This manual describes how to get the full potential out of Teledyne SP Devices’ digitizer ADQ7DC. 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
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 ADQ7DC Architecture

The ADQ7DC architecture is shown in Figure 1. References to the corresponding sections with further information are also included.

![ADQ7DC Architecture Diagram](image)

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
<th>REFERENCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Switch between 1 channel and 2 channels</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>Signal conditioning analog front-end.</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>High speed and high resolution A/D converter. The A/D converter operate interleaved at 5 GSPS per channel. The A/D converters can be further interleaved to get 10 GSPS.</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>Digital calibration of gain and offset.</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>Teledyne SP Devices’ proprietary technologies for signal quality enhancement; ADX for SFDR in radio systems and DBS for baseline stability in pulse data systems.</td>
<td>3</td>
</tr>
<tr>
<td>f</td>
<td>Acquisition engine that handles triggers and controls the data flow.</td>
<td>4, 7</td>
</tr>
<tr>
<td>g</td>
<td>Data FIFO to buffer data before transmission to the host PC.</td>
<td>7</td>
</tr>
<tr>
<td>h</td>
<td>The data transfer to the host PC is through a PCIe or a USB3.0 link.</td>
<td>7</td>
</tr>
<tr>
<td>i</td>
<td>Flexible clock generator</td>
<td>5</td>
</tr>
<tr>
<td>j</td>
<td>General Purpose digital Input and Output control</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 1: ADQ7 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 ADQ7DC has 14 bits resolution, while the data format inside the ADQ7DC and out to the host PC is 16 bits. The 14 bits from the ADCs are MSB aligned in this 16 bit data word. Thus initially the 2 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 2 LSBs may not be zero in the data output from the ADQ7DC. Calibration and other computations in the FPGA may result in fractional result. This is not rounded to 14 bits in order to avoid adding computational noise.

Example 1: A 14 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 2.3. 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.

<table>
<thead>
<tr>
<th>ADC RAW CODES1</th>
<th>GAIN CORRECTION</th>
<th>DIGITAL CODE LEVEL2</th>
<th>ACTUAL ANALOG RANGE</th>
<th>ANALOG LEVEL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>1063 / 1024</td>
<td>0x0000</td>
<td>1 Vpp</td>
<td>0.0 mV</td>
</tr>
<tr>
<td>0x0004</td>
<td>1063 / 1024</td>
<td>0x0004</td>
<td>1 Vpp</td>
<td>0.061 mV</td>
</tr>
<tr>
<td>0x0008</td>
<td>1063 / 1024</td>
<td>0x0008</td>
<td>1 Vpp</td>
<td>0.122 mV</td>
</tr>
<tr>
<td>0x000B</td>
<td>1063 / 1024</td>
<td>0x000B</td>
<td>1 Vpp</td>
<td>0.168 mV</td>
</tr>
<tr>
<td>0x0010</td>
<td>1063 / 1024</td>
<td>0x0011</td>
<td>1 Vpp</td>
<td>0.259 mV</td>
</tr>
<tr>
<td>0x0014</td>
<td>1063 / 1024</td>
<td>0x0015</td>
<td>1 Vpp</td>
<td>0.320 mV</td>
</tr>
<tr>
<td>0x0018</td>
<td>1063 / 1024</td>
<td>0x0019</td>
<td>1 Vpp</td>
<td>0.381 mV</td>
</tr>
<tr>
<td>0x001B</td>
<td>1063 / 1024</td>
<td>0x001C</td>
<td>1 Vpp</td>
<td>0.427 mV</td>
</tr>
</tbody>
</table>

1. This is the raw codes from the ADC. It is 14 bits MSB aligned in 16 bit word. The 2 LSBs are thus 0.
2. This is the result from the gain compensation. The 2 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.
1.2.4 ADQ7DC sampling clock frequency

ADQ7DC is designed for the specified clock frequencies only. A different sampling rate can be achieved by using the sample skip function, Section 5.9.

*The ADQ7DC is available in two different speed modes, 5 GSPS dual channel and 10 GSPS single channel. Both modes are supported by the hardware and included with the ADQ7DC. The firmware of the FPGA has to be changed when switching mode. This may require a re-start of parts of the system. Use the ADQ7 Updater tool to switch firmware.*

Some parts of the descriptions and examples are done for only one of these frequencies. The examples are translated to the other sample rates by multiplying or dividing with the corresponding sample rate ratio.

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 different host PC connections (USB, PCIe etc) has their own respective 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 1 Vpp and is by default symmetrical around zero. The range can be adjusted from [–0 mV +1000 mV] to [–1000 mV +0 mV] by the DC-offset feature, Section 2.2.
The maximum digital code $2^{15}$ represents an analog signal with a level $\text{ACTUAL\_ANALOG\_RANGE} / 2$ at the input. A specific analog signal $\text{ANALOG\_LEVEL}$ will then be represented by the following digital code:

$$\text{DIGITAL\_CODE\_LEVEL} = \frac{\text{ANALOG\_LEVEL}}{\text{ACTUAL\_ANALOG\_RANGE} / 2} \times 2^{15} \quad (1)$$

A specific code $\text{DIGITAL\_CODE\_LEVEL}$ then represent the analog level as:

$$\text{ANALOG\_LEVEL} = \left( \frac{\text{DIGITAL\_CODE\_LEVEL}}{2^{15}} \right) \times \left( \frac{\text{ACTUAL\_ANALOG\_RANGE}}{2} \right) \quad (2)$$
## 2 SETTING UP THE ANALOG FRONT-END

### 2.1 AFE block diagram

The analog front-end setup for ADQ7DC is found in Figure 2.

![Diagram of ADQ7 AFE control](image)

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
<th>USER COMMAND</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>The analog input is terminated 50 ohms to GND.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>Switching between 1 and 2 channels. Load the correct firmware using ADQ7 updater tool. This will set the input switch. The input switch can also be controlled by a software command for disconnecting the input. Note that this command do not change the ADQ7 from 1 to 2 channels acquisition.</td>
<td>SetAfeSwitch</td>
<td>2.1</td>
</tr>
<tr>
<td>c</td>
<td>Set a DC-offset for better using the signal range when the signal is unipolar. Changing the DC-offset requires a settling time of 1 s.</td>
<td>SetAdjustableBias</td>
<td>2.2</td>
</tr>
<tr>
<td>d</td>
<td>The gain and offset are calibrated at DC. 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.</td>
<td>SetGainAndOffset</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Figure 2: ADQ7 AFE control.

The ADQ7DC can operate with two channels at 5 GSPS or one channel at 10 GSPS. To change mode of operation, the single or dual channel firmware has to be activated by using the ADQ7 Updater tool\(^1\). For more information, see [6]. Note that there are different connectors for the single and dual channels mode. This means that one detector for 10 GSPS can be connected to the channel X at the same time as two slower detectors are connected to channel A and B. The switch between the 1 analog input channel and 2 analog input channels modes is set in the correct state when loading the firmware. The switch may also be controlled from software\(^2\). The front-end switch can be used for protecting the input. By switching to the un-used input, the switch can disconnect the detector during certain risky operations like power up/down of detectors.

### 2.2 Set analog DC-offset

A user-controlled DC-offset is available. The analog DC-offset is applied to the signal to better adopt to the signal range of the digitizer. The analog range is by default set symmetrical around zero. If the signal is unipolar or heavily unsymmetrical, the DC-offset function can adjust the signal to an optimal verti-

---

1. Note that there is one ADQ7 updater tool for ADQ7 and another ADQ updater for other ADQ digitizers.
2. Note that the software switch only changes the input. To change function, the firmware has to be loaded.
cal position for the A/D converter. In this way, the full 14 bits can be used for representing the unipolar pulse. The DC-offset is set with the command SetAdjustableBias.

The DC-offset is set in digital codes $\text{DC_OFFSET_CODE}$ in the range $[-2^{15}: 2^{15}-1]$, which correspond to an analog signal level in the range:

$$[-\text{ACTUAL_ANALOG_RANGE} / 2: \text{ACTUAL_ANALOG_RANGE} / 2].$$  \hfill (3)

To determine the parameter of SetAdjustableBias to get a DC-offset at the voltage level $\text{DC_OFFSET_ANALOG}$, use:

$$\text{DC_OFFSET_CODE} = \text{round} \left( \frac{\text{DC_OFFSET_ANALOG}}{\text{ACTUAL_ANALOG_RANGE} / 2} \times 2^{15} \right)$$  \hfill (4)

Since the digitizer has higher resolution than the intrinsic accuracy of the DC-offset generator, the actual digital codes read out from the ADQ may differ from the expected level. For accurate baseline measurements, the Digital Baseline Stabilizer (DBS) offers a digital correction of the baseline to an accuracy of 22 bits, Section 3.1.

2.3 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 $\text{CAL_GAIN}$ and $\text{CAL_OFFSET}$. The raw data from the A/D converter, $\text{ADC_RAW_CODE}$, is corrected with the calibrated values according to:

$$\text{DIGITAL_OUTPUT_CODE} = \text{ADC_RAW_CODE} \times \text{CAL_GAIN} – \text{CAL_OFFSET}$$  \hfill (5)

The user can override these settings by using the software command SetGainAndOffset. The parameter $\text{USER_GAIN}$ and $\text{USER_OFFSET}$ 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

$$\text{DIGITAL_OUTPUT_CODE} = \text{ADC_RAW_CODE} \times \text{CAL_GAIN} \times \text{USER_GAIN} – \text{CAL_OFFSET} – \text{USER_OFFSET}.$$  \hfill (6)

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

$$\text{DIGITAL_OUTPUT_CODE} = \text{ADC_RAW_CODE} \times \text{USER_GAIN} – \text{USER_OFFSET}.$$  \hfill (7)

To get the raw code, $\text{ADC_RAW_CODE}$, use SetGainAndOffset(128+CHANNEL,1024,0).
3 SIGNAL QUALITY ENHANCEMENT

3.1 Digital Baseline Stabilizer

The Digital Baseline Stabilizer, DBS, is designed for pulse data measurement where high accuracy relative to a known baseline is required. The key features of DBS are:

- Tracks and compensates for baseline variations from, for example, temperature and aging.
- Suppresses pattern noise \(^1\) to 22 bits precision.
- Automatically locks the baseline to a user defined-value.

Note that DBS is off at power up. DBS has to be activated by the user’s application software. The time when DBS is activated is important. To get a good initial estimate, DBS should be activated when there is very little signal energy present at the input. If there is too much signal power in the initial estimate, the convergence of DBS is slowed down.

Note that DBS is defined for systems with a baseline and distinct short pulses. DBS is not intended for sinusoidal type of signals. For sinusoidal types of signals use ADX, Section 3.2.

3.2 Interleaving correction ADX

The Interleaving correction ADX is available on the 5 GS/s, which is internally interleaved, and on the interleaved single channel 10 GS/s version of ADQ7.

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 DBS has to be switched off if ADX is used.

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

---

1. Pattern noise is systematic errors that may arise from the actual design of the ADC IC or the board design.
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 3.

# DESCRIPTION USER COMMAND REF
a Connectors for external analog input signals. The number of channel vary on the different models and configurations. SetupLevelTrigger 4.9
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). SetInternalTriggerPeriod 4.10
d Internal trigger generator. SWTrig 4.6
e A software trigger is available for user control. f External trigger input from backplane in PXIe or MTCA. 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. j Activate trigger output. SetTriggerMode 4.11
k Select which channels to record data from. SetStreamConfig 4.11
l Acquisition engine creates a record from streaming data 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 3: Trigger source selection and setup illustrated for a 2 channels mode.
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 4. 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 5. 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.

![Pretrigger timing diagram](image1)

![Trigger delay timing diagram](image2)

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 3 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 digitizer.)

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 3: Assume an ADQ7DC 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

\[
\text{TRIGGER\_TIME} = \text{TIME\_STAMP} \times \text{TIME\_BASE} = 125125\ \text{ps} = 125.125\ \text{ns}
\]

The time for the first sample in the record is

\[
\text{RECORD\_TIME} = (\text{TIME\_STAMP} + \text{RECORD\_START}) \times \text{TIME\_BASE} = 109.000\ \text{ps} = 109\ \text{ns}
\]

The time between two samples are

\[
\text{SAMPLE\_TIME} = \text{SAMPLE\_PERIOD} \times \text{TIME\_BASE} = 200\ \text{ps}
\]

The time from the record start to the trigger is

\[
\text{RECORD\_START} \times \text{TIME\_UNIT} = –16.125\ \text{ns}
\]

The number of samples between the record start and the trigger event is

\[
\left(\frac{\text{TRIGGER\_TIME} – \text{RECORD\_TIME}}{\text{SAMPLE\_TIME}}\right) / \text{SAMPLE\_PERIOD} = \left(\frac{\text{RECORD\_START}}{\text{SAMPLE\_PERIOD}}\right) = 80.625\ \text{samples}
\]

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 method 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 
\[ \text{TIME\_STAMP} = \text{TIME\_STAMP\_OF\_RECORD} - \text{TIME\_STAMP\_REFERENCE}. \]

3. The third method is to apply an external trigger to reset the timestamp, Figure 6. 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 6. 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 and -MTCA formats work in the same way as the sync signal.

---

**Figure 6: Timestamp reset from external trigger.**

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
<th>USER COMMAND</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>External trigger input signal on front panel connector.</td>
<td></td>
<td>4.7</td>
</tr>
<tr>
<td>b</td>
<td>External sync input signal on front panel connector.</td>
<td></td>
<td>4.7</td>
</tr>
<tr>
<td>c</td>
<td>Other available sources (see SetTriggerMode for a list)</td>
<td></td>
<td>4.7</td>
</tr>
<tr>
<td>d</td>
<td>Select source for resetting timestamp.</td>
<td>SetupTimestampSync DisarmTimeStampSync ArmTimeStampSync</td>
<td>4.3.2</td>
</tr>
<tr>
<td>e</td>
<td>Timestamp counter value is reset at power-up of the digitizer.</td>
<td></td>
<td>4.3.2</td>
</tr>
<tr>
<td>f</td>
<td>Reset the timestamp counter on each pulse of the selected source.</td>
<td></td>
<td>4.3.2</td>
</tr>
<tr>
<td>g</td>
<td>Timestamp is then measuring time relative the previous reset signal.</td>
<td></td>
<td>4.3.2</td>
</tr>
<tr>
<td>h</td>
<td>Reset the timestamp counter only on the first pulse of the selected signal. The external signal is then a systems synchronization signal.</td>
<td></td>
<td>4.3.2</td>
</tr>
<tr>
<td>i</td>
<td>The number of times the time stamp has been reset can be read from a register.</td>
<td>GetTriggerBlockingGate Count</td>
<td>4.3.2</td>
</tr>
</tbody>
</table>
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 7. 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 framed by the blocking event. This creates groups of triggers that belong together. The modes of operation for trigger blocking are shown in Figure 7 (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 7 (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 7 (a, b, c).

Note the order of the commands for activating triggers and trigger blockers, Figure 7 (e, g, h, i). Figure 8 illustrates how the triggers are accepted or rejected in the window mode.

![Trigger Blocking Diagram](image.png)

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

Figure 7: Blocking and gating of triggers.
4.4.2 Block triggers once

The mode for blocking triggers once is illustrated in Figure 7 (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 13.

4.4.3 Windowing triggers

The window mode for blocking triggers is illustrated in Figure 7 (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 7 (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 9.

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 $\frac{\text{TRIGGER_CLOCK_PERIOD}}{\sqrt{12}}$. The highest resolution is achieved with an external trigger connected to the TRIG connector. ADQ7DC has a trigger clock at 20 GSPS, $\text{TRIGGER_CLOCK_PERIOD}$ 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.

![Figure 9: Sources of jitter on the trigger signal.](Image)

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 $\text{TIME_STAMP}$ 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 10.

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 10: Extended trigger resolution timing for ADQ7 at 5 GSPS.](image)

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 10, 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.

<table>
<thead>
<tr>
<th>CONNECTOR</th>
<th>DESCRIPTION</th>
<th>TIME RESOLUTION</th>
<th>TOTAL JITTER</th>
<th>IMPEDANCE</th>
<th>TRIG LEVEL</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIG</td>
<td>External trigger on front panel.</td>
<td>50 ps</td>
<td>28 ps</td>
<td>50 Ω / 500 Ω</td>
<td>SW contr.</td>
<td>4.7.1</td>
</tr>
<tr>
<td>SYNC</td>
<td>Sync signal on front panel.</td>
<td>3.2 ns</td>
<td>0.9 ns</td>
<td>50 Ω / 500 Ω</td>
<td>SW contr.</td>
<td>4.7.3</td>
</tr>
<tr>
<td>STARB</td>
<td>Backplane trigger in PXIe systems. Requires trigger timing card.</td>
<td>3.2 ns</td>
<td>0.9 ns</td>
<td>PXIe standard</td>
<td>PXIe standard</td>
<td>4.8.1</td>
</tr>
<tr>
<td>MLVDS</td>
<td>Backplane trigger in MVLDS systems.</td>
<td>3.2 ns</td>
<td>0.9 ns</td>
<td>MLVDS.4 standard</td>
<td>MLVDS.4 standard</td>
<td>4.8.2</td>
</tr>
</tbody>
</table>

**Table 3: External trigger inputs.**

#### 4.7.1 External trigger TRIG front panel connector

The block diagram for the TRIG is shown in Figure 11 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.

![Diagram](image)

**Figure 11: External trigger on front panel.**

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

The block diagram for the SYNC input is shown in Figure 12 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.

![Diagram](image)

(a) (b) (c) (d) (e)

Figure 12: 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 13. In Figure 13 (a) an external source is driving the array of ADQ digitizers. In Figure 13 (b) one of the ADQ digitizer is used as a master and is driving signals to the array of ADQ digitizers. 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 [9].
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 ADQ7DC. 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 backplane 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.

Figure 13: Bussed connections
4.8.2 MTCA.4 interface

The MLVDS lines, port 17 to 20, can be used as trigger input or output, Figure 15. The backplane trigger support both input and output triggers. These operations are independent and can be used simultaneously.
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\(^1\), Figure 16. The level trigger is set to trigger on rising or falling edge. Here, rising edge is illustrated.

---

1. Firmware option FWPD offers a channel independent level trigger. See specific documentation in form of user guides [8] and application note [7].
4.9.1 Setting the level trigger level

The level is given in digital codes:

$$LEVEL\_CODE = \frac{ANALOG\_TRIGGER\_LEVEL}{(ANALOG\_BASELINE / 2)} \times 2^{15}$$ (8)

The **ANALOG\_BASELINE** is the analog full scale range. If a DC-offset is used at the input, add the **DC\_OFFSET\_CODE** from (4).

4.9.2 Level trigger and DBS

The Digital Baseline Stabilizer, DBS, compensates for fluctuations in the baseline. The DC level is then set by a parameter **DBS\_CODE** into DBS. The level trigger is set relative to the analog baseline as:

$$LEVEL\_CODE = DBS\_CODE + \frac{ANALOG\_STEP}{(ANALOG\_BASELINE / 2)} \times 2^{15}$$ (10)

4.9.3 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 17. 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.
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.
- MTCA.4 MLVDS triggers in the backplane, Section 4.8.2.

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 three-dimensional 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 18 (a, b) illustrates how trigger events appear at the trigger output. The trigger events at Figure 18 (c) will generate a record. The event Figure 18 (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.

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 19 and externally, Figure 20.

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.
**Internal trigger**

(a)

Select trigger

(b) Internal trigger output

External Equipment

Trigger event

(c)

Select trigger output

(d)

Internal trigger generator

SetupTriggerOutput 4.11

Select internal trigger as output

SetupTriggerOutput 4.11

Select internal trigger as trigger source

SetTriggerMode

Acquisition engine creates a record from streaming data

SetTriggerMode

(e) Acquistion engine creates a record from streaming data

To data FIFO

Figures 19: Internal routing of internal trigger.

**External trigger**

(a)

Select trigger

(b) Internal trigger output

External Equipment

Trigger event

(c)

Select trigger output

(d)

Internal trigger generator

SetupTriggerOutput 4.11

Select internal trigger as output

SetupTriggerOutput 4.11

Select external trigger as trigger source

SetTriggerMode

Acquisition engine creates a record from streaming data

SetTriggerMode

(e) Acquisition engine creates a record from streaming data

To data FIFO

Figures 20: 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 21 is recorded on the channel D. The trigger timing is then calculated with high precision using interpolation.

![Diagram of Distributing level trigger]

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
<th>USER COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Select a channel as level trigger.</td>
<td>SetupLevelTrigger</td>
</tr>
<tr>
<td>b</td>
<td>Select level trigger as trigger out.</td>
<td>SetupTriggerOutput</td>
</tr>
<tr>
<td>c</td>
<td>Internal hardware connection trigger out to trigger in (shared SMA connector).</td>
<td>SetTriggerMode</td>
</tr>
<tr>
<td>d</td>
<td>Select external trigger for triggering</td>
<td>SetTriggerMode</td>
</tr>
<tr>
<td>e</td>
<td>External cable connection to other digitizers.</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>Select external trigger for triggering.</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 21: 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.

Figure 22: Clock system overview

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
<th>TYPICAL FREQUENCIES</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Clock reference.</td>
<td>10 MHz</td>
<td>5.2</td>
</tr>
<tr>
<td>b</td>
<td>Clock generator that generates several frequencies.</td>
<td>5, 10GHz</td>
<td>5.6</td>
</tr>
<tr>
<td>c</td>
<td>Various interfaces.</td>
<td></td>
<td>4.7</td>
</tr>
<tr>
<td>d</td>
<td>The ADC operates on sampling clock.</td>
<td>5, 10GHz</td>
<td>4</td>
</tr>
<tr>
<td>e</td>
<td>The trigger operates on the trigger clock.</td>
<td>20 GHz</td>
<td>4.6</td>
</tr>
<tr>
<td>f</td>
<td>Other interfaces operate on the Data Clock.</td>
<td>312.5 MHz</td>
<td>4.6</td>
</tr>
<tr>
<td>g</td>
<td>The FPGA operate on the Data Clock.</td>
<td>312.5 MHz</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>The host PC operate on an independent clock.</td>
<td>125 MHz (PCIe)</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>The clock generation for the host interface depend on interface type.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2 Flexible clock network

The preferred clock method is a systems design parameter and the 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 refer-
ence 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 ADQ7DC is given in Figure 23.

![Block diagram of the clock network for ADQ7DC](image)

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
<th>USER COMMAND</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>The SMA connector is common for external clock reference input, external clock input and external reference output.</td>
<td>SetClockInputImpedance</td>
<td>5.5</td>
</tr>
<tr>
<td>b</td>
<td>Selecting the usage of the external connector is implicit from setting up the clock system.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>The internal clock reference is a high performance VCTCXO.</td>
<td></td>
<td>5.4</td>
</tr>
<tr>
<td>d</td>
<td>PXIe and MTCA.4 support clock reference from the backplane</td>
<td></td>
<td>5.5</td>
</tr>
<tr>
<td>e</td>
<td>Select which clock reference source to use.</td>
<td>SetClockSource</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>The external clock reference is cleaned from jitter.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>The internal clock generator.</td>
<td></td>
<td>5.6</td>
</tr>
<tr>
<td>h</td>
<td>Select which clock source to use</td>
<td>SetClockSource</td>
<td>5.7</td>
</tr>
<tr>
<td>i</td>
<td>Reduce sample rate with sample skip.</td>
<td>SetSampleSkip</td>
<td>5.9</td>
</tr>
<tr>
<td>j</td>
<td>Turn on clock reference output. Note that the connector is shared with the clock reference input.</td>
<td>EnableClockRefOut</td>
<td>5.8</td>
</tr>
</tbody>
</table>

Figure 23: 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 13).

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 and MTCA.4 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 for both 5 GSPS and 10 GSPS modes. 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.

Note: The GPIO connector configuration is different for different options, Table 4.

<table>
<thead>
<tr>
<th>CONNECTOR WITH GPIO SUPPORT</th>
<th>GENERAL OPTION</th>
<th>INTERFACE OPTION</th>
<th>–USB</th>
<th>–PCIE</th>
<th>–PXIE</th>
<th>–MTCA</th>
<th>–10GBE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIG</td>
<td>Any</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SYNC</td>
<td>Any</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GPIO connector</td>
<td>Any</td>
<td>–</td>
<td>Yes</td>
<td>Yes</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 4: GPIO function availability.

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 24.

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
<th>USER COMMAND</th>
<th>REF</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>The external pin is automatically connected to the activated function.</td>
<td></td>
<td>6.2</td>
</tr>
<tr>
<td>b</td>
<td>The GPIO input function always reads the state of the pin. The GPIO output function is activated by the user.</td>
<td>SetDirectionGPIOPort</td>
<td>6.2</td>
</tr>
<tr>
<td>c</td>
<td>The user may access the pin by reading and writing from the software.</td>
<td>ReadGPIOPort, WriteGPIOPort</td>
<td>6.3</td>
</tr>
<tr>
<td>d</td>
<td>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.</td>
<td>ReadUserRegister, WriteUserRegister</td>
<td>6.4</td>
</tr>
</tbody>
</table>

Figure 24: 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

This section is only valid for ADQ7 hardware with options –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 bidirectional signal, 2 LVDS inputs, 4 LVDS outputs and 2 LVDS clock capable inputs. The GPIO connections are described in Figure 27.

The structure for the single-ended pins is shown in Figure 25 and for the LVDS pins in Figure 26.

![Diagram of GPIO connections](image)

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

Figure 25: ADQ7 structure of each of the 12 single-ended pins available with options –PCIe and –PXIe.

![Diagram of LVDS connections](image)

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

Figure 26: ADQ7 structure of each of the 12 single-ended pins available with options –PCIe and –PXIe.
<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
<th>16</th>
<th>17</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GPDIC0_P LVDS input / clock</td>
<td>GND</td>
<td>GND</td>
<td>GPIO0</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>GPDIC0_N LVDS input / clock</td>
<td>GND</td>
<td>GND</td>
<td>GPIO1</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>GPD11_P LVDS input</td>
<td>GND</td>
<td>GND</td>
<td>GPIO2</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>GPD11_N LVDS input</td>
<td>GND</td>
<td>GND</td>
<td>GPIO3</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>GPD12_P LVDS input</td>
<td>GND</td>
<td>GND</td>
<td>GPIO4</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>GPD12_N LVDS input</td>
<td>GND</td>
<td>GND</td>
<td>GPIO5</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GPDIC3_P LVDS input / clock</td>
<td>GND</td>
<td>GND</td>
<td>GPIO6</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GPDIC3_N LVDS input / clock</td>
<td>GND</td>
<td>GND</td>
<td>GPIO7</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>GPD04_P LVDS output</td>
<td>GND</td>
<td>GND</td>
<td>GPIO8</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>GPD04_N LVDS output</td>
<td>GND</td>
<td>GND</td>
<td>GPIO9</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>GPD05_P LVDS output</td>
<td>GND</td>
<td>GND</td>
<td>GPIO10</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GPD05_N LVDS output</td>
<td>GND</td>
<td>GND</td>
<td>GPIO11</td>
<td>Single-ended 3.3 V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>GPD06_P LVDS output</td>
<td>GND</td>
<td>GND</td>
<td>VDD</td>
<td>3.3V 250mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>GPD06_N LVDS output</td>
<td>GND</td>
<td>GND</td>
<td>VDD</td>
<td>3.3V 250mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>NC</td>
<td>GND</td>
<td>GND</td>
<td>VDD</td>
<td>3.3V 250mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 27: ADQ7 GPIO connector on ADQ7–PXle 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 28 shows the flow of data through acquisition 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 29, 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 5. 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.

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 30. 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 FPGA. This requires the ADQ Development Kit.

Data transfer
Triggers
Continuous data
(at reduced data rate)

Figure 30: 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.

Data transfer
Triggers
Records

Figure 31: 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.

Data transfer
Record data
Triggers
Records

Figure 32: 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.
Block diagram and timing of a streaming data transfer is in Figure 33 and Figure 34. 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 recorded 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.
Note that the boxes for each stage in Figure 34 are drawn a little shorter than 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 straightforward 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 35.
The timing of a user-scheduled multi-record transfer is in Figure 36. The acquisition is set up and armed. The user’s software checks for available records. The available records can be transferred with the \texttt{GetDataWHTS} command\footnote{WHTS stand for ‘with header and timestamp’}.

### DESCRIPTION

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

Figure 35: Block diagram of user-scheduled data transfer.
7.4.3 Transfer buffers

The data transfer buffers are the kernel buffers in Figure 33 and Figure 35. 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 33 and Figure 35 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. For FWPD, the record length may vary between records. The data buffer size cannot be matched to the data. The example code in ADQAPI_example1 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 37. 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.
7.6 Record header

7.6.1 Metadata

The record header contains the information described in Table 6. Example code is available to pack this information into a header and write to disk. The example C-code defines a struct which reads the header information from the header buffer.

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

Table 6: Header data.

7.6.2 Record Status

The status parameters indicate if the record was transferred correctly, Table 7.

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.

### Table 7: Status.

<table>
<thead>
<tr>
<th>OVER RANGE</th>
<th>FIFO FILL</th>
<th>LOST DATA</th>
<th>DESCRIPTION</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>No over-range detected</td>
<td>Anywhere in the data.</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>FIFO fill factor</td>
<td>Steps of 12.5%. 111 means &gt; 87.5% and 000 is below 12.5%.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>No lost data</td>
<td>The software has generated a header with no data to indicate a lost record.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Lost record in beginning of the record</td>
<td>The start of the record is missing. Timestamp information is also missing.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Lost record in the middle of the record</td>
<td>One or many section(s) anywhere inside the data is missing.</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Lost record in the end of the record</td>
<td></td>
</tr>
</tbody>
</table>

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-0404”. 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.

7.6.7 Data format

The header data parameter 'Data format' (Table 6) informs the user on how to interpret data. Allowed values are given in Table 8.

### Table 8: Data formats.

<table>
<thead>
<tr>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16 bits data word in two’s complement</td>
</tr>
<tr>
<td>1</td>
<td>32 bits data word in two’s complement</td>
</tr>
</tbody>
</table>

7.6.8 Record length

This is the length of the data record in number of 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 38 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.

---

#### Figure 38: Under-range in the ADC calibration.

**Table 7.7.1**: Description of components in the diagram.

<table>
<thead>
<tr>
<th>#</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>ADC with offset.</td>
</tr>
<tr>
<td>b</td>
<td>Digital offset calibration.</td>
</tr>
<tr>
<td>c</td>
<td>Digital baseline stabilizer that dynamically adjust baseline to zero. That is adjust the offset. Under-range has already occurred prior to this stage.</td>
</tr>
<tr>
<td>d</td>
<td>Acquisition create a header which contain over/under-range indication.</td>
</tr>
<tr>
<td>e</td>
<td>The analog input signal with offset. The ADC also contribute to the offset.</td>
</tr>
<tr>
<td>f</td>
<td>Digital calibration cannot account for offset on the analog signal. Under-range occur when signal is out of range on the negative side.</td>
</tr>
<tr>
<td>g</td>
<td>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.</td>
</tr>
<tr>
<td>h</td>
<td>Overflow indication is set when over/under-range occur and is kept for the record.</td>
</tr>
</tbody>
</table>
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 if 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, –MTCA, 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 6) 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 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] 16-1692 ADQ7DC Datasheet
[3] 08-0214 ADQAPI User guide
[4] 18-2104 ADQ7-FWATD application note
[5] 17-1957 ADQ7-FWATD user guide
[6] 18-2059 ADQ7 updater user guide
[7] 18-2118 FWPD application note
[8] 18-2132 ADQ7-FWPD user guide
[9] 16-1790 Application note: Connecting a detector to a 14-bit digitizer
[10]
Important Information

Teledyne Signal Processing Devices Sweden AB (Teledyne SP Devices) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to Teledyne SP Devices’ general terms and conditions supplied at the time of order acknowledgment.

Teledyne SP Devices warrants that each product will be free of defects in materials and workmanship, and conform to specifications set forth in published data sheets, for a period of one (1) year. The warranty commences on the date the product is shipped by Teledyne SP Devices. Teledyne SP Devices’ sole liability and responsibility under this warranty is to repair or replace any product which is returned to it by Buyer and which Teledyne SP Devices determines does not conform to the warranty. Product returned to Teledyne SP Devices for warranty service will be shipped to Teledyne SP Devices at Buyer’s expense and will be returned to Buyer at Teledyne SP Devices’ expense. Teledyne SP Devices will have no obligation under this warranty for any products which (i) has been improperly installed; (ii) has been used other than as recommended in Teledyne SP Devices’ installation or operation instructions or specifications; or (iii) has been repaired, altered or modified by entities other than Teledyne SP Devices. The warranty of replacement products shall terminate with the warranty of the product. Buyer shall not return any products for any reason without the prior written authorization of Teledyne SP Devices.

In no event shall Teledyne SP Devices be liable for any damages arising out of or related to this document or the information contained in it.

TELEDYNE SP DEVICES’ EXPRESS WARRANTY TO BUYER CONSTITUTES TELEDYNE SP DEVICES’ SOLE LIABILITY AND THE BUYER’S SOLE REMEDY WITH RESPECT TO THE PRODUCTS AND IS IN LIEU OF ALL OTHER WARRANTIES, LIABILITIES AND REMEDIES. EXCEPT AS THUS PROVIDED, TELEDYNE SP DEVICES DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

TELEDYNE SP DEVICES DOES NOT INDEMNIFY, NOR HOLD THE BUYER HARMLESS, AGAINST ANY LIABILITIES, LOSSES, DAMAGES AND EXPENSES (INCLUDING ATTORNEY’S FEES) RELATING TO ANY CLAIMS WHATSOEVER. IN NO EVENT SHALL TELEDYNE SP DEVICES BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFIT, LOST DATA AND THE LIKE, DUE TO ANY CAUSE WHATSOEVER. NO SUIT OR ACTION SHALL BE BROUGHT AGAINST TELEDYNE SP DEVICES MORE THAN ONE YEAR AFTER THE RELATED CAUSE OF ACTION HAS ACCRUED. IN NO EVENT SHALL THE ACCRUED TOTAL LIABILITY OF TELEDYNE SP DEVICES FROM ANY LAWSUIT, CLAIM, WARRANTY OR INDEMNITY EXCEED THE AGGREGATE SUM PAID TO SP BY BUYER UNDER THE ORDER THAT GIVES RISE TO SUCH LAWSUIT, CLAIM, WARRANTY OR INDEMNITY.

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

Copyright © 2019 Teledyne Signal Processing Devices Sweden AB. All rights reserved, including those to reproduce this publication or parts thereof in any form without permission in writing from Teledyne SP Devices.