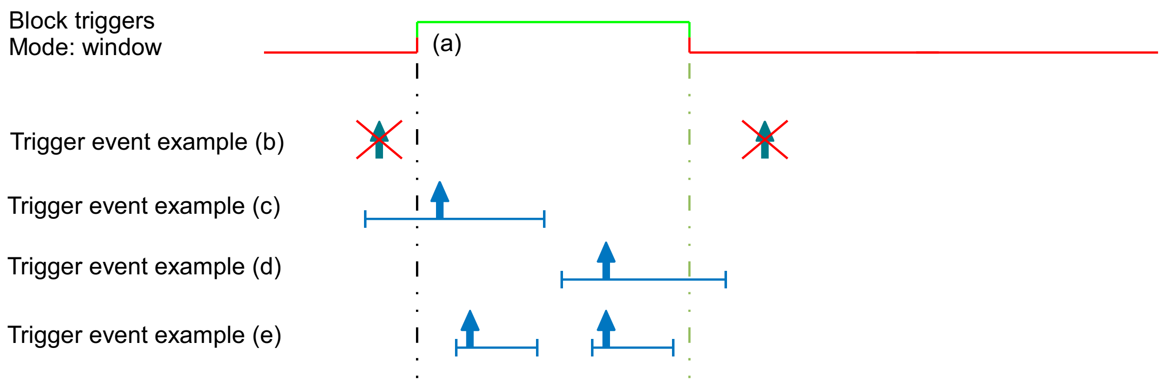
Note the order of the commands for activating triggers and trigger blockers, **Figure 10** (e, g, h, i).

Figure 11 illustrates how the triggers are accepted or rejected in the window mode.



#	DESCRIPTION	USER COMMAND	REF
a	External trigger input signal on front panel connector.		4.7
b	External sync input signal on front panel connector.		4.7
c	Other available sources (see *SetTriggerMode* for a list).		
d	Select source for blocking triggers.	*SetupTriggerBlocking*	4.4
e	Before activating the trigger blocking any selected trigger may pass.	*SetupTriggerBlocking*	
f	This signal is ignored as the trigger blocker is not armed		
g	Select trigger source	*SetTriggerMode*	
h	Start receiving triggers. Note that triggers are still blocked.	*ArmTrigger*	
i	The unblocking of triggers is armed and can be activated by (d).	*ArmTriggerBlocking*	
j	Triggers are blocked until the first accepted blocker signal.	*SetupTriggerBlocking*	4.4
k	The trigger blocker can also be set up with a window function where triggers are accepted or rejected during a user-defined window.	*SetupTriggerBlocking*	4.4.1
l	The trigger blocker can also be set up as a gate where triggers are accepted during a gated time set by signal (d).	*SetupTriggerBlocking*	4.4.1

Figure 10: Blocking and gating of triggers.


Block triggers
Mode: window
(a)
Trigger event example (b)
Trigger event example (c)
Trigger event example (d)
Trigger event example (e)

#	DESCRIPTION	USER COMMAND	REF
a	The trigger blocker in window or gate mode allows triggers during a certain period.	***SetupTriggerBlocking***	**4.4.1**
b	Example of rejected triggers outside the window.		
c	Trigger within the window is accepted and a data record is recorded. Note that the pretrigger starts before the window.		
d	Trigger within the window is accepted and a data record is recorded. Note that the record extends after the window.		
e	Several triggers within the same window.		

Figure 11: Trigger blocker examples.

4.4.2 Block triggers once

The mode for blocking triggers once is illustrated in **Figure 10** (j). This mode is used for starting the operation simultaneously in several units. The first time the trigger blocking signal is applied, the triggers are allowed through. Here is the motivation for this mode:

There is no way to broadcast a software command to several units. When setting up acquisition in several units, they will therefore be activated at different times. By using the trigger blocker, an electrical signal to all units can activate them simultaneously. The trigger blocking signal can be external or it can be generated internally using the bussed connections proposed in **Figure 16**.

4.4.3 Windowing triggers

The window mode for blocking triggers is illustrated in **Figure 10** (k). The edge of the trigger blocking signal is activating a window of user-defined length which allows triggers through. There is also a mode where triggers are blocked during the window.

The window mode can be used for two-dimensional triggering where, for example, the trigger signal is a point trigger and a sync signal is a line trigger.

4.4.4 Gating and windowing triggers

The gate mode for blocking triggers is illustrated in **Figure 10** (l). The length of the window where triggers are accepted is equal to the length of the trigger blocking signal.

4.4.5 Programming sequence for using trigger blocking

The order of commands is important when programming the trigger blocking. This is because the ADQ digitizer interact with other external equipment. This external equipment is synchronized to the digitizer through the setup procedure.

The setup of the functions has to be aligned with the expected operation. For example, by asserting the trigger blocking through the ***SetupTriggerBlocking*** command before setting up the acquisition, no triggers are let through before the digitizer is ready.

The function on the digitizer has to be activated (armed) in reverse order compared to the data flow. This means that one stage is set up to be prepared to receive data before the preceding stage is set up to generate data. This is especially important in streaming applications where the DRAM FIFO may overflow if the triggering is activated before the read-out to the host PC has started.

4.5 Trigger jitter

4.5.1 Trigger jitter definitions

The triggering operation is subject to two different types of jitter, **Figure 12**.

1. At the trigger input is a Gaussian distributed jitter which affects the timing of the incoming trigger signal edge. This jitter is called excess jitter and is caused by noise in the input stage. The RMS value of this excess jitter is 25 ps.
2. The actual sampling process causes a timing uncertainty. Since the trigger is sampled with the trigger clock, the time points for reading the trigger are discrete. The difference between the incoming physical trigger signal and the digital representation of the trigger is a stochastic variable with a rectangular distribution. The RMS value of such a process is ***TRIGGER_CLOCK_PERIOD***/sqrt(12). The highest resolution is achieved with an external trigger connected to the TRIG connector. ADQ7WB has a trigger clock at 20 GSPS, ***TRIGGER_CLOCK_PERIOD*** of 50 ps and a trigger jitter of 14 ps RMS (theoretical value), **Section 4.5.4**.

See **Table 3** for time resolution all the external trigger sources.

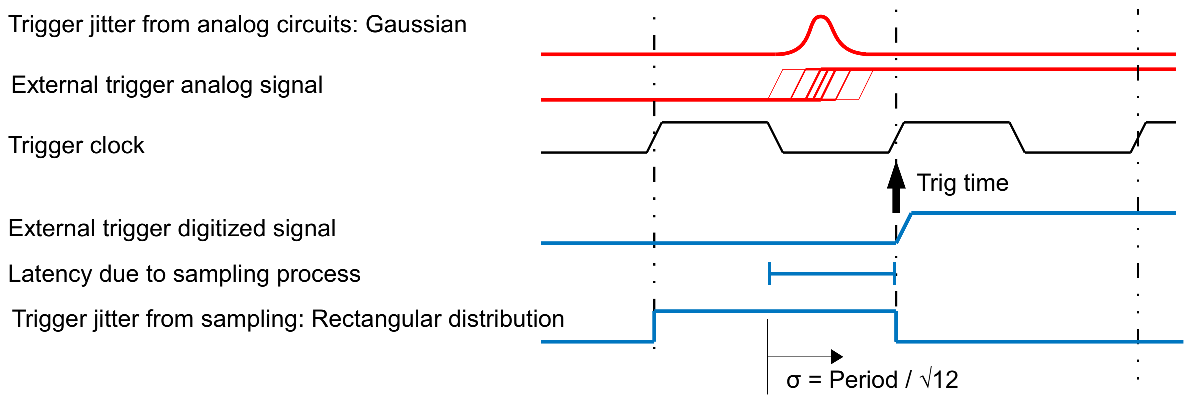

Trigger jitter from analog circuits: Gaussian
External trigger analog signal
Trigger clock
Trig time
External trigger digitized signal
Latency due to sampling process
Trigger jitter from sampling: Rectangular distribution
σ = Period / √12

Figure 12: Sources of jitter on the trigger signal.

4.5.2 Asynchronous triggering

If the trigger signal is not phase-locked to the reference clock it is called asynchronous. This trigger does not have a well-determined relation to the sampling clock and will appear at various positions within the sampling period. The time resolution of an asynchronous trigger connected to the TRIG input is set by the Trigger Clock (20 GHz). The time resolution for other triggers is determined by the Data Clock (312.5 MHz).

The asynchronous trigger will be exposed to both trigger sources from **Section 4.5.1**. These independent stochastic processes are added to 28 ps. See **Table 3** for time resolution of all the external trigger sources.

There are some advantages with the asynchronous trigger:

- Any pattern noise will be reduced in repeated measurements.
- The trigger resolution of 50 ps can be used for accurate timing calculations. The ***TIME_STAMP*** contains the information about the trigger time. See **Section 4.5.4**.

4.5.3 Synchronous trigger

A synchronous trigger is phase locked to the clock of the digitizer. The trigger source needs access to the clock reference of the digitizer. There are three ways to achieve this synchronization:

1. Output the internal clock reference of the ADQ and send it to the trigger source, **Section 5.8**.
2. Use the clock reference of the trigger source as clock reference for the ADQ, **Section 5.5**.
3. Use the internal trigger of the ADQ and output it to trigger the external equipment, **Section 4.10**.

When the trigger is phase-locked to the clock reference the timing is comparable to a digital signal which defines setup and hold time.

4.5.4 Extended trigger resolution

The basic sampling process maps the trigger to the sampling rate of the digitizer. There is also additional trigger time information available; Extended trigger resolution, **Figure 13**.

The Trigger Clock is operating at 20 GHz. This means that the time resolution of the trigger input TRIG is reduced to 50 ps.

Note: The extended trigger resolution is available on triggers connected to TRIG only.

The extended trigger information is included in the timestamp information, **Section 4.3**.

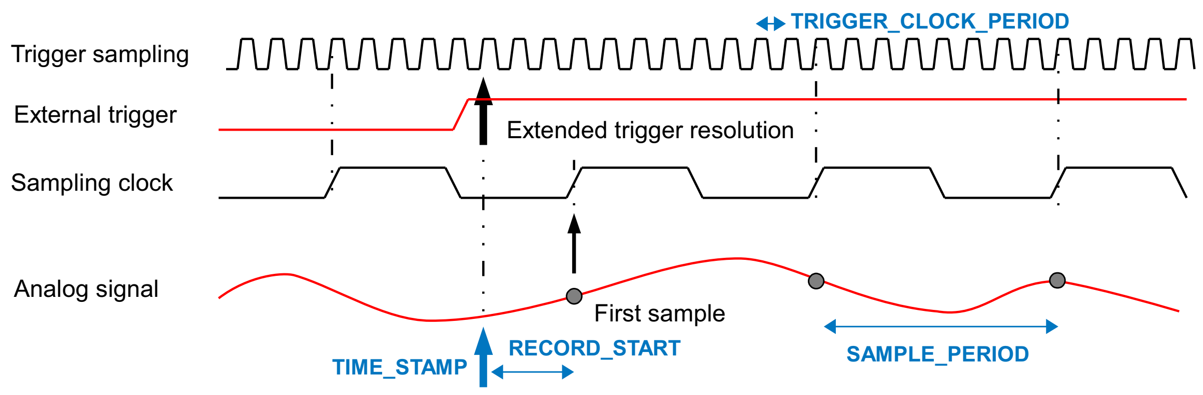


Figure 13: Extended trigger resolution timing for ADQ7 at 5 GSPS.

The position of the first sample is rounded up from the trigger position. The parameter ***RECORD_START*** tells where the trigger was. Referring to **Figure 13**, the ***RECORD_START*** parameter can have values in the range –75 ns up to +100 ns. A positive value means that the first sample is after the trigger. The given range is without pretrigger or trigger delay. With pretrigger or trigger delay, the ***RECORD_START*** will have a larger (absolute) value.

4.6 Software trigger

The software trigger is a user command that triggers the ADQ. This is for direct user control of the acquisition and is useful for looking at continuous signals where the timing of the trigger is not critical. The software trigger is sent through several layers of software and the time when it arrives to the digitizer cannot be predicted. However, the time when it actually arrived can be read from the time stamp in the record header, **Section 4.3**.

The software trigger may also be used for time-out function. This is a way to discover faults in the setup. When the device do not trigger for some reason within a certain time frame, a software command sequence may be sent and the data can be analyzed to find out what is wrong, **Example 4**.

Example 4: A time-out function using software trigger can be implemented like this:

1. Time-out occurs

2. *DisArmTrigger*
3. *SetTriggerMode*(“software trigger”)
4. *ArmTrigger*
5. *SWTrig*
6. Read data and analyze the situation

4.7 External Trigger Inputs

An external trigger is a dedicated signal on a dedicated input to the ADQ. There are several inputs for external trigger, **Table 3**.

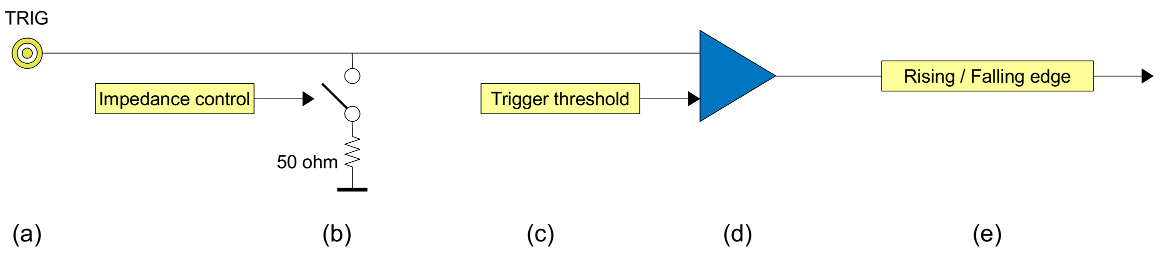
CONNECTOR	DESCRIPTION	TIME RESOLUTION	TOTAL JITTER	IMPEDANCE	TRIG LEVEL	REF
TRIG	External trigger on front panel.	50 ps	28 ps	50 Ω / 500 Ω	SW contr.	**4.7.1**
SYNC	Sync signal on front panel.	3.2 ns	0.9 ns	50 Ω / 500 Ω		**4.7.3**
STARB	Backplane trigger in PXIe systems. Requires trigger timing card.	3.2 ns	0.9 ns	PXIe standard	PXIe standard	**4.8.1**

Table 3: External trigger inputs.

4.7.1 External trigger TRIG front panel connector

The block diagram for the TRIG is shown in **Figure 14** and related parameters are listed in **Table 3**. The user can control the external trigger function for adapting it to the system in the following ways:

- The input impedance can be set in 50 Ω (default) or high impedance mode, see **Section 4.7.3**.
- Configure the threshold level.
- Set the trigger edge to rising or falling to adjust to the polarity of the trigger signal.



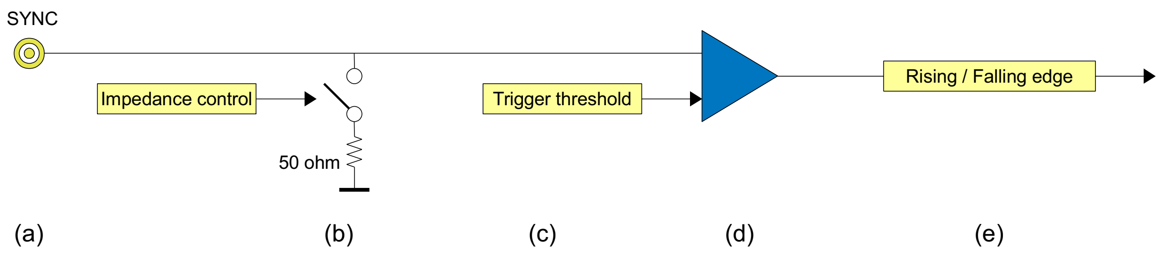
#	DESCRIPTION	USER COMMAND	REF
a	The input is available on an SMA connector on the front panel.		
b	The input impedance can be set as 50 Ω (default) or high impedance 500 Ω.	*SetTriggerInputImpedance*	4.7.3
c	The trigger threshold is 0.5 V (default) and user-controlled.	*SetExtTrigThreshold*	
d	High speed comparator.		
e	Select rising or falling edge.	*SetExternTrigEdge*	

Figure 14: External trigger on front panel.

4.7.2 External trigger SYNC connector

The block diagram for the SYNC input is shown in **Figure 15** and related parameters are listed in **Table 3**. The user can control the SYNC function for adapting it to the system in the following ways:

- The input impedance can be set in 50 Ω (default) or high impedance mode, see **Section 4.7.3**.
- Configure the threshold level.
- Set the trigger edge to rising or falling to adjust to the polarity of the trigger signal.



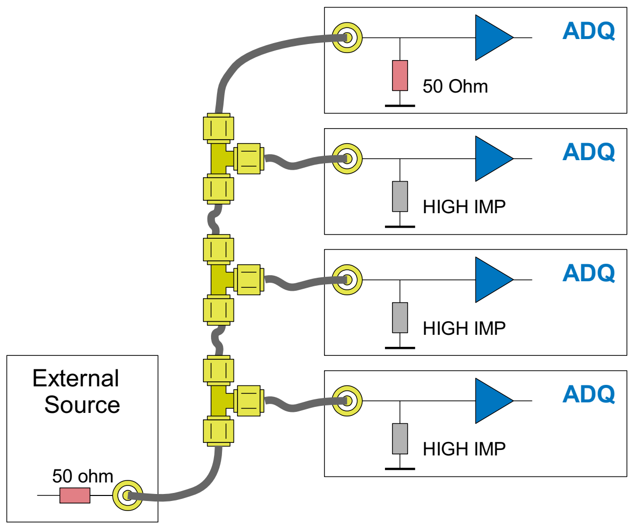
#	DESCRIPTION	USER COMMAND	REF
a	The input is available on an SMA connector on the front panel.		
b	The input impedance can be set as 50 ohm (default) or high impedance.	*SetTriggerInputImpedance*	4.7.3
c	The trigger threshold is 0.5 V (default) and user-controlled.	*SetExtTrigThreshold*	
d	High speed comparator.		
e	Select rising or falling edge.	*SetExternTrigEdge*	

Figure 15: Using SYNC as trigger input.

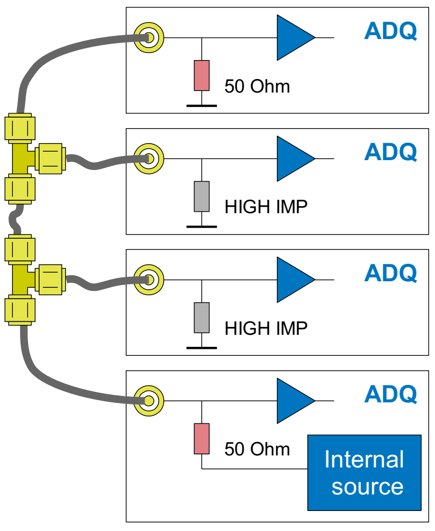
4.7.3 Driving the external TRIG/SYNC signal by controlling input impedance

The TRIG/SYNC input impedance is by default 50 Ω. The trigger is optimized for systems where the trigger source output impedance is 50 Ω and the cables has a characteristic impedance of 50 Ω. This setup results in an optimal high-frequency response and low reflections, which is important for precise timing.

However, in a high fan-out situation, where a trigger source has to drive many nodes, the load can be too high. The trigger input can then be set in a high impedance mode and a bussed connection can be used, **Figure 16**. In **Figure 16** (a) an external source is driving the array of ADQ digitizers. In **Figure 16** (b) one of the ADQ digitizer is used as a master and is driving signals to the array of ADQ digitzers.One has to be careful with the trigger distribution network to handle the reflections. If the trigger is periodic, reflections are less critical and can be handled. For more information on reflections, see application note **[5]**.



(a) External source



(b) ADQ internal source

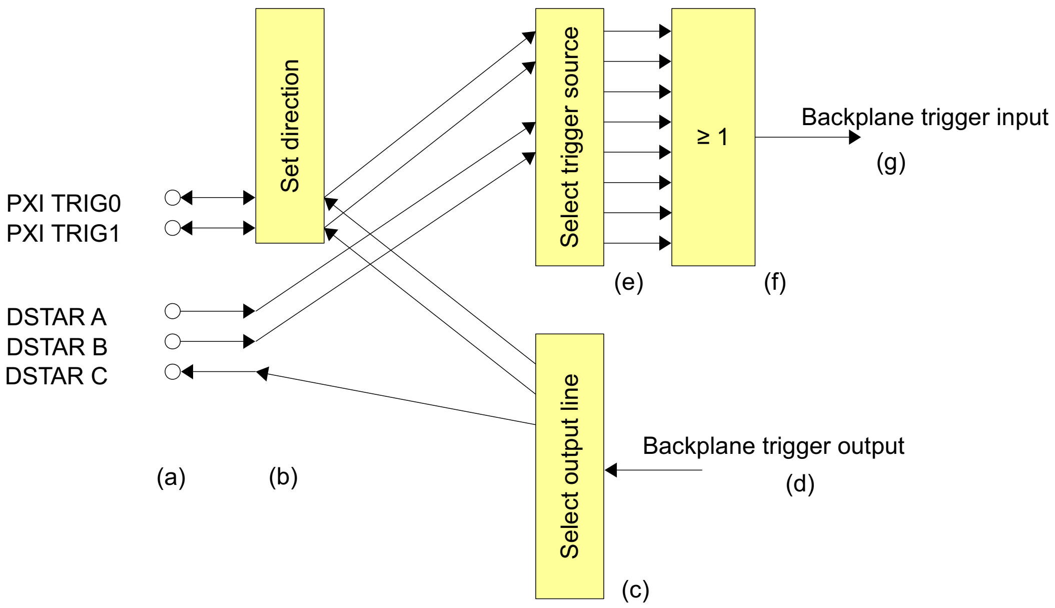
Figure 16: Bussed connections

4.8 External trigger in the backplane

4.8.1 PXIe interface

There are an external trigger in the backplane of the PXIe version of ADQ7WB. The DSTAR signals are dedicated matched trigger lines from the system timing slot. To use these triggers, a dedicated timing generation board has to be used in the system timing slot. The TRIG bus is a general bus in the back-plane which can be used for triggering. The ADQ support connection to port 0 and port 1 of that bus.

The backplane trigger support both input and output triggers. These operations are independent and can be used simultaneously.

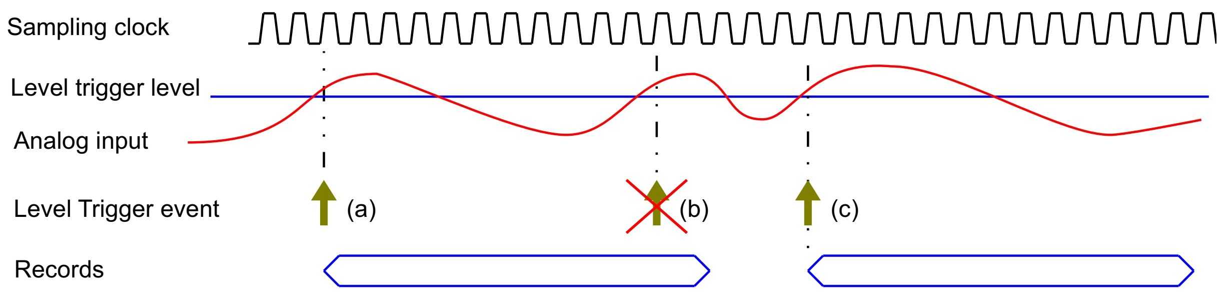


#	DESCRIPTION	USER COMMAND	REF
a	Backplane Trigger bus and DSTAR connections		
b	Set direction for each port in the backplane	*SetDirectionPXI*	
c	Output: Select output port for trigger output signal	*SetupTriggerOutput*	**4.11**
d	Output: This is the source for the trigger output signal	*SetupTriggerOutput*	**4.11**
e	Input: Select port for trigger sources	*SetTriggerMaskPXI*	
f	Input: The backplane trigger signal is OR:ed from the selected ports		
g	Input: this is the backplane trigger to the trigger module	*SetTriggerMode*	**4.1**

Figure 17: Trigger in the PXIe backplane.

4.9 Level trigger

The level trigger allows data-driven acquisition. When the data on a selected channel crosses the trigger level, all channels on the ADQ is triggered, **Figure 18**. The level trigger is set to trigger on rising or falling edge. Here, rising edge is illustrated.



#	DESCRIPTION	USER COMMAND	REF
a	When the signal passes the trigger level, a trigger event is generated and the first record is captured.	*SetupLevelTrigger*	**4.9.1**
b	During the record, incoming triggers are ignored.		
c	When the signal passes the trigger level, a trigger event is generated and the second record is captured. Re-triggering is also controlled by a hysteresis function.	*SetupLevelTrigger*	**4.9.2**

Figure 18: Level trigger introduction.

4.9.1 Setting the level trigger level

The level is given in digital codes:

LEVEL_CODE = ANALOG_TRIGGER_LEVEL / (ACTUAL_ANALOG_RANGE / 2) * 2^15 **(6)**

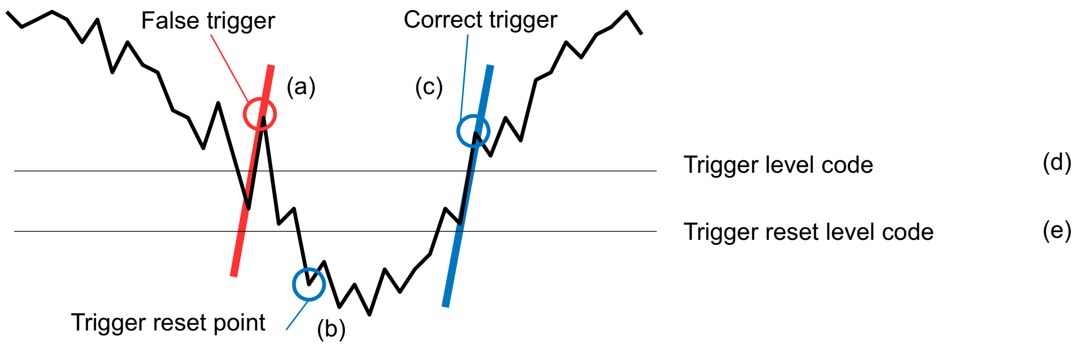
The ***ACTUAL_ANALOG_RANGE*** is the analog full scale range.

4.9.2 Controlling noise sensitivity

The level trigger is sensitive to noise since it can detect a step as small as one digital code. This can cause unwanted triggering. The noise sensitivity is controlled by a hysteresis function, **Figure 19**. After triggering, the signal has to cross a reset level before it can trigger again.

Setting the reset level far from the trigger level will give a robust trigger.

Setting the reset level close to the trigger level will give a sensitive trigger.



#	DESCRIPTION	USER COMMAND	REF
a	The level trigger has a hysteresis function to avoid false triggering on noise.		**4.9.2**
b	When the signal passes below the ***RESET_LEVEL_CODE***, the ADQ may trigger again.		
c	Trigger position is the first sample above the trigger ***LEVEL_CODE***		
d	Set the trigger ***LEVEL_CODE***.	***SetupLevelTrigger***	**4.9.1**
e	Set the trigger ***RESET_LEVEL_CODE***.	***SetupLevelTrigger***	**4.9.2**

Figure 19: Trigger reset level.

4.10 Internal trigger

The internal trigger generates a periodic trigger signal. The internal trigger period is specified in number of samples.

Note that if the internal trigger signal used as a trigger output, the actual output signal is updated at the Data Clock rate. This may appear as jitter on the output. It is recommended to use a period that is a multiple of the Data Clock is the internal trigger shall drive external equipment.

4.11 Trigger output

The trigger output signal is intended for triggering external equipment connected to the ADQ to build a synchronized system. It can also be used for indicating that a trigger event occurred in the digitizer. The trigger output is updated at the rate of the Data Clock.

The trigger output function consists of two parts:

- The first part selects the source of the trigger output signal.
- The second part selects the physical output port for the trigger output signal.

Note that the trigger output is the same physical TRIG connector as the external trigger input.

4.11.1 Trigger output port selection

The trigger output port is selected to one of these ports:

- Trigger connector on the front panel. Note that the trigger output is the same physical TRIG connector as the external trigger input.
- PXIe backplane triggers, **Section 4.8.1**.

The availability of these ports depends on the interface option for the digitizer.

4.11.2 Frame sync output on SYNC connector

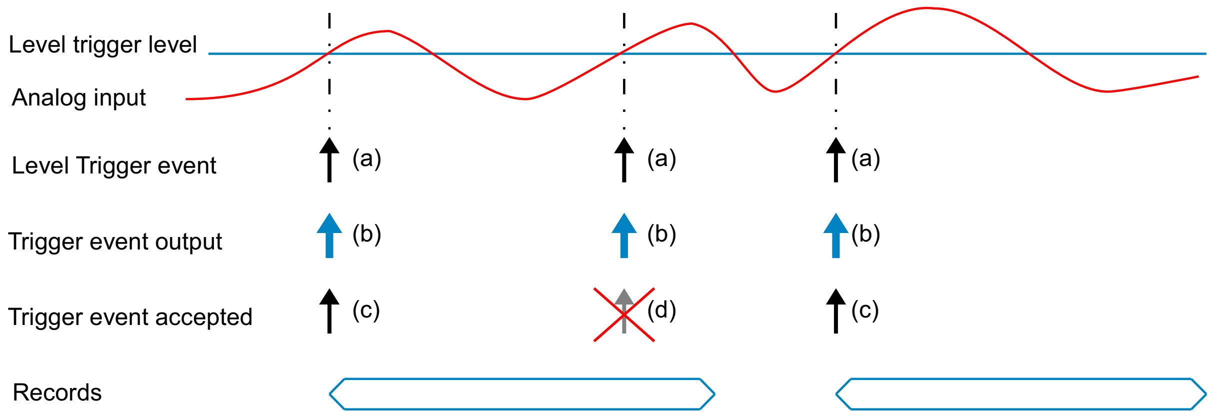
The frame sync feature enables grouping of trigger signals into frames or blocks or lines. The name for this feature relate to the actual application. This function can, for example, be used in scanning threedimensional measurements where a record is the first dimension, the trigger is the second and the frame sync is the third dimension.

The frame sync count triggers and output a signal at a certain period. The period is measured in trigger events.

4.11.3 Trigger event indicator

The trigger output can be used for indicating that trigger event occurred in the digitizer. A trigger event is some event that causes the digitizer to trigger. When used as a trigger event indicator all trigger events are presented at the trigger output port, **Figure 20** (a, b) illustrates how trigger events appear at the trigger output. The trigger events at **Figure 20** (c) will generate a record. The event **Figure 20** (d), however, occurs while the recording is ongoing and will thus not generate a record.

Note that the timing of the trigger event in the capturing of a record is based on the Sample Clock but the trigger output is based on the Data Clock.



#	DESCRIPTION	USER COMMAND	REF
a	Level trigger generates trigger events.		
b	Trigger output of the the level trigger events.	*SetupTriggerOutput*	
c	Accepted trigger events create records.		
d	This trigger event is falling within the recording and is discarded by the acquisition control. This trigger will not create any new record.		

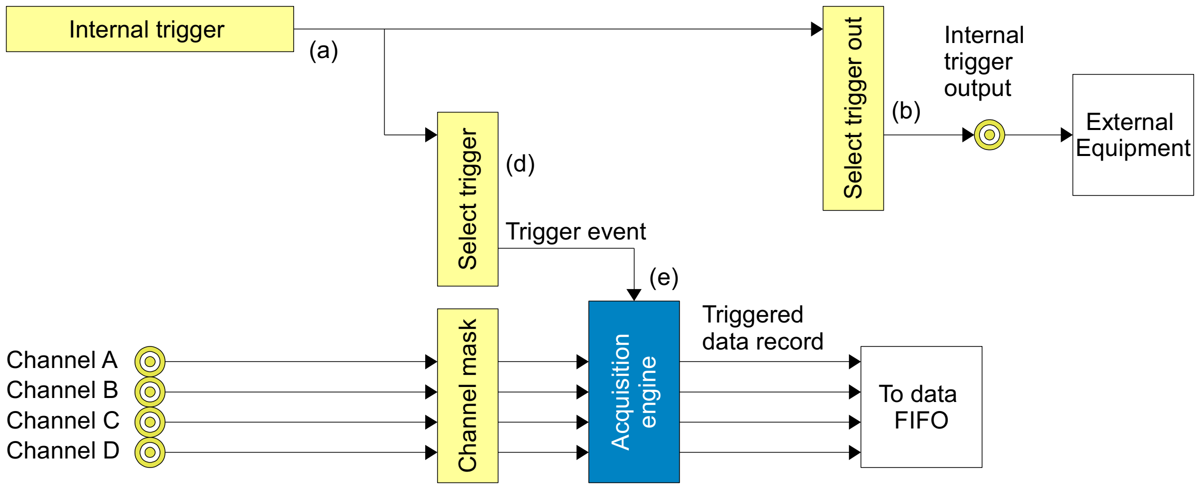
Figure 20: Trigger event output for a level triggered system.

4.11.4 Triggering external equipment with internal trigger

Triggering external equipment and the digitizer with the internal trigger may be done in two ways; internally, **Figure 21** and externally, **Figure 22**.

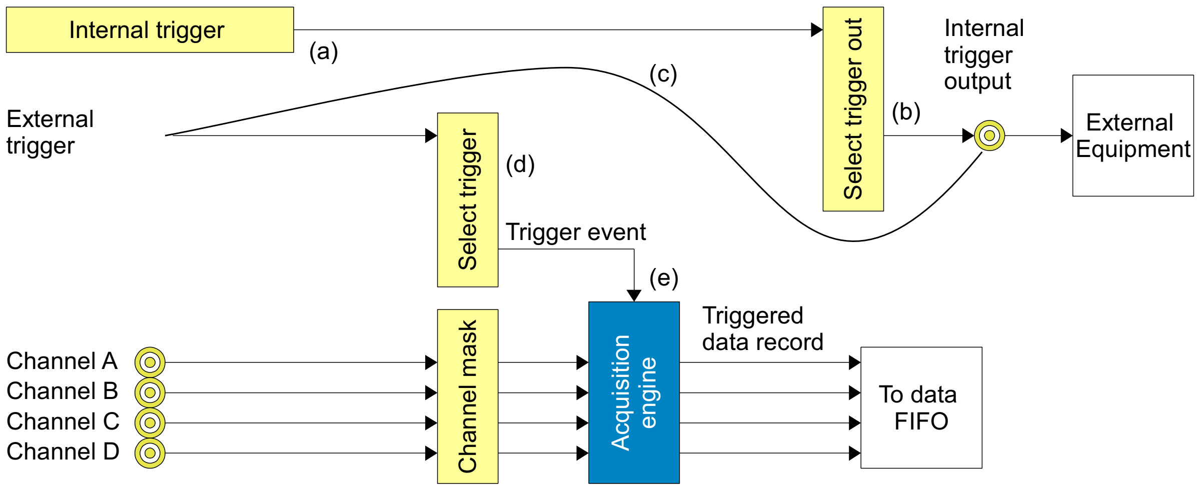
The internal connection is preferred when the trigger is only used for triggering the digitizer. The triggering is then done inside the FPGA as a logical signal and the trigger time is guaranteed to be exact on the expected sample.

The external connection is preferred when the trigger is used for triggering both the digitizer and the external equipment. Then the ADQ will listen to the same physical signal as the external equipment is using.



#	DESCRIPTION	USER COMMAND	REF
a	Internal trigger generator	*SetInternalTriggerPeriod*	4.10
b	Select internal trigger as output	*SetupTriggerOutput*	4.11
c	–	–	–
d	Select **internal** trigger as trigger source	*SetTriggerMode*	
e	Acquisition engine creates a record from streaming data		7

Figure 21: Internal routing of internal trigger.



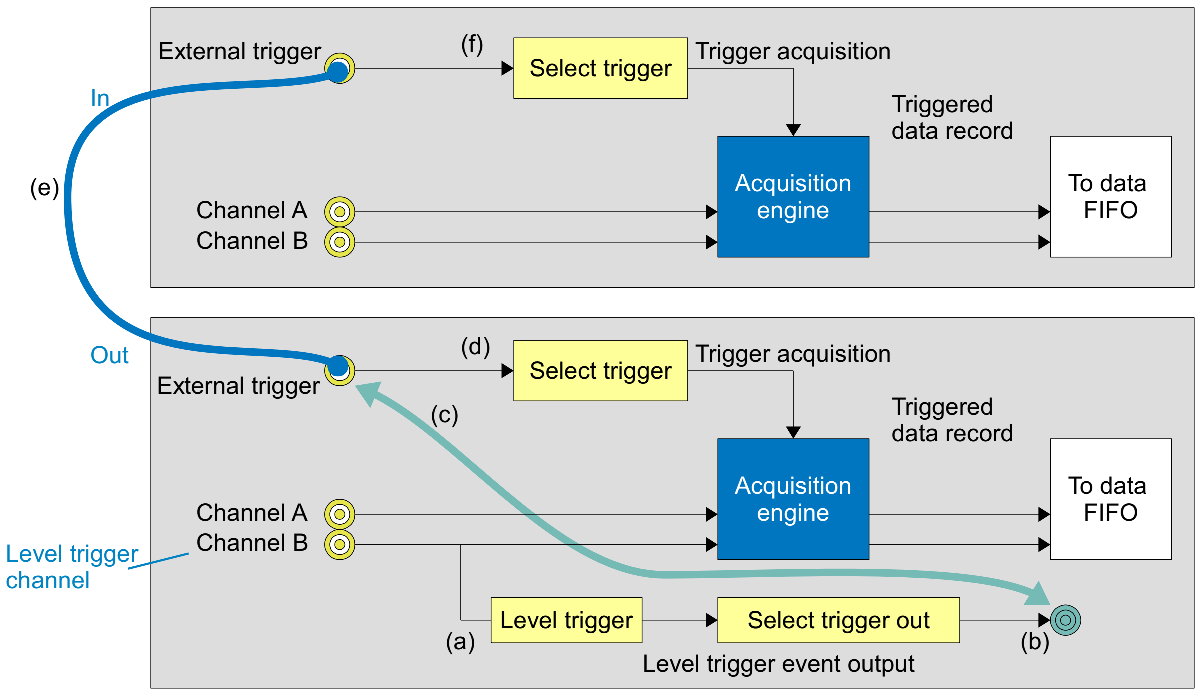
#	DESCRIPTION	USER COMMAND	REF
a	Internal trigger generator	*SetInternalTriggerPeriod*	4.10
b	Select internal trigger as output	*SetupTriggerOutput*	4.11
c	The trigger output and the external trigger input are electrically connected inside the digitizer		
d	Select **external** trigger as trigger source	*SetTriggerMode*	
e	Acquisition engine creates a record from streaming data		7

Figure 22: External routing of internal trigger.

4.11.5 Distributing level trigger

Multiple boards may be triggered by a level trigger on one channel in one of the digitizers. This mode is intended for systems with a reference event on one channel that starts the acquisition on all channels.

It may also be used for designing a high precision trigger. The trigger event on channel D in **Figure 23** is recorded on the channel D. The trigger timing is then calculated with high precision using interpolation.



#	DESCRIPTION	USER COMMAND
a	Select a channel as level trigger.	***SetupLevelTrigger***
b	Select level trigger as trigger out.	***SetupTriggerOutput***
c	Internal hardware connection trigger out to trigger in (shared SMA connector).	
d	Select external trigger for triggering	***SetTriggerMode***
e	External cable connection to other digitizers.	
f	Select external trigger for triggering.	***SetTriggerMode***

Figure 23: Distributing level trigger.

5 CLOCK

5.1 Clock domains

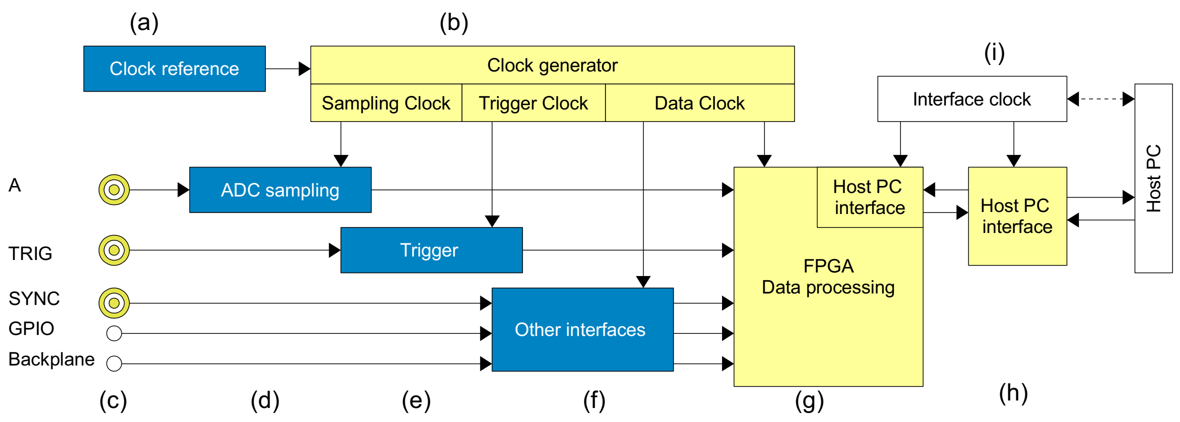
Different parts of the digitizer system operate on different clocks. The core of the clocking system is the clock reference. This is the phase and frequency reference of the digitizer system. It is possible to use different clock reference sources to meet the requirements of different applications.

The sampling clock is generated based on clock reference and drive the ADCs. This clock is important for the signal quality since any timing deviation (jitter) will impact the actual time of the sample.

The trigger signal is sampled by a separate clock at a higher frequency than the sampling clock. This is to achieve a good timing resolution of the trigger. This frequency is the highest in the digitizer and is also the base for the timestamp.

The FPGA cannot operate at the high rate of the sampling clock. Instead the data processing operate on a derived lower frequency denoted Data Clock and several samples are processed in parallel to maintain the throughput. The Data Clock is also synchronized to the clock reference.

Finally the host PC interface also operate on a different clock. The PCIe system clock is provided from the PCIe bus. This part of the design is not phase-locked to the sampling clock.



#	DESCRIPTION	TYPICAL FREQUENCIES	REF
a	Clock reference.	10 MHz	5.2
b	Clock generator that generates several frequencies.		5.6
c	Various interfaces.		4.7
d	The ADC operates on sampling clock.	5GHz	
e	The trigger operates on the trigger clock.	20 GHz	4
f	Other interfaces operate on the Data Clock.	312.5 MHz	4 6
g	The FPGA operate on the Data Clock.	312.5 MHz	
h	The host PC operate on an independent clock.	125 MHz	
i	The clock generation for the host interface depend on interface type.		

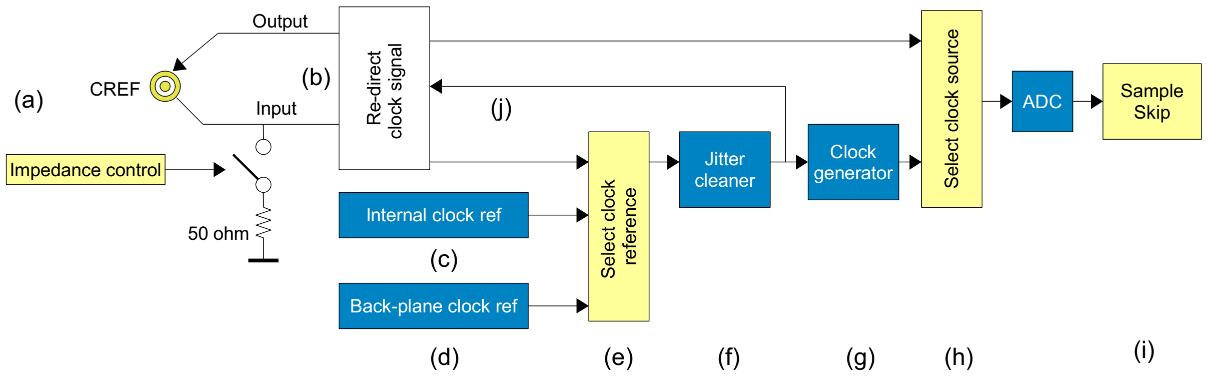
Figure 24: Clock system overview

5.2 Flexible clock network

The preferred clock method is a systems design parameter and teh digitizer supports many options. The clock system of the digitizer consist of two key parts; the clock reference and the clock generator. The clock reference is a low frequency (10 MHz) high-precision and high-stability signal that sets the accuracy of the clocks in the digitizer. The clock generator translates the frequency of the clock reference to the sampling clock rate. The digitizer supports its specified sample rate only. This sample rate can be tuned to allow phase locking to external equipment.

To reduce the sample-rate, a sample skip function is available.

Block diagrams of the clock network for ADQ7WB is given in **Figure 25**.



#	DESCRIPTION	USER COMMAND	REF
a	The SMA connector is common for external clock reference input, external clock input and external reference output.	*SetClockInputImpedance*	**5.5**
b	Selecting the usage of the external connector is implicit from setting up the clock system.		
c	The internal clock reference is a high performance VCTCXO.		**5.4**
d	PXIe support clock reference from the backplane		**5.5**
e	Select which clock reference source to use.	*SetClockSource*	
f	The external clock reference is cleaned from jitter.		
g	The internal clock generator.		**5.6**
h	Select which clock source to use	*SetClockSource*	**5.7**
i	Reduce sample rate with sample skip.	*SetSampleSkip*	**5.9**
j	Turn on clock reference output. Note that the connector is shared with the clock reference input.	*EnableClockRefOut*	**5.8**

Figure 25: ADQ7 clock network.

5.3 Front panel SMA connector

The front panel SMA connector is used for external clock reference input, direct external sampling clock input or clock reference output. The input impedance is 50 Ω to match the cable impedance. It can be set to high impedance (200 Ω) for using a bussed clock distribution (see **Figure 16**).

The high impedance mode allow that one digitizer can act as clock reference for two other digitizers. If the source of the clock reference has a high power, more digitizers can be connected in the bussed networks.

5.4 Internal clock reference

The internal clock reference is a high accuracy VCTCXO at 10 MHz.

5.5 External clock reference

The free running internal clock reference of the digitizer offers high precision and is suitable for most measurements. However, for some applications an absolute phase-lock to other parts of the system may be necessary. To support that, the ADQ offers several options to accept an external clock reference. A long-term phase stability to other equipment is then guaranteed.

The connector on the front panel accepts a clock reference from external equipment. The clock reference quality is improved in a jitter cleaning circuitry. To match the tuning of the jitter cleaning circuitry the clock reference has to be 10 MHz.

The PXI Express allows clock reference input from the backplane, and can then benefit from the infrastructure of the chassis.

5.6 Internal clock generator

There is an internal high quality clock generator that is used for generating the Sampling Clock for the A/D converters. The data and trigger clocks are also generated by this clock generator.

5.7 External clock

If the system is designed with an external high quality signal it may be used for clocking the ADQ. The external clock frequency must be 2.5 GHz.

If an external clock source is used, all the internal clocks are generated from that to maintain the phase and frequency ratio.

5.8 Clock reference output

In addition to the synchronization solution with an external clock reference source, the digitizer can also act as master and output its clock reference to external equipment. The selected clock reference source will then be present at the clock connector on the front panel. Note that the connector is shared with clock input.

5.9 Sample skip

The data rate out from the A/D converter is set by the sample rate of the digitizer. The data rate can be reduced by the sample skip function. Setting the sample skip factor to, for example, 4 means that every 4th sample is kept and the others are discarded. This will efficiently reduce the data rate.

Note that there is no decimation filter in this function. Decimation filter is available in the –FW4DDC firmware option.

6 GPIO

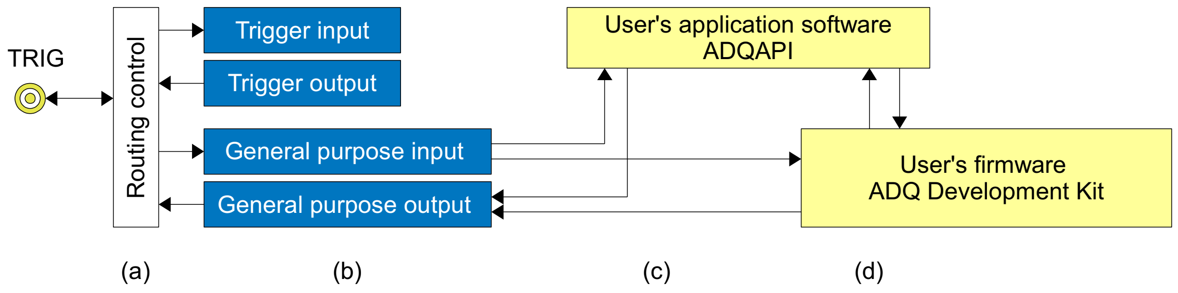
The General Purpose Input and Output (GPIO) are digital signals available from the front panel of the digitizer. The user assigns a function to these pins, either in the firmware through the ADQ Development Kit or from software.

The digitizer offers several levels of GPIO

1. GPIO through multiple usage of TRIG and SYNC connectors, **Section 6.1**.
2. Dedicated GPIO connector, **Section 6.5**.

6.1 GPIO with TRIG and SYNC

The connectors for TRIG and SYNC can be used as GPIO. The process is illustrated in the block diagram in **Figure 26**.



#	DESCRIPTION	USER COMMAND	REF
a	The external pin is automatically connected to the activated function.		**6.2**
b	The GPIO input function always reads the state of the pin. The GPIO output function is activated by the user.	***SetDirectionGPIOPort***	**6.2** **6.3**
c	The user may access the pin by reading and writing from the software.	***ReadGPIOPort*** ***WriteGPIOPort***	**6.2** **6.3**
d	The user may build firmware in the ADQ Development Kit for real-time interaction with the GPIO signals.Then GPIO is accessed through register access commands.	***ReadUserRegister*** ***WriteUserRegister***	**6.4**

Figure 26: Using front panel connector TRIG as GPIO. The connector SYNC operates in the same way.

6.2 Using GPIO as a trigger

The GPIO can be used for sending a trigger command from the application software to an external device. In such a situation this GPIO signal can also be used for triggering the digitizer itself synchronous to the external device. This is possible since the external trigger input logic always listen to the signal on the TRIG connector. The following example illustrate how to trigger the digitizer and an external device through GPIO function on the TRIG connector.

Example 3: Triggering the digitizer and an external device with GPIO

1. Connect a cable from the TRIG connector to the external device.
2. ***SetTriggerMode***(“external trigger”) /* This will activate the trigger module to listen to TRIG”
3. ***SetDirectionGPIOPort***(“output”) /* This enables output on TRIG connector*/
4. ***ArmTrigger***
5. ***WriteGPIOPort***(“1”) /* This sends a signal on the TRIG connector that triggers the devices*/

The GPIO input function always listen to the trigger pin. This means that the external trigger pin value can always be read from the GPIO function ***ReadGPIOPort***.

6.3 Output

The output is activated through the software command ***SetDirectionGPIOPort*** and the signal level is set by ***WriteGPIOPort***. The input function ***ReadGPIOport*** will then return the output level.

6.4 GPIO in ADQ Development Kit

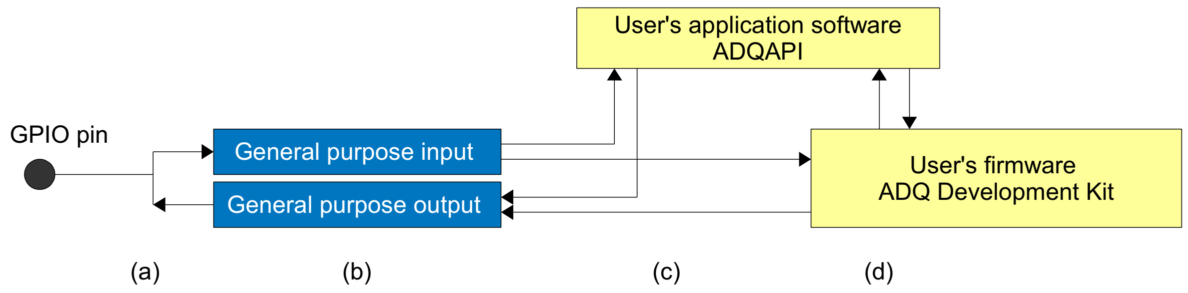
The GPIO signals from TRIG and SYNC are available in the ADQ Development Kit for real-time interaction with the signal flow.

6.5 Dedicated GPIO connector on form factor –PCIe or –PXIe

In addition to the GPIO functions of TRIG and SYNC signals there is a HD-DSUB connector with additional GPIO connections. The GPIO signals are available in the software development kit for integration in the user's application software. The signals are also available in the ADQ Development Kit for integration in real-time firmware.

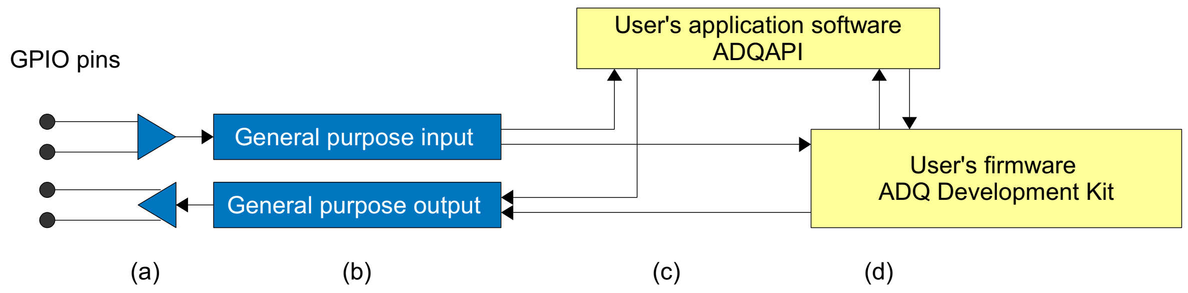
The HD-DSUB on the front panel is a 44-pin connector with 12 single-ended individually controlled bi-directional signal, 2 LVDS inputs, 4 LVDS outputs and 2 LVDS clock capable inputs. The GPIO connections are described in **Figure 29**.

The structure for the single-ended pins is shown in **Figure 27** and for the LVDS pins in **Figure 28**.



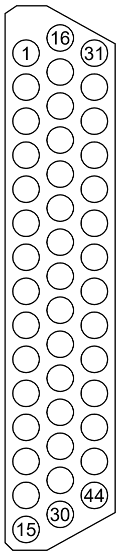
#	DESCRIPTION	USER COMMAND
a	There are 12 individually controlled external GPIO signals.	
b	GPIO input function always reads the state of the pin. The GPIO output function is activated by the user.	***SetDirectionGPIOPort***
c	The user may access the pin by reading and writing from the software.	***ReadGPIOPort*** ***WriteGPIOPort***
d	The user may build firmware in the ADQ Development Kit for real-time interaction with the GPIO signals.Then GPIO is accessed through register access commands.	***ReadUserRegister*** ***WriteUserRegister***

Figure 27: ADQ7 structure of each of the 12 single-ended pins available with options –PCIe and –PXIe.



#	DESCRIPTION	USER COMMAND
a	There are 2 LVDS inputs, 2 LVDS clock capable inputs, and 4 LVDS output signals in the connector.	
b	The direction is fixed for these pins	
c	The user may access the pin by reading and writing from the software.	***ReadGPIOPort*** ***WriteGPIOPort***
d	The user may build firmware in the ADQ Development Kit for real-time interaction with the GPIO signals.Then GPIO is accessed through register access commands.	***ReadUserRegister*** ***WriteUserRegister***

Figure 28: ADQ7 structure of each of the 12 single-ended pins available with options –PCIe and –PXIe.



1	GPDIC0_P	LVDS input / clock	16	GND	31	GPIO0	Single-ended 3.3 V
2	GPDIC0_N	LVDS input / clock	17	GND	32	GPIO1	Single-ended 3.3 V
3	GPDI1_P	LVDS input	18	GND	33	GPIO2	Single-ended 3.3 V
4	GPDI1_N	LVDS input	19	GND	34	GPIO3	Single-ended 3.3 V
5	GPDI2_P	LVDS input	20	GND	35	GPIO4	Single-ended 3.3 V
6	GPDI2_N	LVDS input	21	GND	36	GPIO5	Single-ended 3.3 V
7	GPDIC3_P	LVDS input / clock	22	GND	37	GPIO6	Single-ended 3.3 V
8	GPDIC3_N	LVDS input / clock	23	GND	38	GPIO7	Single-ended 3.3 V
9	GPDO4_P	LVDS output	24	GND	39	GPIO8	Single-ended 3.3 V
10	GPDO4_N	LVDS output	25	GND	40	GPIO9	Single-ended 3.3 V
11	GPDO5_P	LVDS output	26	GND	41	GPIO10	Single-ended 3.3 V
12	GPDO5_N	LVDS output	27	GND	42	GPIO11	Single-ended 3.3 V
13	GPDO6_P	LVDS output	28	GND	43	VDD	3.3V 250mA
14	GPDO6_N	LVDS output	29	GND	44	VDD	3.3V 250mA
15	NC	NC	30	GND			

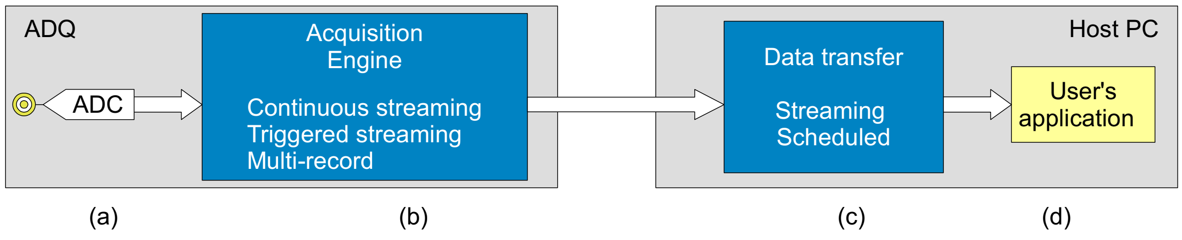
Figure 29: ADQ7 GPIO connector on ADQ7–PXIe and ADQ7–PCIe versions.

7 ACQUISITION CONTROL

The acquisition control consists of two partly independent parts;

- acquisition of data in a record
- transfer to host PC.

Figure 30 shows the flow of data through acquisition and data transfer.



#	DESCRIPTION	REF
a	The A/D converter digitizes the analog signal and generate a flow of data.	
b	The acquisition engine manages the data acquisition and builds records of the data. Select acquisition mode: Continuous streaming Triggered streaming Multi-record	7.3
c	The transfer of data to the host PC delivers data in buffers for the user. Select mode of data transfer: Streaming (Real-time driven) Scheduled (Multi-record)	7.4
d	User's application reads data and headers for further processing and/or storage.	7.1, 7.5,

Figure 30: Acquisition control and data transfer.

7.1 Multi-thread notice

Note that the digitizer does not support multi-threaded applications. In the triggered streaming mode, however, a multi-threading programming style has advantages. In such an application, make sure that only one thread communicates with the digitizer at a time.

In the example in Section 7.4.1, one thread handles the control of the digitizer. The other thread only processes data.

Example code available with the digitizer is sometimes written with several threads. Study these examples carefully to see how multi-threading can be used.

7.2 Acquisition memory

The acquisition memory, Figure 31, is of size 4 GBytes.

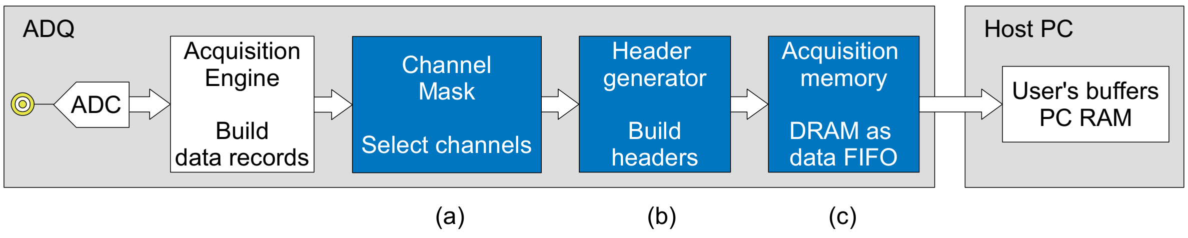
The memory is shared by all activated channels which means that if only one channel is activated, the entire memory is available for that channel.

The data memory is also shared between data and headers, Table 4. A header contains information about the data record, for example, timestamp and channel number. The headers can be setup in two different modes:

- In the normal mode, the headers are activated. This mode is recommended for all standard acquisition modes.
- In the raw mode, there are no headers and all available memory is used for data. The raw mode is only recommended for custom firmware built in the ADQ Development Kit.

The memory is organized as a FIFO where it is possible to record data and read out data simultaneously. This is called readout-while-recording and is available both in multi-record and streaming mode. If the readout process is as fast as the acquisition process, the memory size does not impose any limitations.

If readout is scheduled after recording, the total data set from the measurement has to be limited to 4 GBytes to fit in the acquisition memory.



#	DESCRIPTION	USER COMMAND	REF
a	The channel mask selects which channels to record and which to discard.	*SetStreamConfig* *MultiRecordSetChannelMask*	
b	The header generator apply header information to each record. Select to turn headers on or off.	*SetStreamConfig*	
c	The acquisition memory is a DRAM on the digitizer which operate as a FIFO.		

Figure 31: Acquisition memory.

ACQUISITION MODE	RECORD SIZE	–2A –4A –2C –4C	–1X –2X
ACQUISITION MODE	RECORD SIZE	ADQ7WB 5 GSPS	
Header	< 256 samples	> 5.9 %	
Header	> 256 samples	< 5.9 %	
Header	Asymptotic lower limit	3.1 %	
Raw mode no headers	Any	0 %	

Table 4: Header memory requirement for triggered streaming acquisition mode.

7.3 Acquisition modes

There are three main modes of data acquisition:

- Continuous streaming, see **Section 7.3.1**.
- Triggered streaming, see **Section 7.3.2**.
- Multi-record, see **Section 7.3.3**.

7.3.1 Continuous streaming acquisition

Continuous streaming means a constant flow of data from the ADQ to the host PC. At a trigger event or start command (software trigger), the recording and data transfer starts, **Figure 32**. This continues until stopped by the user.

Note that the data rate from the ADCs is up to 20 GBytes/s but the data rate to the PC is limited to 6.8 GBytes/s[1]. The continuous streaming mode thus assume some type of data reduction in the ADQ to reduce the data rate to a level that the PC can handle. Some typical methods for data rate reduction are listed below:

1. Theoretical limit. Dependent on host PC capacity and PCIe configuration.

- Use sample skip to reduce the data rate.
- Use decimation to reduce data rate. Decimation is sample skip combined with digital filtering to reduce noise and potential aliasing. This method requires the option –FW4DDC.
- Use custom real-time implementation of the firmware in the FGPA. This requires the ADQ Development Kit.

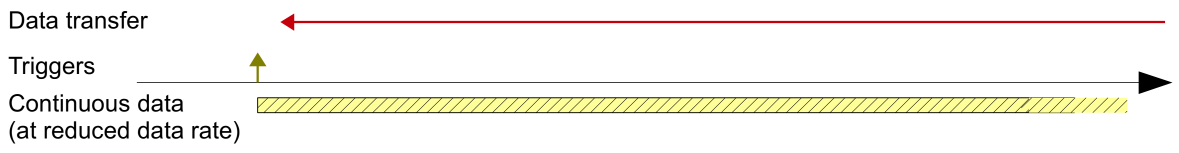


Figure 32: Continuous streaming timing.

Use continuous streaming when there is a real-time infinite event going on, for example, a radio transmission. Another application is when searching for a rarely occurring event and constantly analyzing data in real time in the host PC for finding that event.

7.3.2 Triggered streaming acquisition

The triggered streaming is similar to the continuous streaming except that the data is cut into records. Each record is recorded at a trigger event and has a limited distribution in time. Each record has a header with timing information. The effective data rate to the PC is set by the trigger rate in combination with the length of the record.

The acquisition memory has an important role for triggered streaming; handle bursts of triggers. Readout of a record can start as soon as the recording has started.

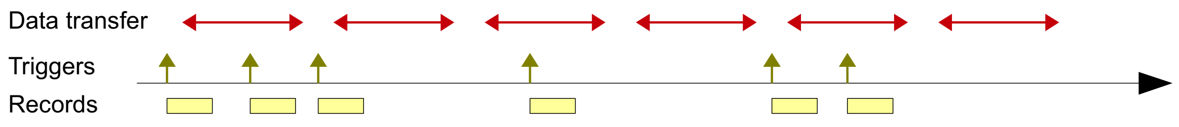


Figure 33: Triggered streaming timing.

Use triggered streaming when a short re-arm time is critical. Or when the acquisition is driven by real-time external events.

7.3.3 Multi-record acquisition

In the multi-record mode, a record is recorded into the acquisition memory and read out on request from the user. The multi-record mode is straight forward to implement and is easy to use if the data set is smaller than the on-board memory.

Readout only can start when the complete record is captured and is always initiated from the user's application.

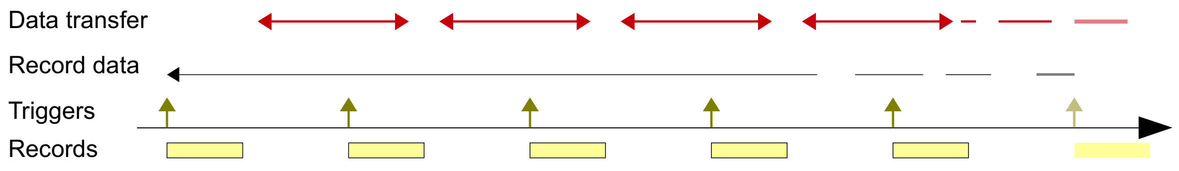


Figure 34: Multi-record timing.

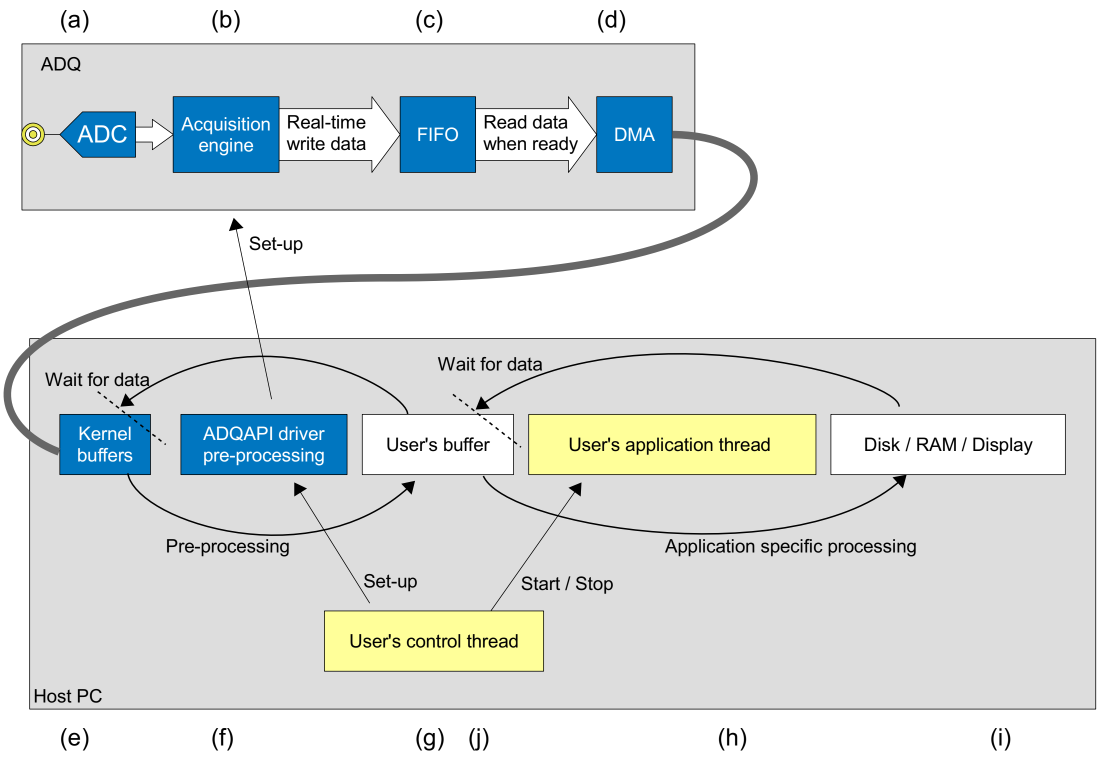
7.4 Data transfer modes

The data transfer can be set up in two main ways:

- Event-driven streaming, see **Section 7.4.1**.
- User-scheduled multi-record, see **Section 7.4.2**.

7.4.1 Streaming data transfer mode

The streaming mode is preferred for high-performance event-driven data transfer. This is the main mode for both continuous streaming (**Section 7.3.1**) and triggered streaming (**Section 7.3.2**) acquisition. The flow is driven by the real-time acquisition in the digitizer. The software is set up to be prepared to receive and process data when it arrives. Since the PC is not a real-time system, the data buffer (**Section 7.2**) on the ADQ is necessary. Each processing step in the software has to complete on time. Otherwise there is a risk of overflow in the FIFO on the digitizer.



#	DESCRIPTION
a	The A/D converter delivers a stream of data into the acquisition engine.
b	The acquisition engine applies triggers, headers, etc.
c	Data is sent in real-time to the FIFO on the ADQ.
d	The DMA is set up to transfer data to the PC.
e	Kernel buffers in the host PC receives the data form the digitizer.
f	The ADQAPI is set up to wait for incoming data and is ready to process it directly. The ADQAPI receives the data and do necessary pre-processing, for example, lost packages, headers, and sends data to the user's buffers.
g	User's buffers in RAM. These are accessed via API commands.
h	The user's application thread is set up to wait for data and process the data as soon as it is available. This is application specific code.
i	Example of output devices.
j	User's control thread set up the system and starts and stops the user's application thread. Note that only this thread communicates with the digitizer.

Figure 35: Block diagram of triggered streaming acquisition and streaming data transfer.

Block diagram and timing of a streaming data transfer is in **Figure 35** and **Figure 36**. First, the entire path for data processing, DMA, API and user's application thread is set up to receive the data. Then the acquisition is set up and armed by the user's control thread.

In case of a burst trigger scenario, several records may be recored into the FIFO while the previous records are transferred to the host PC. The FIFO will handle the load as long as the average data rate is maintained within limits.

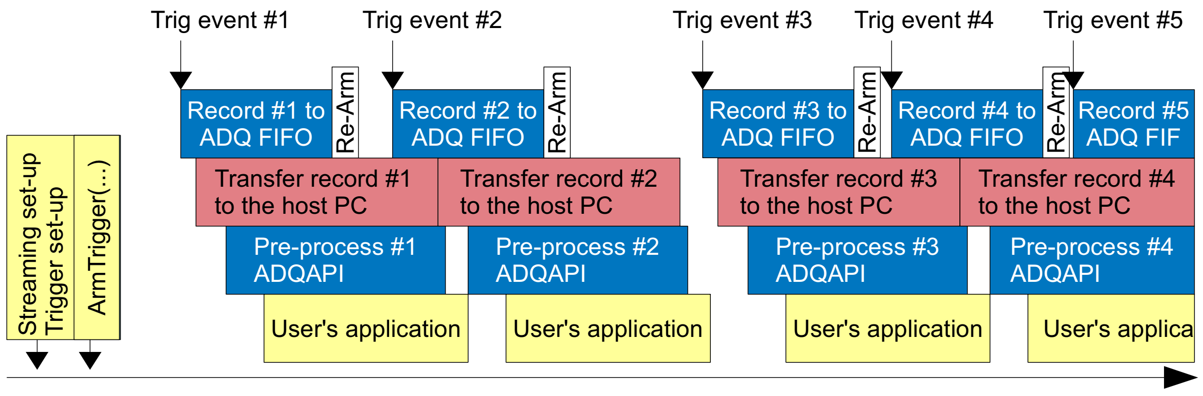


Figure 36: Timing of triggered streaming acquisition and streaming data transfer.

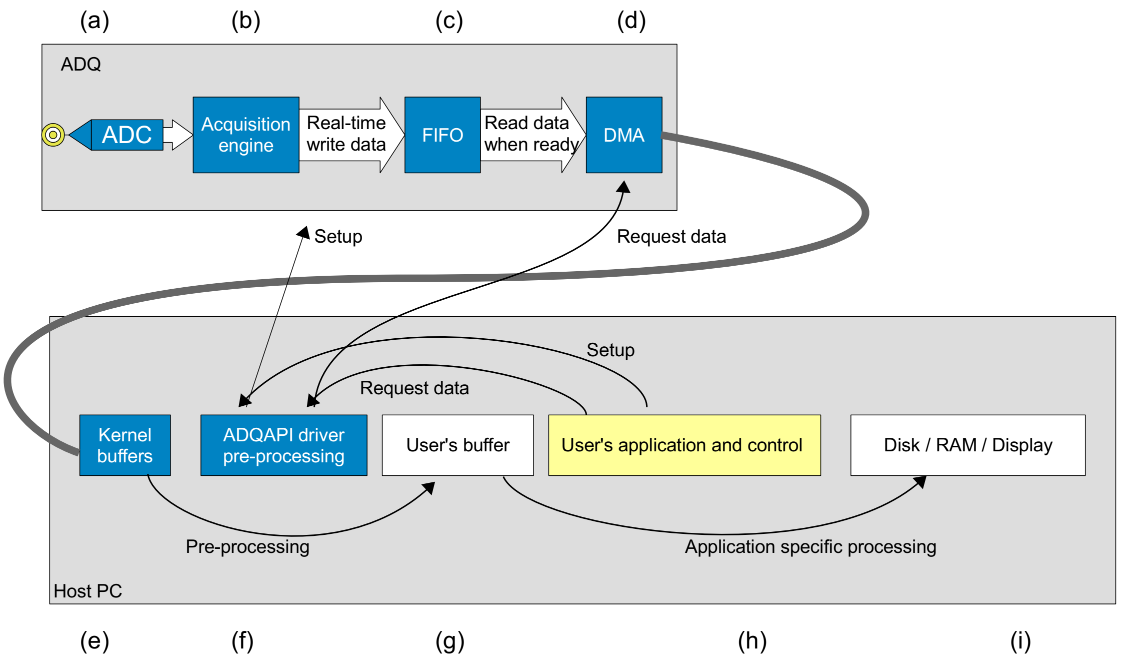
Note that the boxes for each stage in **Figure 36** are drawn a little shorter that the above stage. This is to illustrate that the task must finish in time to be ready to receive the next record. The timing in the diagram is set by the incoming trigger events.

7.4.2 User-scheduled data transfer mode

In the user-scheduled mode the data flow is controlled by the user in a schedule. The user first sets up the digitizer, then starts the acquisition, and after that requests the data. This is the straight forward method if the total amount of data is less than the data buffer size (**Section 7.2**), that is, it can be stored in the buffer of the digitizer.

Note that if using readout while recording, the multi-record is not automatically limited by the data buffer size.

A block diagram of triggered streaming acquisition and user-scheduled data transfer is given in **Figure 37**.



#	DESCRIPTION
a	A/D Converter delivers a stream of data int the acquisition engine.
b	The acquisition engine applies triggers, headers, etc.
c	Data is sent in real-time to the FIFO on the ADQ.
d	The DMA transfers data to the PC when requested by the user.
e	Kernel buffers in the host PC receives the data form the digitizer.
f	The ADQAPI receives incoming data and do necessary pre-processing, for example lost packages, and sends data to the user’s buffers.
g	User’s buffers in RAM. These are accessed via API commands.
h	The user’s application sets up the digitizer for acquiring data. Then the software requests the acquired data and perform application specific processing. This is the user’s software.
i	Examples of output devices.

Figure 37: Block diagram of user-scheduled data transfer.

The timing of a user-scheduled multi-record transfer is in **Figure 38**. The acquisition is set up and armed. The user’s software checks for available records. The available records can be transfered with the ***GetDataWHTS*** command[1].

1. WHTS stand for ‘with header and timestamp’.

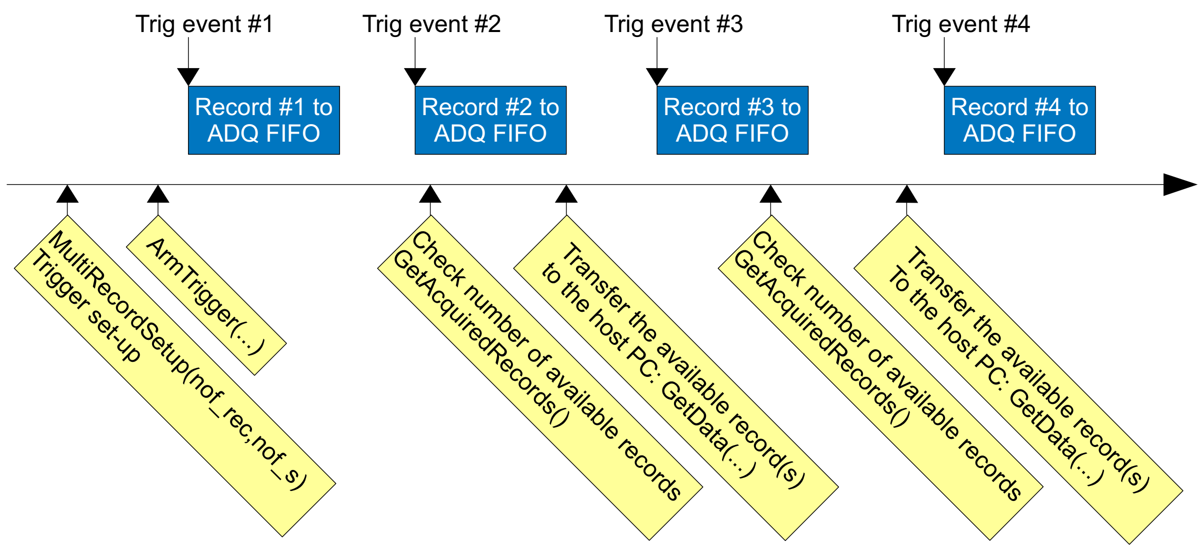

Trig event #1
Trig event #2
Trig event #3
Trig event #4
Record #1 to ADQ FIFO
Record #2 to ADQ FIFO
Record #3 to ADQ FIFO
Record #4 to ADQ FIFO
MultiRecordSetup(nof_rec,nof_s) Trigger set-up
ArmTrigger(...)
Check number of available records GetAcquiredRecords()
Transfer the available record(s) to the host PC: GetData(...)
Check number of available records GetAcquiredRecords()
Transfer the available record(s) To the host PC: GetData(...)

Figure 38: Timing of user-scheduled data transfer.

7.4.3 Transfer buffers

The data transfer buffers are the kernel buffers in **Figure 35** and **Figure 37**. These buffers are used by the DMA and the ADQAPI to transfer the data from the ADQ digitizer to the host PC.

The transfer buffers are owned and managed by the ADQAPI, but the user sets the size and number of buffers with the command ***SetTransferBuffers***.

7.4.4 User's buffers

The user's buffers in **Figure 35** and **Figure 37** are allocated and managed by the user. The ADQAPI recreates the data record and puts the result in these buffers.

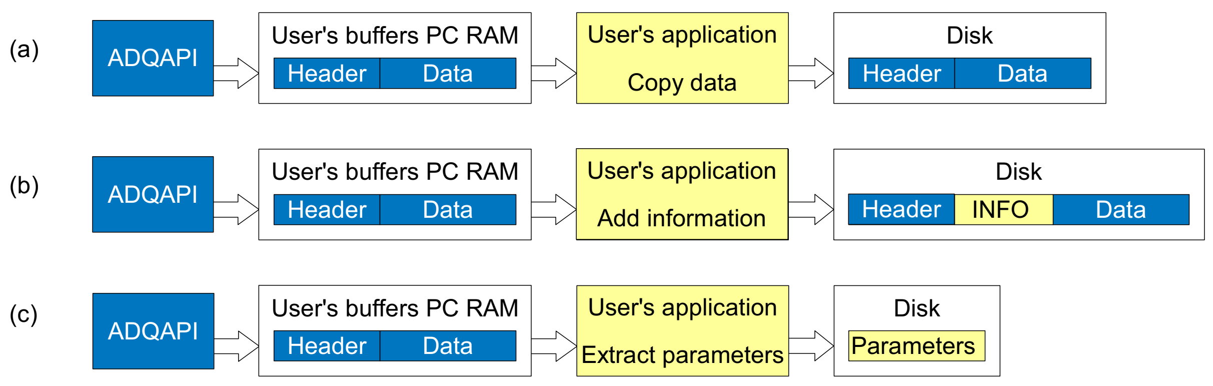
Create the buffers using ***malloc***. Then provide the pointers of these buffers to the ADQAPI through the commands ***GetDataStreaming*** (for triggered streaming **Section 7.4.1**) or ***GetDataWHTS*** (for multi-record **Section 7.4.2**).

The user's buffers consist of two sets of buffers; one for header information and one for data. The header is always 40 bytes per record and the content is described in **Section 7.6**. The data buffer size is depending on the amount of data in each record. For FWDAQ, the record size is always constant and the buffer size can be set to match the record size. The example code in ADQAPI_example[1] illustrate how to handle data buffers in general.

7.5 Users application software consuming data

The users application can consume the data in different ways. Some examples are in **Figure 39**. The component "disk" is used for illustrating the end point for the data. It may also be a display or some other device such as an alarm.

1. The ADQAPI_example is found in the folder installation of the digitizer software.



#	DESCRIPTION	REF
a	Copy the data to disk for offline analysis. Header is in default format.	7.6.1
b	Copy the data to disk for offline analysis. Additional information about e.g. the experiment, is added to the header.	
c	Header and data is analyzed in real-time and only requested parameters are stored.	7.6

Figure 39: Data flow through the system

7.6 Record header

7.6.1 Metadata

The record header contains the information described in **Table 5**. Example code is available to pack this information to a header and write to disk. The example C-code defines a struct which reads the header information from the header buffer.

PARAMETER	FORMAT	DESCRIPTION	REF
Record Status	Byte	Over/under range, FIFO fill factor and lost data.	7.6.2
User ID	Byte	A user-configurable value to identify the ADQ unit.	7.6.3
Channel	Byte	The channel number.	7.6.5
Data format	Byte	Information about data.	7.6.5
Serial number	uint32	Serial number of the ADQ digitizer.	7.6.4
Record number	uint32	The current record number.	7.6.6
SAMPLE_PERIOD	int32	Time between two samples in steps of 25 ps.	4.3
TIME_STAMP	uint64	Timestamp of trigger event in steps 25 ps.	4.3
RECORD_START	int64	Time between trigger event and record start in steps 25 ps.	4.3
Record length	uint32	Length in number of samples of data in record.	7.6.8
General purpose vector	uint16	Usage varies with option.	
Timestamp reset	uint16	Number of times that the timestamp was reset.	4.3.2

Table 5: Header data.

7.6.2 Record Status

The status parameters indicates if the record was transferred correctly, **Table 6**.

Over-range or under-range condition within the data is indicated. Over-range and under-range can appear at various stages in the signal chain and the result is not easy to predict. A more detailed description of over-range and under-range is in **Section 7.7**.

FIFO fill factor is indicated. This is useful when tuning a data-driven process to avoid FIFO overflow and still get maximum efficiency from the experiment. For very long records, the maximum fill factor during the record is given.

If there is an overflow in the FIFO on the digitizer, data will be lost. The ***LOST_DATA*** bits inform about this.

OVER RANGE	FIFO FILL			LOST DATA				DESCRIPTION	COMMENT
7	6	5	4	3	2	1	0		
0								No over-range detected	
1								Over/under-range detected	Anywhere in the data.
	X	X	X					FIFO fill factor	Steps of 12.5%. 111 means > 87.5% and 000 is below 12.5%.
				0	0	0	0	No lost data	
							1	Lost record	The software has generated a header with no data to indicate a lost record.
						1		Lost data in beginning of the record	The start of the record is missing. Timestamp information is also missing.
					1			Lost data in the middle of the record	One or many section(s) anywhere inside the data is missing.
				1				Lost data in the end of the record	

Table 6: Status.

7.6.3 User ID

User ID is a custom byte that can be set by the user. Set this parameter by the ***EnableUseOfUserHeaders*** command.

7.6.4 Serial number

The serial number is the serial number of the ADQ hardware. The serial number is printed on a label on the digitizer in the form “S/N SPD-04004”. The serial number field is the number part, that is 04004.

7.6.5 Channel

This is an indication of from which channel the record originates. The first numbers of the channel parameter are reserved for the physical channels. The remaining combinations are available for ADQ Development Kit users to create artificial channels.

7.6.6 Record number

The Record number is counting the number of records captured from the power-up of the digitizer. The record number will wrap at 2^{32}-1 and start at 0. The user has to keep track of this wrap if necessary.

7.6.7 Data format

The header data parameter ‘Data format’ (**Table 5**) informs the user on how to interpret data. Allowed values are given in **Table 7**.

VALUE	DESCRIPTION
0	16 bits data word in two’s complement
1	32 bits data word in two’s complement

Table 7: Data formats

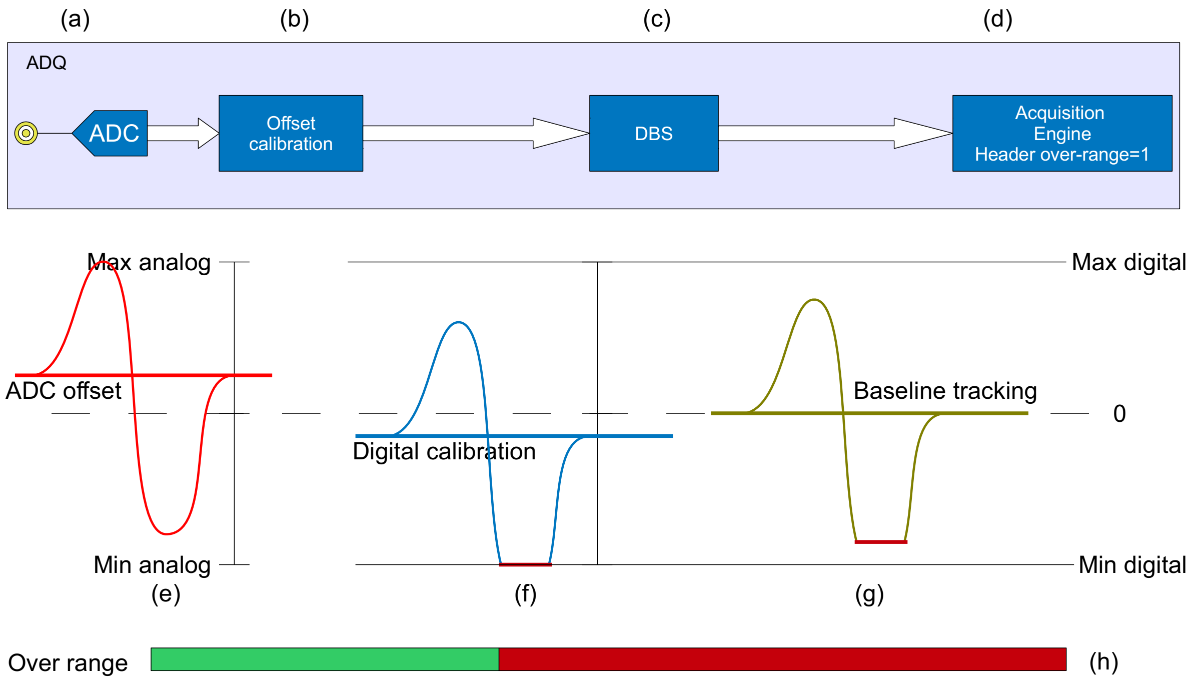
7.6.8 Record length

This is the length of the data record in number of samples. Use the field Data Format to calculate the length in bytes.

7.7 Over-range and under-range

The over/under-range bit in the header indicates that over-range or under-range occurred at one or several samples within the record and at any stage in the signal chain. The result of the over-range is not straightforward to predict, see **Example 6**.

Example 6: ***Figure 40*** *shows under-range in ADC before the Offset calibration. The result is that the digital code where the under-range occurred is not the maximum code.*



#	DESCRIPTION
a	ADC with offset.
b	Digital offset calibration.
c	Digital baseline stabilizer that dynamically adjust baseline to zero. That is adjust the offset. Under-range has already occurred prior to this stage.
d	Acquisition create a header which contain over/under-range indication.
e	The analog input signal with offset. The ADC also contribute to the offset.
f	Digital calibration cannot account for offset on the analog signal. Under-range occur when signal is out of range on the negative side.
g	The DBS fine-tunes the offset, that is add a value to the signal. Then the under-range is not at the minimum digital code anymore.
h	Overflow indication is set when over/under-range occur and is kept for the record.

Figure 40: Under-range in the ADC calibration.

8 HOST PC CONNECTION

The host PC connection is either USB or PCI Express. From the programmers' perspective, there is no difference which interface to use. It only matters when the data rate to the host PC is critical.

A general software that takes the interface into account can use the ***IsUSBDevice***, ***IsUSB3Device*** and ***IsPCIeDevice*** commands to check the present connection.

8.1 USB interface

The ADQ may be connected to a USB2.0 or USB3.0 port in the host PC. The USB connection is internal in the –PCIE and –PXIE versions of the digitizer. It is intended for firmware upgrade is the PCIe firmware upgrade fails. It is also possible to use this interface for debugging if the PCIe link is not responding properly.

8.2 PCI Express interface

The –PCIE, and –PXIE versions of the digitizer use PCI Express electrical interface to communicate with the host PC. Generation 1, 2, and 3 up to 8 lanes is supported by the digitizer.

8.3 Using several units

8.3.1 Using several digitizers from a single application.

Several digitizers can be connected to the same PC.

Each unit is then available and can be accessed individually.

Each software command contains a pointer to the control unit for all ADQ digitizers and an instance number that points out the current ADQ.

To identify a specific unit, read the serial number ***GetBoardSerialNumber***. This gives a mapping between the instance number and a physical unit.

The record header (**Table 5**) contains a byte field (User ID) where the user can set an identifier for each card. This gives a mapping between physical unit and data.

8.3.2 Using several digitizers from a several applications.

When several separate applications or threads are used for talking to different digitizers the commands ***ListDevices*** and ***OpenDevice*** has to be used. ***FindDevices*** will not work since ***FindDevices*** locks all available digitizers to the same application.

9 REFERENCES

[1] 19-2226 ADQ7WB Datasheet

[2] 14-1351 ADQAPI Reference guide

[3] 08-0214 ADQAPI User guide

[4] 18-2059 ADQ7 updater user guide

[5] 16-1790 Application note: High performance digitizer theory

[6] 17-2010 ADQ7 Development Kit user guide

Important Information

Worldwide Sales and Technical Support

www.spdevices.com

Teledyne SP Devices Corporate Headquarters

Teknikringen 6
SE-583 30 Linköping
Sweden

Phone: +46 (0)13 465 0600
Fax: +46 (0)13 991 3044
Email: info@spdevices.com