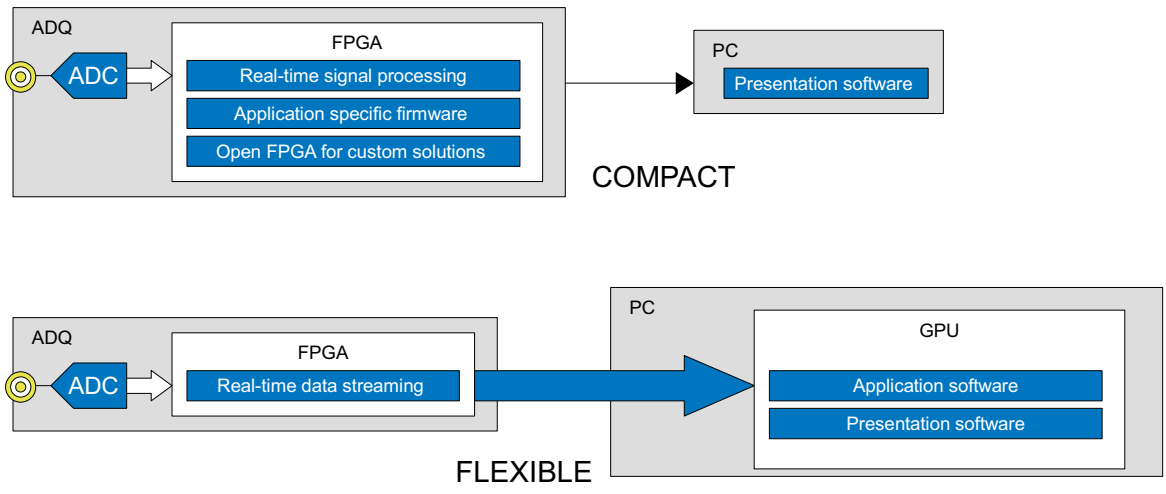
ADQ system design for data processing Application Note

The ADQ digitizer series offer a complete high speed and high resolution digitizing solution. The data processing is as important as the acquisition, and this application note describes ways of processing the large amount of data that the ADQ-series of digitizers produce. These steps complete the acquisition chain from detector to application. The application note contain three parts:

- *Open Field-programmable gate array (FPGA) for real-time on-board processing.*
- *Peer-to-peer streaming for processing in a graphical processing unit (GPU).*
- *Data rate management.*



1 Introduction

1.1 Scope of the application note

Parts of this application note are valid for ADQ14 and parts for ADQ7.

1.2 Concept #1: on-board FPGA processing

The ADQ digitizers produce large amounts of data at very high speeds. The ADQ14 and ADQ7 can produce up to 8 and 20 GBytes/s of raw data respectively whereas the available host PC interfaces can handle up to approximately 6.8 GBytes/s of data[1]. This mismatch in data rate means that the full raw data set can only be processed inside the FPGA.

The ADQ-series of digitizers has an open FPGA which means that parts of its resources are available for implementation of custom real-time signal processing which operate on the full data set. This processing is also often combined with some type of data reduction which helps adjust the data rate to the capacity of the PC interface and at the same time significantly lowers the requirements on the host PC. This is illustrated in **Figure 1** and the concept is described further in-depth in **Section 1.5**.

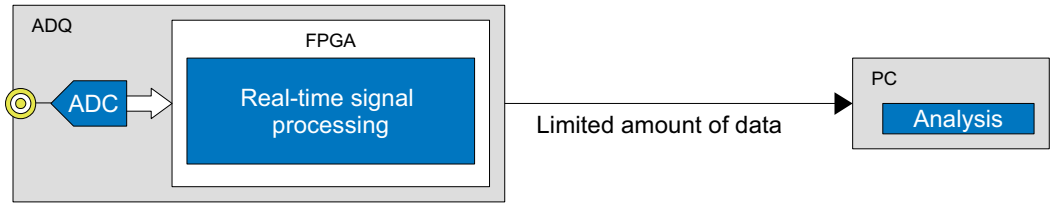


Figure 1: Main processing in the on-board FPGA.

1. This is the present commonly used Generation 3 by 8 lanes PCIe interface. The continuing development of both digitizers and PC interfaces will increase all data rates over time. It is expected that the data acquisition rate will continue to stay ahead of the data transfer rate for high-end digitizers.

1.3 Concept #2: peer-to-peer streaming to GPU

The FPGA on the ADQ digitizer is crucial for processing data and for supporting data streaming at very high rates. However there are situations where the fixed number of parallel processing elements can impose limitations on which types of algorithms that can be implemented. One such example is Fast Fourier transform (FFT), where it can be challenging to implement long FFTs inside the on-board FPGA.

As a complement to the on-board processing concept the ADQ-series is therefore also available with a high speed data interface for streaming data to a processing unit. For some applications this can be the central processing unit (CPU) of the host PC whereas other cases require higher computational performance. For these situations it can be beneficial to use a GPU. These boards share similarities with FPGAs through their massively parallel structure and their capability of performing efficient high-performance computing, however one benefit with GPUs is that they do not require knowledge about hardware description languages (HDL) such as VHDL or Verilog but are instead programmed in more commonly used languages such as C.

In order not to load the host PC with streaming data, the ADQ digitizers support peer-to-peer streaming at up to 6.8 GBytes/s. Large sets of data can therefore be transferred at high rates to the GPU for additional processing, see **Figure 2**. The concept is described further in-depth in **Section 1.8**.

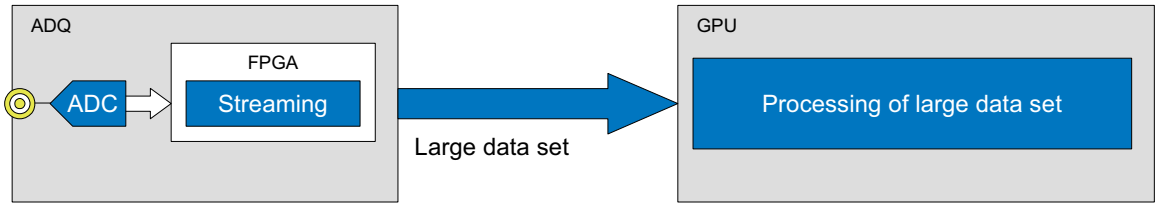


Figure 2: Main processing in GPU via peer-to-peer streaming.

1.4 Comparison of the methods

The two processing methods are preferred for different types of tasks and applications, see **Table 1**. The application examples shown in the table are further described in **Section 3**.

Examples of hardware and firmware products suitable for use in different applications and for specific measurement scenarios are listed in **Table 2**.

CASE	REAL-TIME PROCESSING IN FPGA		STREAMING TO GPU		
	ADVANTAGE	DISADVANTAGE	ADVANTAGE	DISADVANTAGE	REF
Large FFT		Size limitation	“No” size limit		3.1
Small FFT	Efficient		“No” size limit		3.1
SDR	DDC on all data	Limited flexibility for radio decoder	C-programming of radio decoder	Reduced data set for DDC	3.2
Peak analysis	All data available			Reduced data set	3.3
Image analysis		Random access of multi-dimensional image	Random access of multi-dimensional image		3.4
Multi-dimensional filter		Linear access only	Matrix array of processor elements		3.5
OCT	Compact system			Limited data set	3.6

Table 1: Comparison of processing method for different applications.

APPLICATION	PRODUCT		COMPUTATION
	ADQ	FIRMWARE	
Large FFT	ADQ14	–FWDT	Streaming peer-to-peer to GPU
Large FFT	ADQ7	–FWDAQ –FWSDR	Streaming peer-to-peer to GPU
OCT	ADQ14	–FWDT	Streaming peer-to-peer to GPU
OCT	ADQ14OCT	–FWOCT	In FPGA
Random Pulses	ADQ14 ADQ7	–FWPD	In FPGA
3D Images	ADQ14	–FWDT	Streaming peer-to-peer to GPU
3D Images	ADQ7	–FWDAQ	Streaming peer-to-peer to GPU
RF Recording	ADQ7	–FWSDR	Streaming peer-to-peer to GPU

Table 2: Examples of suitable products for different applications and measurements.

1.5 Processing in on-board FPGA

The main advantage of processing data in the FPGA is to perform real-time processing on the full data set in order to identify and extract key information and - as a result - reduce the amount of data to be transferred to the host PC. The data reduction is often significant and therefore cabled standard interfaces such as USB 3.0 and 10 Gigabit Ethernet (GbE) can be used.This allows for the digitizer to be placed close to the detector. With this approach the analog signal cables can be kept short which in turn reduces analog design issues, for example reflections.

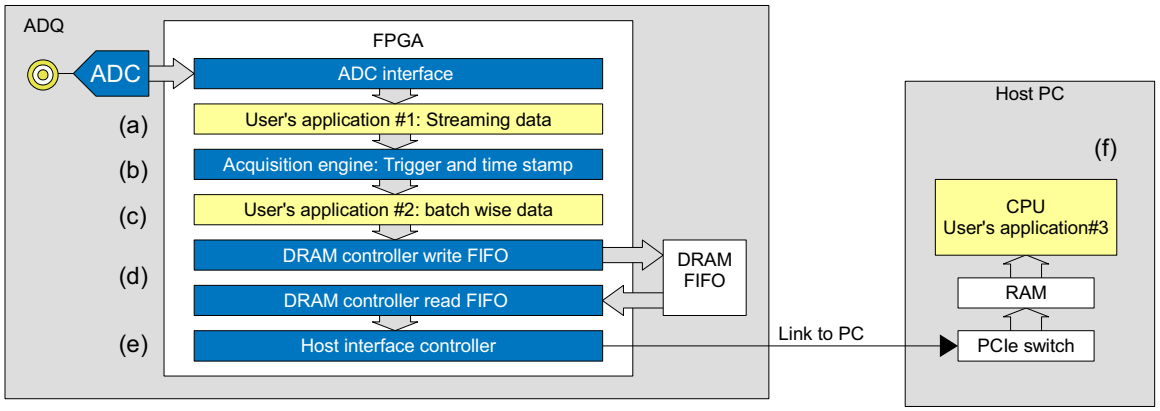
1.6 Processing algorithm alternatives

The powerful open FPGA of the ADQ is available both for standard application-specific signal processing as well as custom real-time algorithms. The application support is provided in several levels:

- Stand-alone application-specific firmware packages are available for several advanced applications. These packages include commonly used functions for different measurement scenarios and often include some type of built-in data reduction. Available firmware packages include:
  - Pulse detection (FWPD) where zero-suppression reduces the data set to the samples of interest only. Data without interest, between the pulses, are discarded.
  - Averaging (FWATD) where accumulation of samples reduce the data rate.
  - Digital down conversion (FWSDR), where bandwidth control limits the data rate.
  - The firmware for swept-source optical coherence tomography (SS-OCT) implements k-space re-mapping so that the k-clock does not have to be forwarded to the PC.
- Custom real-time algorithms can be implemented via a firmware development kit which opens up the FPGA to the user.
- Teledyne SP Devices' design service can provide custom firmware solutions for short time-to-market (TTM).

1.7 Compact system design

The firmware development kit in combination with multiple stand-alone firmware packages offers great flexibility for real-time signal processing and data reduction. It also allows for the use of standard interfaces that simplify system-level integration. The digitizers can therefore be used with for example a laptop, a tablet, or an embedded computer to create compact solutions. The data flow for the on-board FPGA signal processing concept is illustrated in **Figure 3**.



#	DESCRIPTION
a	The full raw data stream from the analog-to-digital converter (ADC) is made available to the user via the firmware development kit. Algorithms that operate on the full data set must be implemented in the first user logic area (see the block diagram).
b	The acquisition engine add trigger and time stamp information to the data and splits it into batches of consecutive samples, so called records.
c	In the second user logic area the data consist of records and the processing hence operate on batches of data.
d	The DRAM controller and the DRAM FIFO acts as a buffer to handle bursts of data and ensure that no data is lost during transfer to the host.
e	The interface to the host PC.
f	Further processing can be done on the CPU of the host computer. This is often a combination of additional computations as well as visualization of the result.

Figure 3: Custom processing in the FPGA.

1.8 Processing in GPU

GPUs are affordable yet very powerful processing tools and they are programmed in commonly used programming languages. Integrating both the digitizer and the GPU into the same PC results in a very powerful and compact system capable of high-performance computing. However, it is crucial that the digitizers supports peer-to-peer streaming in order to both optimize data transfer performance and at the same time minimize the involvement of - and load on - the CPU of the host computer.

1.9 Processing algorithms

The architecture of a GPU is optimized to process vast amounts of data in parallel. A typical partitioning between the GPU and the CPU is therefore that the main signal processing is done in the GPU whereas the control and monitoring of the overall system is done in the CPU. The former may still consume significant CPU resources and it is therefore important to minimize CPU load caused by other processes/tasks.

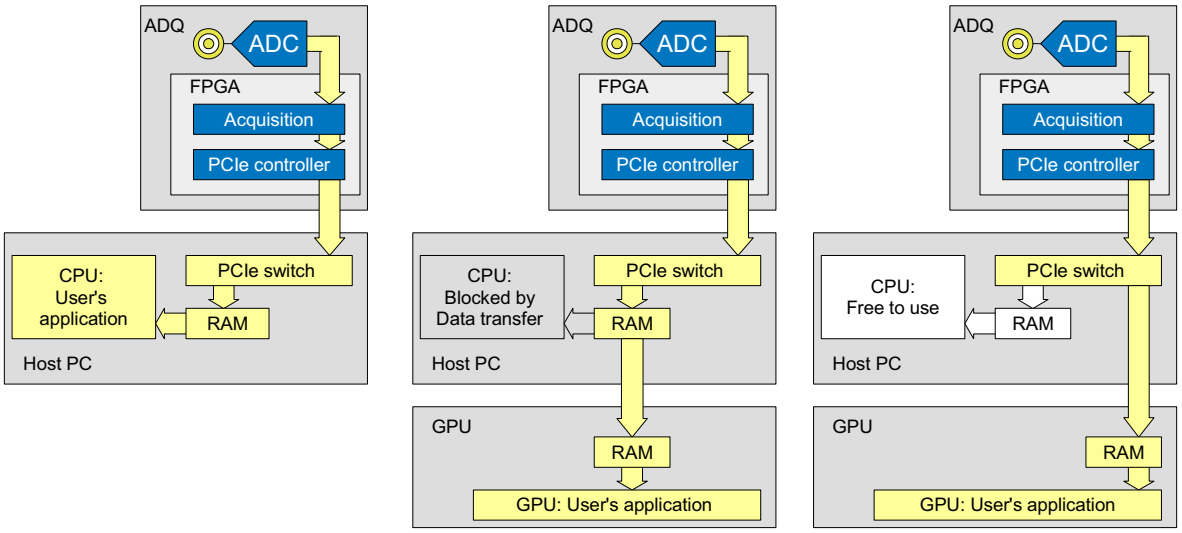
1.10 System design with peer-to-peer streaming

These systems consists of a host PC with multiple PCI Express (PCIe) slots.The digitizer and GPU are inserted into two of these slots. Data transfers from the digitizer to the GPU should be done at very high sustained rates and it is therefore crucial to minimize any potential negative impact caused by other parts of the system. The solution is to use a so called peer-to-peer streaming approach and the method is best described in a number of steps:

The fundamental data management in a PC is that the CPU controls all the data flow between different parts and that the data always passes through the random access memory (RAM) of the PC. This is the normal mode of operation when data is processed by the CPU of the host PC, **Figure 4** (a). This is sufficient in many cases as long as the CPU is capable of processing the data fast enough. However, in applications which require more advanced and computationally heavy signal processing a GPU may be required in order to support the high data throughput and to perform complex calculations fast enough so that it does not introduce performance bottlenecks.

When adding a GPU, the CPU is no longer used for calculations. However, the load on the CPU is still significant as the data transfer between the digitizer and the GPU is done via the PCs RAM. Due to the high data rates both the CPU and the RAM are occupied by the scheduling and copying of the data, **Figure 4** (b).

The most efficient way to do this,is to use peer-to-peer streaming. The data flow is then controlled by the PCIe switch and the data stream is connected directly from the ADQ to the GPU, **Figure 4** (c). The data transfer does not load neither the CPU nor the RAM and the host PC can therefore be used for other tasks.



(a) Processing in CPU

(b) Processing in GPU
Streaming via host

(b) Processing in GPU
Peer-to-peer streaming

#	DESCRIPTION	REF
a	Streaming of data directly to the user's application which is running on the CPU of the host PC. This is a common solution but the performance is limited by the performance of the CPU that offer a low degree of parallelism.	2
b	Streaming to GPU via the CPU and RAM of the PC. This is a straightforward method which uses the native drivers and application programming interfaces of the digitizer and the GPU. The solution is therefore widely adopted but has the drawback that the CPU is occupied - or even blocked - by the transfer of data. This approach often introduce unnecessary performance bottlenecks.	
c	Data is streamed directly to the GPU via the PCIe switch and does not load the CPU. This is the most efficient implementation but it is rarely supported by digitizer manufacturers as it requires custom software that enable optimized data transfer at very high rates. Selected digitizers from Teledyne SP Devices offers this functionality.	e.g. 3.4, 3.5

Figure 4: Data flow examples.

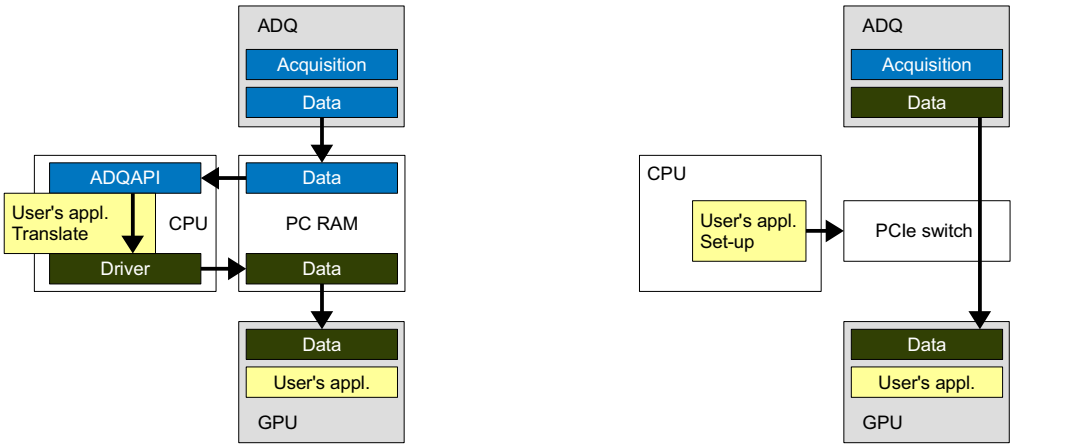
1.11 Transfer data from ADQ to GPU

The basic method for data transfer via PC RAM is easily integrated into the system and there are very few limitations on which hardware to use:

- The ADQ and the ADQAPI delivers data to the host PC in a readable format.
- The user's application software translate the data to the format required by the GPU.
- The GPU driver handles the data transfer to the GPU.

The specific details of each respective hardware is managed by its corresponding driver, **Figure 5** (a).

In peer-to-peer streaming, however, the digitizer requires direct access to the GPU. The driver for the GPU has to be able to set up the transfer and the ADQ has to produce correctly formatted output data. This requires specialized low-level drivers and is only available for a limited set of GPUs and under specific conditions. In this situation, the user's software only set up the communication and does not operate on the data stream, **Figure 5** (b).



(a) Processing in GPU
Streaming via host

(b) Processing in GPU
Peer-to-peer streaming

#	DESCRIPTION
a	The user's application in the CPU translate the ADQ data to the GPU format and uses the standard GPU driver. This is the most flexible method and work with most hardware combinations.The downside is that the load on the CPU can be significant.
b	The ADQ has to write data in the correct format for the specific GPU. The user's application in the CPU must have access to the setup of the GPU driver to configure the transfer directly.

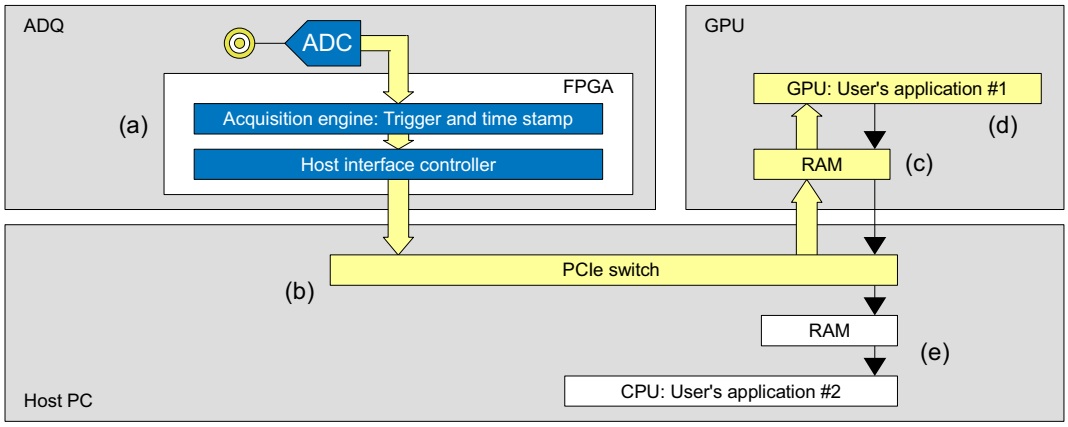
Figure 5: Software for data flow control.

1.12 Processing flow

Although the transfer rate to the GPU is up to 6 GBytes/s, some data rate reduction still has to be done in the FPGA. The most common type of data reduction is triggering, see **Figure 6**, where the average data rate is reduced by capturing a record of limited length at each trigger event. See **Section 4.1**. This is handled by the built-in acquisition engine in the ADQ.

The records are then transferred to the GPU via the PCIe backplane for further processing in software (denoted user's application #1 in **Figure 6**).This software may implement additional data reduction, for example by identifying and/or calculating key parameters, such as time-domain peaks or frequency contents in the incoming data records.

The resulting parameter set can be returned from the GPU to the host CPU and be used for additional tasks, such as automatic system control or visualization. This is implemented in user's application software #2.



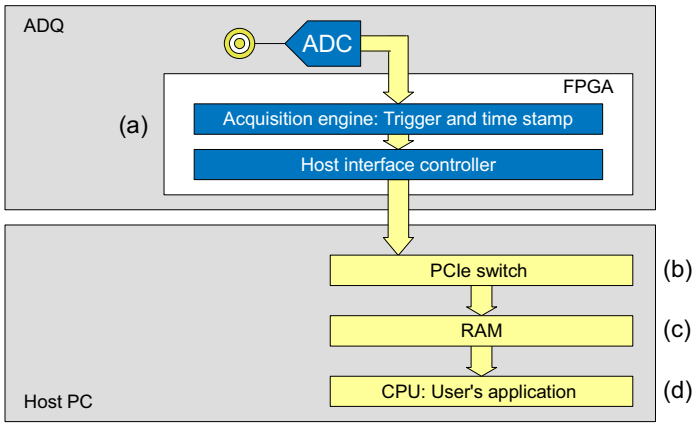
#	DESCRIPTION	REF
a	The acquisition engine implements the trigger logic which allows the captured ADC data to be separated into individual records of fixed length. The data between the captured records is discarded and thereby the total amount of data forwarded to subsequent processing steps is reduced.	4
b	The streaming PCIe interface of the ADQ is connected to the PCIe switch on the motherboard in the host PC. The peer-to-peer streaming application re-directs the data directly to the GPU without loading the CPU in the host PC.	
c	Data is directly received in the RAM of the GPU via the PCIe switch.	
d	Computationally heavy signal processing is done in the user's application that run on the GPU. Key parameters can be extracted and forwarded to the host PC.	
e	The GPU output is sent to the CPU for additional tasks such as control and visualization.	

Figure 6: Block diagram for processing on a GPU.

2 Processing data in CPU

2.1 Architecture

The ADQ digitizers also support streaming to CPU RAM and this is the normal mode of operation, **Figure 7**. The general system design principle is the same as for a GPU solution, but the CPU has other ways of operating than a GPU and is therefore suitable for other types of algorithms. However, the trade-off between FPGA processing and CPU is the same as the trade-off between FPGA and GPU.



#	DESCRIPTION	REF
a	The acquisition engine implements the trigger logic which allows the captured ADC data to be separated into individual records of fixed length. The data between the captured records is discarded and thereby the total amount of data forwarded to subsequent processing steps is reduced.	4
b	The streaming PCIe interface of the ADQ is connected to the PCIe switch on the motherboard in the host PC. The streaming application directs the data to the RAM of the host PC.	
c	Data is received in the RAM of the host PC.	
d	The signal processing in the user's application is runon the CPU of the host PC.	

Figure 7: Block diagram for processing on the CPU of the host PC.

2.2 Note on ADQ14-FWDT

This note is only valid for ADQ14-FWDT. This not is NOT applicable on any other firmware and should be ignored for all other firmwares.

This firmware (ADQ14-FWDT) has a very small data buffer on the FPGA. The link to the PC is therefore very sensitive to interruptions in the PC and also to DRAM bandwidth in the PC.

The number of accesses to the DRAM memory in the PC has to be carefully managed. For example, transferring data without headers simplifies the processing and improve reliability. However, without headers, there is no lost package indication. Then the user has to poll the FPGA to verify that no data was lost.

3 Application examples

3.1 Frequency event trigger using FFT

A frequency triggered system is used for frequency surveillance or fault detection of short and rarely occurring signal spurs. This example illustrates trade-offs between FPGA and GPU.

The core operation in the application is a FFT which is a batch wise processing task. All data is recorded into a memory buffer and butterfly operations are performed on the data batch. There are FFT IPs available for implementation in an FPGA but the number of FFT bins that can be supported might not be enough. A GPU on the other hand, has a huge array of processing elements, which makes it ideal for this type of operations.

Another aspect of the application is the ability to detect a short event. The frequency event trigger is based on an event in the frequency domain, that is a signal that at some point in time appear within a certain frequency band. In spectral monitoring this could for example be an unlicensed radio transmitter. The frequency event trigger is implemented using FFT and a frequency mask that specify a certain power level for each individual FFT bin that should act as a trigger threshold. The FFT calculates the power per bin for each record but if a signal appear in the time duration between captured records, it may be lost and hence remain undetected. It is therefore beneficial to use so called overlapping FFTs since the overlap ensures that there are no gaps between FFTs during which appearing signals could remain undetected.Overlapping FFTs can be implemented in the on-board FPGA and even signals of very short time duration can thus be detected, see **Figure 8**.

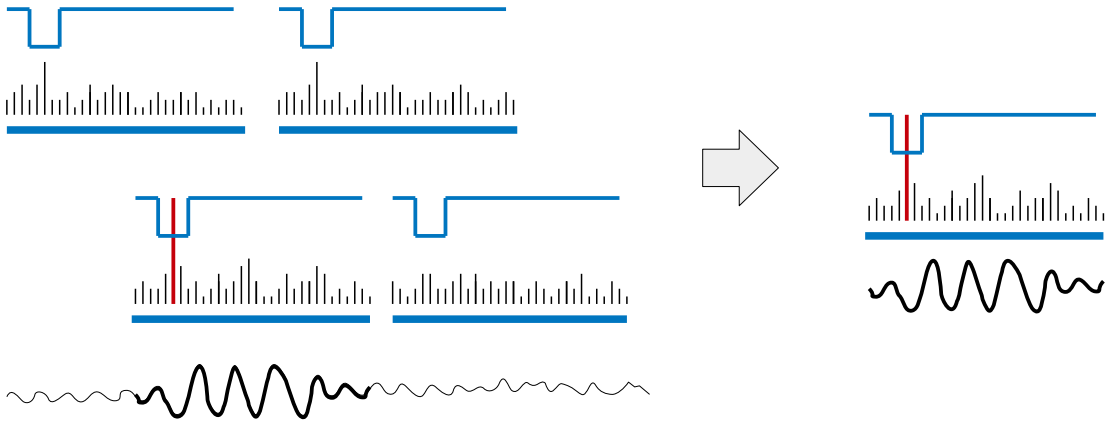


Figure 8: Overlapping FFT in the FPGA capture a pulse with signal power above the threshold specified by the frequency mask. This generate a trigger event and one record is stored. Time-domain illustrations are shown below their frequency domain counterparts.

The benefit of implementing FFTs in the on-board FPGA is that it can operate on the entire raw data stream. However, limited resources such as multipliers and memory may notbe enough for the required FFT length (number of bins). Furthermore the implementation if typically using fixed-point arithmetics and it can be challenging to achieve sufficient accuracy. For long and high accuracy FFTs it can therefore be beneficial to instead use a GPU. Due to its massively parallel structure a GPU can perform a vast number of floating-point operations per second. Onthe other hand, the data may have to be reduced in order to match the data transfer capacity of the PCIe link. Data reduction is often done by

using a trigger to capture selected parts of the signal, discarding the rest, **Figure 9**.

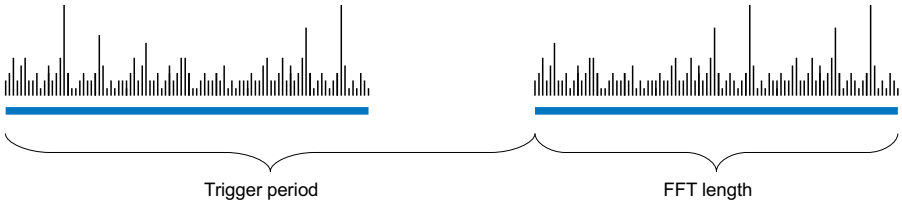


Figure 9: FFT in GPU with a timing gap. The FFTs are accurate, but may miss information occurrig during the gap.

Another way is to limit the frequency span of interest. By combining the Digital Down Converter (DDC) in the software defined radio firmware package (–FWSDR) with an FFT calculation in the GPU an optimal solution is found,see **Figure 10**.

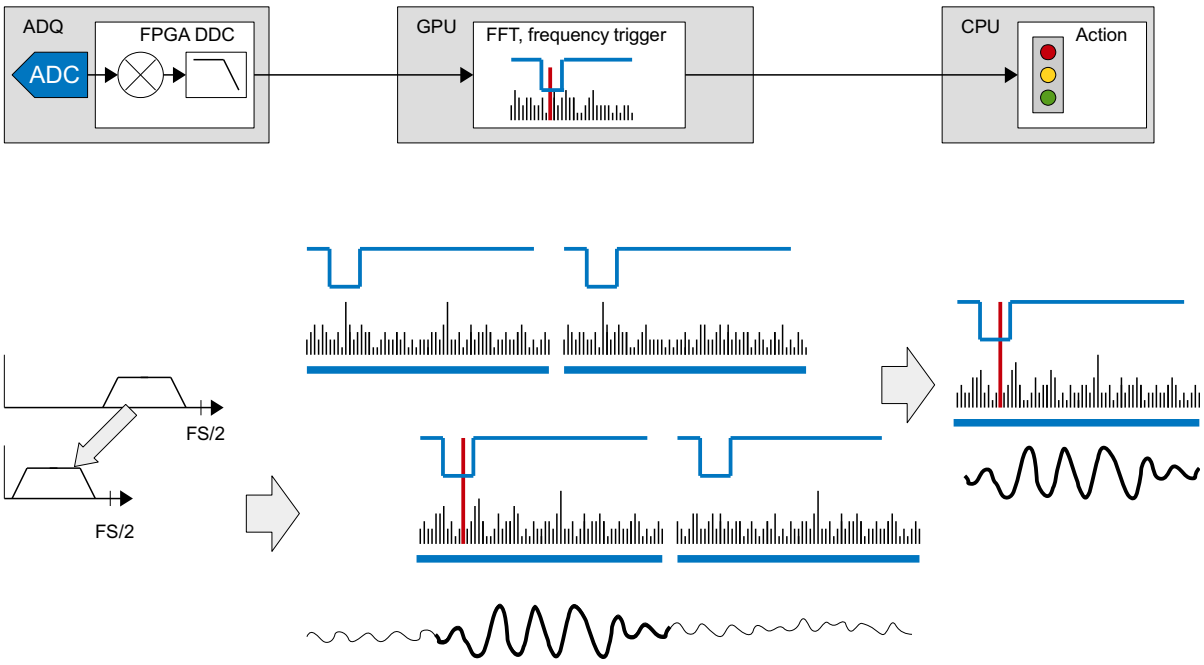


Figure 10: Partitioned frequency trigger system.

Example 1: ADQ7 has a data transfer rate of 5GBytes/s. With two bytes per sample, this gives 2.5 GSPS and 1.25 GHz bandwidth. Due to the transition band of the decimation filters, the effective bandwidth is 1 GHz. The DDC can select any 1 GHz frequency band within the analog bandwidth of the ADQ7. Streaming this data to the GPU means that overlapping long FFTs with bandwidth of 1 GHz can be produced.

3.2 Software defined radio

An software defined radio (SDR) consists of two parts; the Digital Down Converter (DDC) and the radio decoder. The DDC consists of a quadrature mixer, a digital filter, and a decimator (sample skip). Decimation means reducing the data rate by discarding samples. As the sample rate is reduced, so is the data rate. To avoid signal quality loss through aliasing, the filter operation can be performed. Note that this method also reduces the bandwidth. An illustration of SDR is given **Figure 16**.The DDC is preferably performed in an FPGA as the algorithms are of streaming type that operate on only a few samples at the time. The radio decoder implements the actual modulation scheme and this is preferably done in the GPU as the algorithms are complex. The data rate for a single radio channel is assumed to be significantly lower than the raw sampling rateThereby,the DDC in the FPGA can reduce the data rateenough so that the result can be streamed to a GPU without loss of information. This is illustrated in **Figure 11**.

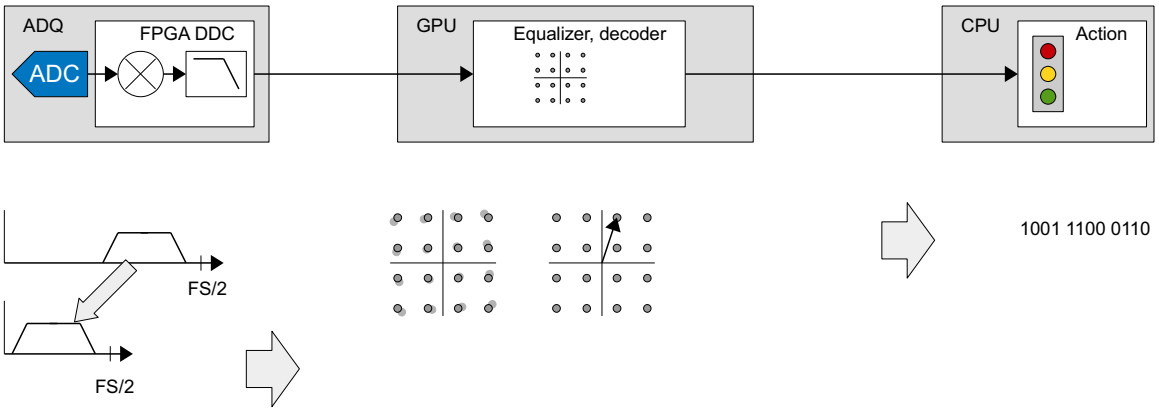


Figure 11: Partitioned SDR system.

3.3 Peak detection and analysis

Peak detection and analysis is intended for data reduction as early as possible in the signal chain. This is an automatic classification of signals which is available in the firmware package FWPD. This classification can detect an event and extract an extremely reduced data set. An alternative for heavy processing on the data set is to use the level trigger and transfer pulses to the GPU for analysis, **Figure 12**.

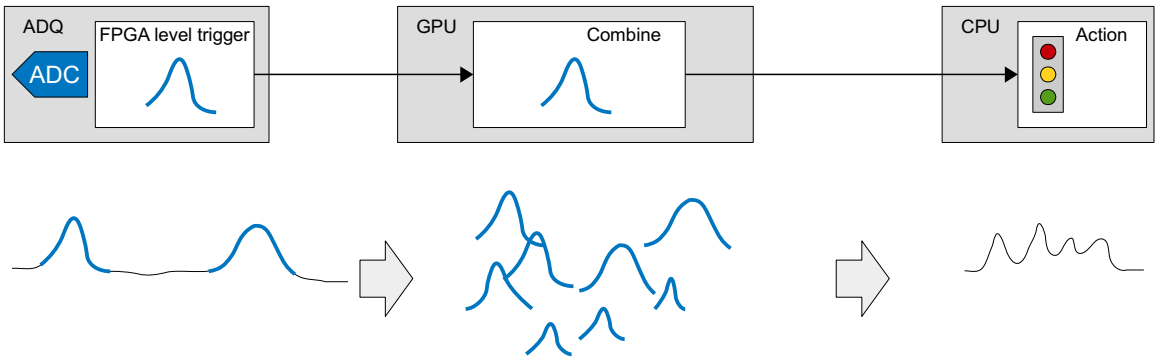


Figure 12: Partitioned pulse analysis system.

3.4 Image analysis

Image processing is a multi-dimensional procedure, but the recording of an image takes one pixel or line at the time. The processing involves performing computations on multiple lines and part of lines to form areas. This is suitable for GPU implementation where the many processing elements can handle pixels and neighboring pixels, see **Figure 13**.

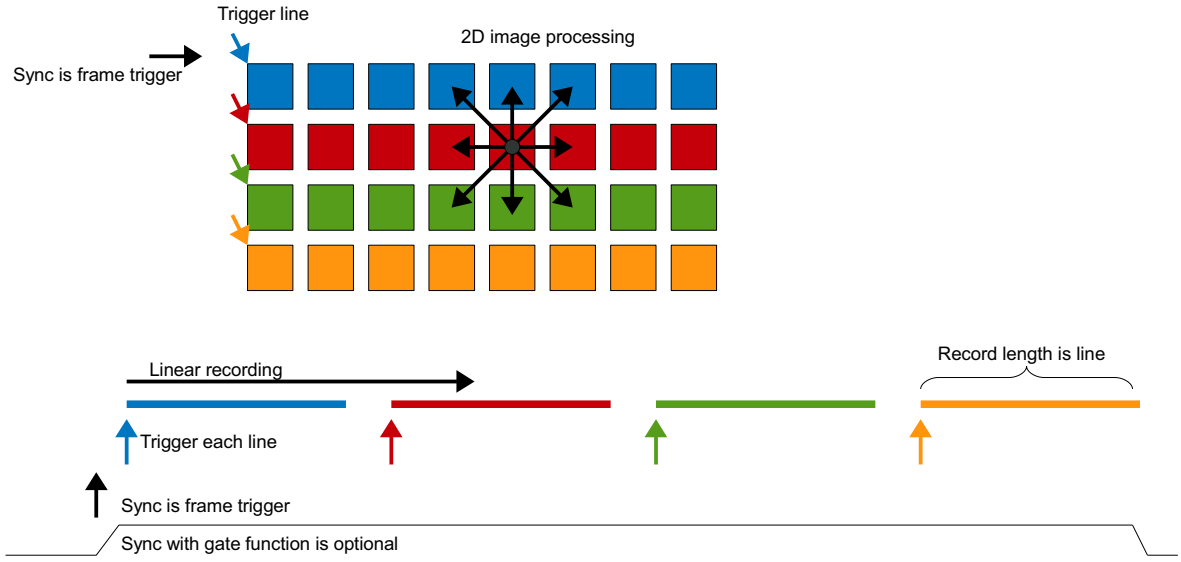


Figure 13: Linear recording and matrix processing.

3.5 Multi-dimensional filtering

Multi-dimensional filtering is similar to image processing. The FPGA is excellent for one-dimensional data processingbut for multi-dimensional tasks, a GPU is preferred.

3.6 Swept-source OCT

Swept-source OCT (SS-OCT) is characterized by two parts; a non-uniform sampling and an FFT. The re-mapping of uniformly sampled data onto a non-uniform grid, (using the k-clock) is a task with several aspects. With internal re-mapping, the clock is not needed in the host and thus the data out from the ADQ is halved. On the other hand, if the re-mapping is done in the GPU, both data and k-clock has to be transfered. The advantage of the high-speed peer-to-peer streaming is then obvious.

The applications uses an FFT for creating the image. The FFT on the FPGA is limited in size, but generally big enough for the SS-OCT application.

There are many other application-specific algorithms such as dispersion compensation which requires computational resources. To determine the optimal implementation, an overall system analysis is required. Teledyne SP Devices supports on-board k-space re-mapping and FFT through ADQ14OCT and streaming peer-to-peer to GPU through ADQ14-FWDT.A diagram of the three-dimensional recording is shown in **Figure 14**.

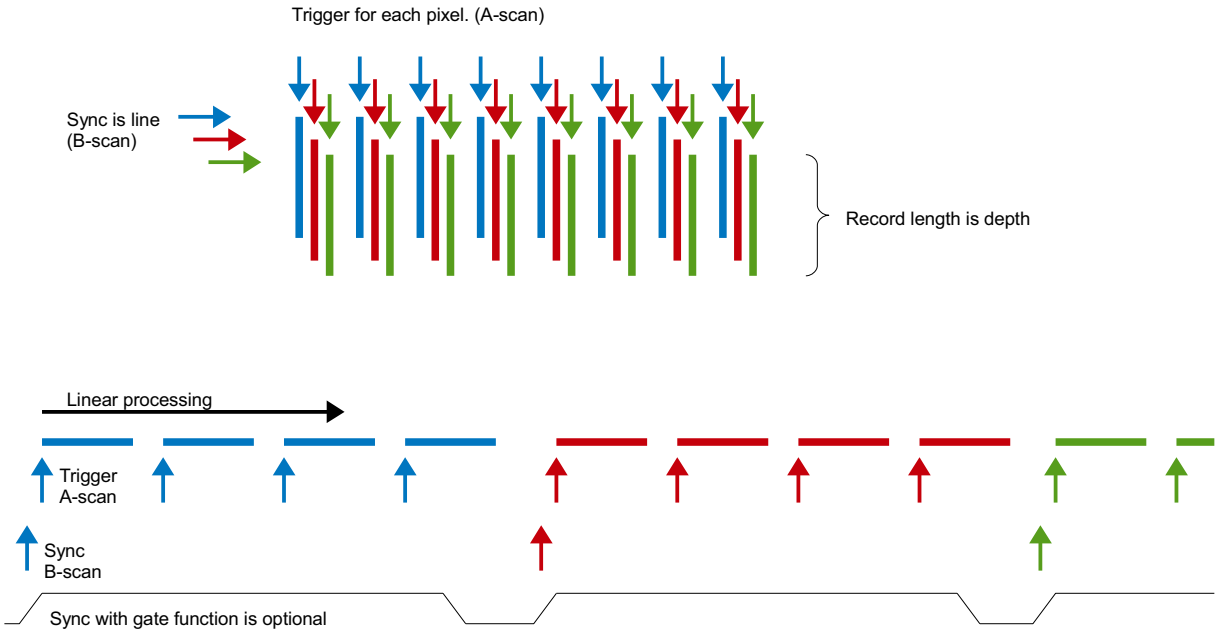


Figure 14: 3D-recording of an OCT image.

4 Tutorial

4.1 Data rate adaptation

The data rate from the ADCs on an ADQ7 can be up to 20GBytes/s whereas the data rate over PCIe is limited to a lower number. The actual rate depends on the type of PCIe interface where, for example, PCIe x8 Gen3 support 6.8 GBytes/s.

The data rate from the ADC hence has to be adopted to the streaming capacity and there are several ways of achieving that. Here are some common examples:

Trigger: At each trigger event a record of a limited number of samples is captured. The record size and the trigger rate is then used for controlling the average data rate. During the recording, the data is captured at full speed. The average data rate is reduced by [Duty cycle] = [record length] / [Trigger period], **Figure 15**.

Example 2: The ADQ7 produces 20 GBytes/s of raw data. The streaming capacity to the host is 6 GByte/s. The maximum allowed duty cycle is then 6/20 = 30%.

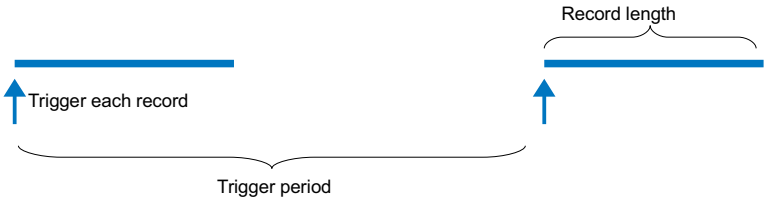


Figure 15: Data rate adaptation by trigger.

Filter and decimation: Decimation means reducing the data rate by discarding samples. As the sample rate is reduced, so is the data rate. To avoid signal quality loss through aliasing, a filter operation can be performed. Note that this method reduce the bandwidth, **Figure 16**.

Example 3: The ADQ7 produces 20 GBytes/s of raw data. The required bandwidth is 1 GHz, which means at least 2 GSPS (Nyquist theorem). The DDC can do data rate reduction in 2^n, where n is an integer. Setting n=4 gives $20/2^4$ = 2.5 GSPS. This means 5 GBytes/s which can be streamed continuously to the host via the Gen3x8 PCIe interface.

Figure 16: Data rate adaptation in frequency.

Channel selection: By selecting a subset of channels, the data rate is reduced.

*Example 4: Selecting two of the four channels of an ADQ14DC–4C–FWDT produces 1GSPS * 2 * 2 = 4 GBytes/s. The Gen3x8 interface of the FWDT firmware option can stream that continuously to the host PC.*

4.2 Streaming processed data

Some algorithms,e.g. digital filters, are time-invariant. They operate identically on all samples and are therefore typically suitable for implementation in the FPGA. The algorithm only uses a subset of all the samples during the calculation. For example, an FIR filter with 17 taps needs access to 17 samples. A peak detection algorithm only needs access to one sample at the time.

If the algorithm is performed continuously and indefinitely, it should be implemented in user logic 1 of the firmware development kit (see **Figure 3**). If there is a time constraint, such as finding a peak within a certain record, this needs to be implemented in user logic 2 and framed by the record.

4.3 Batch processing

Batch wise processing is algorithms that are *not* the same for all samples. An example of this is the FFT. Batch wise processing in general need access to a large amount of data during the entire computation and such algorithms are often more suitable for implementation in a GPU. Note that some operations, for example peak detection, may appear to be batch wise, since they are limited in time by the size of the record (batch). However, this operation in itself only looks at one (or a few) samples at the time; data is streaming through the algorithm. The batch then only indicates when to start and stop. This section (**4.3**) refers to operations where all data from the entire batch is used.

4.4 Multi-dimensional signal processing

Random access algorithms are also preferably performed in a GPU as samples can be addressed in any order. This is similar to 2D and 3D image computations. An FPGA is good at one-dimensional processing.

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