# Technical Information Manual

Revision n. 5

06 February 2012

**MOD. DT5720**

*4 CHANNEL 12 BIT*

*250 MS/S DIGITIZER*

**MANUAL REV.5**

**NPO:**

**00100/09:5720x.MUTx/05**

CAEN will repair or replace any product within the guarantee period if the Guarantor declares that the product is defective due to workmanship or materials and has not been caused by mishandling, negligence on behalf of the User, accident or any abnormal conditions or operations.

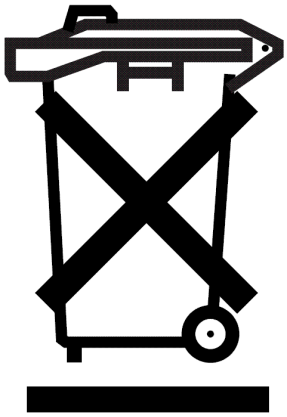
**CAEN declines all responsibility for damages or injuries caused by an improper use of the Modules due to negligence on behalf of the User. It is strongly recommended to read thoroughly the CAEN User's Manual before any kind of operation.**

CE

*CAEN reserves the right to change partially or entirely the contents of this Manual at any time and without giving any notice.*

## Disposal of the Product

*The product must never be dumped in the Municipal Waste. Please check your local regulations for disposal of electronics products.*



**MADE IN ITALY** : We stress the fact that all the boards are made in Italy because in this globalized world, where getting the lowest possible price for products sometimes translates into poor pay and working conditions for the people who make them, at least you know that who made your board was reasonably paid and worked in a safe environment. (this obviously applies only to the boards marked "MADE IN ITALY", we can not attest to the manufacturing process of "third party" boards).

# TABLE OF CONTENTS

# *LIST OF FIGURES*

## *LIST OF TABLES*

# 1. General description

## 1.1. Overview


**Fig. 1.1: Mod. DT5720 Desktop Waveform Digitizer**

The Mod. DT5720 is a 4 Channel 12 bit 250 MS/s Desktop Waveform Digitizer with 2 Vpp dynamic range on single ended MCX coax. input connectors.

The digitizer is avalilable in the following versions:

- Mod. DT5720 - 4 Channel version mounting Cyclone EP1C4 FPGA.
- Mod. DT5720A - 2 Channel version mounting Cyclone EP1C4 FPGA.
- Mod. DT5720B - 4 Channel version mounting Cyclone EP1C20 FPGA.
- Mod. DT5720C - 2 Channel version mounting Cyclone EP1C20 FPGA.

The DC offset adjustment (± 1V range) on each channel by 16bit DACs allows a right sampling of a bipolar (Vin=± 1V) up to a full positive (Vin= 0 ÷ +2V) or negative (Vin= 0 ÷ -2V) analog input swing without losing dynamic resolution.

The module features a front panel clock In and a PLL for clock synthesis from internal/external references.

The data stream is continuously written in a circular memory buffer. When triggered, the FPGA writes further N samples for the post trigger and freezes the buffer that can be read via USB or optical link. The acquisition can continue dead-timeless in a new buffer.

Each channel has a SRAM memory buffer (1.25 M Samples/ch) divided in buffers of programmable size (1 - 1024). The readout (from USB or Optical link) of a frozen buffer is independent from the write operations in the active circular buffer (ADC data storage).

Zero suppression and data reduction algorithms allow substantial savings in data amount readout and processing, rejecting samples smaller than programmable thresholds.

DT5720 supports multi-board synchronization: an external reference clock can be distributed to all modules (CLK IN) and a common input (GPI) can be used to align all event trigger time tags

DT5720 houses USB 2.0 and optical link interfaces. USB 2.0 allows data transfers up to 30 MB/s. The Optical Link supports transfer rate of 80 MB/s, and offer daisy-chain capability. Therefore it is possible to connect up to 8 ADC modules to an A2818 Optical Link Controller or 32 modules to an A3818 (4 channel version).

CAEN provides also for this model a Digital Pulse Processing firmware for Physics Applications. This feature allows to perform on-line processing on detector signal directly digitized. DT5720 is well suited for data acquisition and processing of signals from scintillators/photomultipliers or SiPM detectors, implementing function functionalities of a digital QDC.

**Table 1.1: Available items**

| Code | Description |
|---|---|
| WDT5720XAAAA | DT5720 - 4 Ch. 12 bit 250 MS/s Digitizer: 1.25MS/ch, C4, SE |
| WDT5720AXAAA | DT5720A - 2 Ch. 12 bit 250 MS/s Digitizer: 1.25MS/ch, C4, SE |
| WDT5720BXAAA | DT5720B - 4 Ch. 12 bit 250 MS/s Digitizer: 1.25MS/ch, C20, SE |
| WDT5720CXAAA | DT5720C - 2 Ch. 12 bit 250 MS/s Digitizer: 1.25MS/ch, C20, SE |
| WA654XAAAAAA | A654 - Single Channel MCX to LEMO Cable Adapter |
| WA654K4AAAAA | A654 KIT4 - 4 MCX TO LEMO Cable Adapter |
| WA654K8AAAAA | A654 KIT8 - 8 MCX TO LEMO Cable Adapter |
| WA659XAAAAAA | A659 - Single Channel MCX to BNC Cable Adapter |
| WA659K4AAAAA | A659 KIT4 - 4 MCX TO BNC Cable Adapter |
| WA659K8AAAAA | A659 KIT8 - 8 MCX TO BNC Cable Adapter |
| WA2818XAAAAA | A2818 - PCI Optical Link |
| WA3818AXAAAA | A3818A - PCIe 1 Optical Link |
| WA3818BXAAAA | A3818B - PCIe 2 Optical Link |
| WA3818CXAAAA | A3818C - PCIe 4 Optical Link |
| WAI2730XAAAA | AI2730 - Optical Fibre 30 m. simplex |
| WAI2720XAAAA | AI2720 - Optical Fibre 20 m. simplex |
| WAI2705XAAAA | AI2705 - Optical Fibre 5 m. simplex |
| WAI2703XAAAA | AI2703 - Optical Fibre 30cm. simplex |
| WAY2730XAAAA | AY2730 - Optical Fibre 30 m. duplex |
| WAY2720XAAAA | AY2720 - Optical Fibre 20 m. duplex |
| WAY2705XAAAA | AY2705 - Optical Fibre 5 m. duplex |
| WFWDPPCIAAAA | DPP-CI - Digital Pulse Processing for Charge Integrationon for (x720) |
| WFWDPPCI02AA | DPP-CI - Digital Pulse Processing for Charge Integration for (x720) - 2 Licence Pack |
| WFWDPPCI05AA | DPP-CI - Digital Pulse Processing for Charge Integration for (x720) - 5 Licence Pack |
| WFWDPPCI10AA | DPP-CI - Digital Pulse Processing for Charge Integration for (x720) - 10 Licence Pack |
| WFWDPPCI20AA | DPP-CI - Digital Pulse Processing for Charge Integration for (x720) - 20 Licence Pack |

## 1.2. Block Diagram

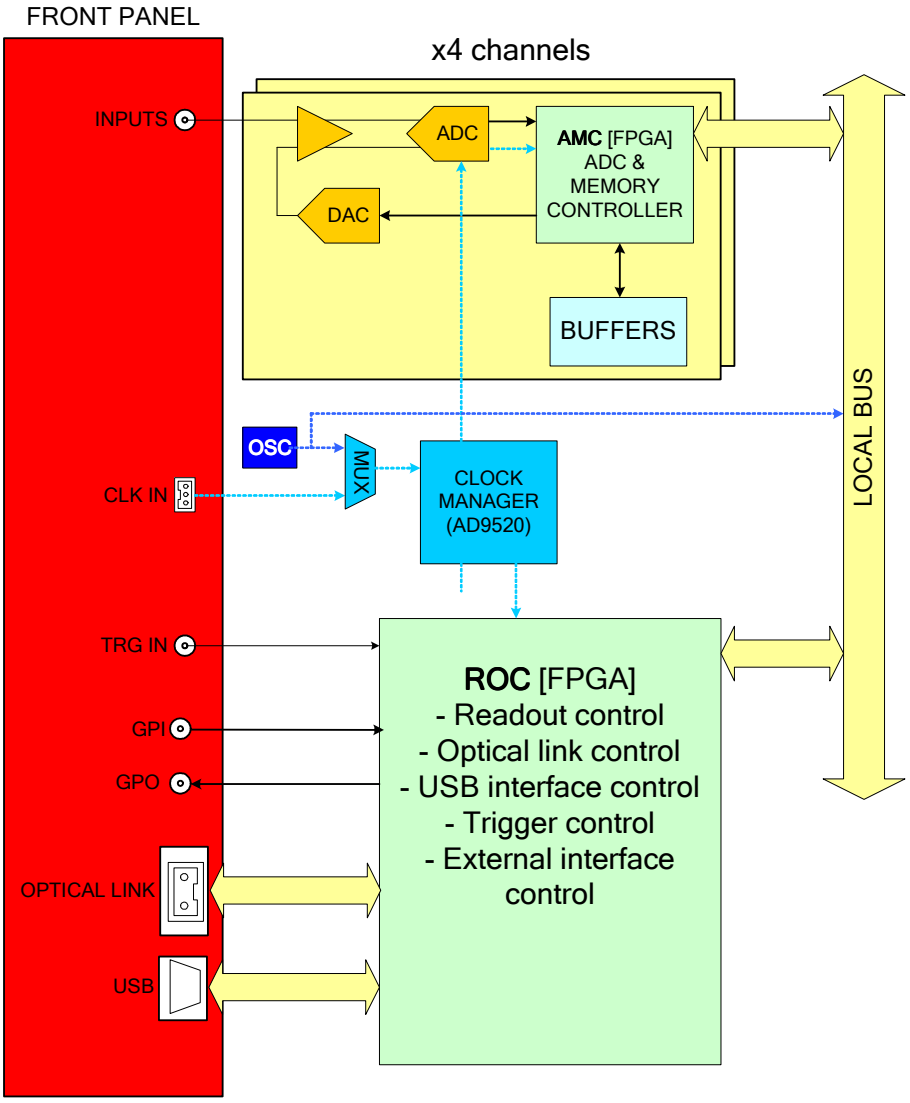



**Fig. 1.1: Mod. DT5720 Block Diagram**

The function of each block will be explained in detail in the subsequent sections.

# 2. Technical specifications

## 2.1. Packaging and Compliancy

The unit is a Desktop module housed in a 154x50x164 $mm^3$ alloy box.

## 2.2. Power requirements

The module is powered via the external AC/DC stabilized power supply (Meanwell GS40A12-P1J 40W, 12V DC Output, 3.34A).
N.B.: the provided power supply may result unfit to some applications requiring low noise, therefore, in such cases, the digitizer shall be powered with a linear +12V power supply; the power jack is a 2.1mm type, a suitable cable is the RS 656-3816 type (or similar).

## 2.3. Front and Back Panel

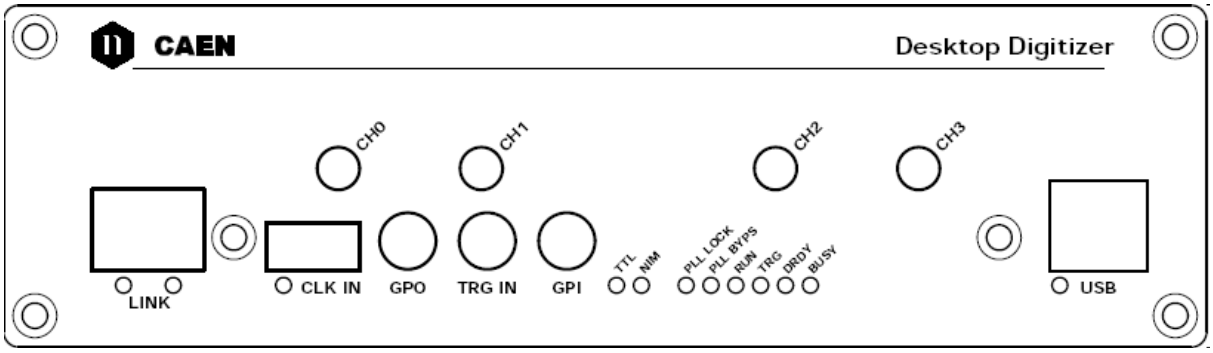



**Fig. 2.1: Mod. DT5720 front panel**

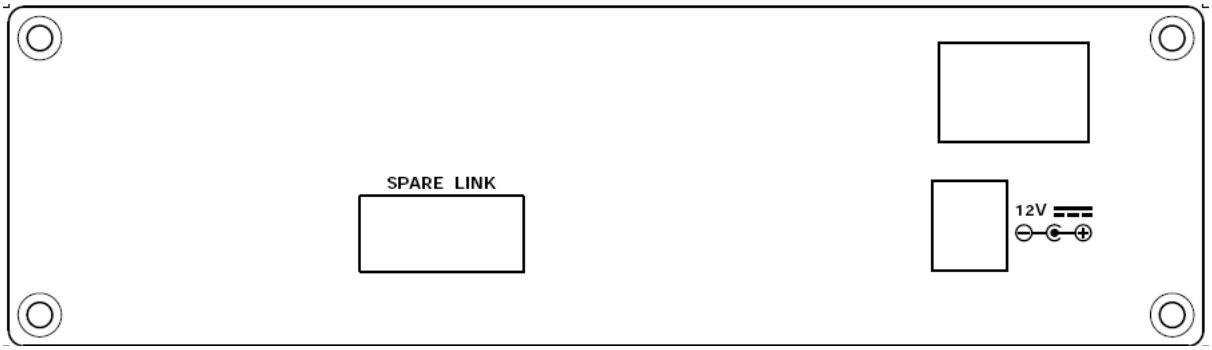



**Fig. 2.2: Mod. DT5720 back panel**

## 2.4. External connectors

### 2.4.1. ANALOG INPUT connectors


**Fig. 2.3: MCX connector**

*Function*:
Analog input, single ended, input dynamics: 2Vpp Zin=50Ω
*Mechanical specifications*:
MCX connector (CS 85MCX-50-0-16 SUHNER)

### 2.4.2. CONTROL connectors

*Function*:
TRG IN: External trigger input (NIM/TTL, Zin= 50Ω)
*Mechanical specifications*:
00-type LEMO connectors

### 2.4.3. ADC REFERENCE CLOCK connectors

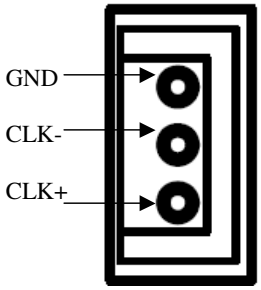



**Fig. 2.4: AMP CLK IN Connector**

*Function:*
CLK IN: External clock/Reference input, AC coupled (diff. LVDS, ECL, PECL, LVPECL, CML), Zdiff= 110Ω.
*Mechanical specifications:*
AMP 3-102203-4 AMP MODUII

### 2.4.4. Digital I/O connectors

*Function:*
- GPI: programmable front panel input (NIM/TTL, Zin=50Ω)

- GPO: programmable front panel output (NIM/TTL across 50Ω); used as output for trigger propagation

*Mechanical specifications*:
00-type LEMO connectors

### 2.4.5. Optical LINK connector

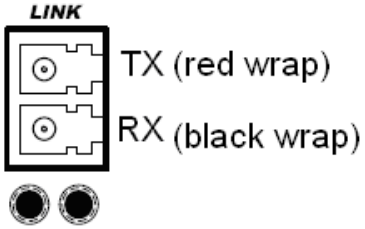



**Fig. 2.5: LC Optical Connector**

*Mechanical specifications:*
LC type connector; to be used with Multimode 62.5/125µm cable with LC connectors on both sides
*Electrical specifications:*
Optical link for data readout and slow control with transfer rate up to 80MB/s; daisy chainable.

### 2.4.6. USB Port

*Mechanical specifications:*
B type USB connector
*Electrical specifications:*
USB 2.0 and USB 1.1 compliant

### 2.4.7. 12V External

*Mechanical specifications:*
RAPC722X SWITCHCRAFT PCB DC Power Jack
*Electrical specifications:*
+12V DC Input

### 2.4.8. Spare Link

*Mechanical specifications:*
3M-7610-5002 connector
*Electrical specifications:*
T.B.D.

## 2.5. Other components

### *2.5.1. Displays*

The front panel hosts the following LEDs:

**Table 2.2: Front panel LEDs**

| Name: | Colour: | Function: |
|---|---|---|
| **CLK_IN** | green | External clock enabled. |
| **NIM** | green | Standard selection for GPO, TRG IN, GPI. |
| **TTL** | green | Standard selection for GPO, TRG IN, GPI. |
| **USB** | green | Data transfer activity |
| **LINK** | green/yellow | Network present; Data transfer activity |
| **PLL _LOCK** | green | The PLL is locked to the reference clock |
| **PLL _BYPS** | green | The reference clock drives directly ADC clocks; the PLL circuit is switched off and the PLL_LOCK LED is turned off. |
| **RUN** | green | RUN bit set (see § 5.18) |
| **TRG** | green | Triggers are accepted |
| **DRDY** | green | Event/data (depending on acquisition mode) are present in the Output Buffer |
| **BUSY** | red | All the buffers are full |

# Technical specifications table

**Table 2.3: Mod. DT5720 technical specifications**

| | |
|---|---|
| **Packaging** | Desktop module; 154x50x164 $mm^3$ (WxHxD), Weight: 680 gr |
| **Analog Input** | 4 channels (MCX 50 Ohm) for DT5720 and DT5720B models<br>2 channels (MCX 50 Ohm) for DT5720A and DT5720C models.<br>Single-ended<br>Input range: 2 Vpp;<br>Bandwidth: 125 MHz.<br>Programmable DAC for Offset Adjust x ch., adjustment range: ±1V |
| **Digital Conversion** | Resolution: 12 bit; Sampling rate: 10 to 250 MS/s simultaneously on each channel |
| **ADC Sampling Clock generation** | Three operating modes:<br>- PLL mode - internal reference (50 MHz loc. oscillator).<br>- PLL mode - external reference on CLK_IN (Jitter<100ppm).<br>- PLL Bypass mode: Ext. clock on CLK_IN drives directly ADC clocks (Freq.: 10 ÷ 250 MHz). |
| **Digital I/O** | CLK_IN (AMP Modu II):<br>- AC coupled differential input clock LVDS, ECL, PECL, LVPECL, CML (single ended NIM/TTL available with cable adapter)<br>- Jitter<100ppm<br>TRG_IN (LEMO, NIM/TTL, Zin = 50 Ohm)<br>GPI (LEMO, NIM/TTL, Zin = 50 Ohm)<br>GPO (LEMO, NIM/TTL, across 50 Ohm) |
| **Memory Buffer** | 1.25 M sample/ch Multi Event Buffer<br>Programmable event size and pre-post trigger.<br>Divisible into 1 ÷ 1024 buffers.<br>Readout of Frozen buffer independent from write operations in the active buffer (ADC data storage). |
| **Trigger** | Common Trigger<br>- TRG_IN (External signal)<br>- Software (from USB or Optical Link)<br>- Self trigger (Internal threshold auto-trigger)<br>Daisy chain trigger propagation among boards (using GPO) |
| **Trigger Time Stamp** | 32bit – 8ns (34s range) |
| **Multi Modules Synchronization** | Allows data alignment and consistency across multiple DT5720 modules:<br>- CLK_IN allows the synchronization to a common clock source<br>- GPI ensures Trigger time stamps and start acquisition times alignment |
| **USB interface** | USB2.0 and USB1.1 compliant<br>Up to 30 MB/s transfer rate |
| **Optical Link** | CAEN proprietary protocol, up to 80 MB/s transfer rate, with Optical Link Controller (Mod. A2818/A3818). |
| **Upgrade** | Firmware can be upgraded via Optical Link or USB interface |
| **Software** | General purpose C and LabView Libraries<br>Demo and Software Tools for Windows and Linux |
| **Electrical Power** | Voltage range: 12 ± 10% Vdc |

# 3. Functional description

## 3.1. Analog Input

Input dynamics is 2V (Zin= 50 Ω). A 16bit DAC allows to add a DC offset to the signal in the ±1 V range.
The input bandwidth ranges from DC to 125 MHz (with 1st order anti-aliasing low pass filter).

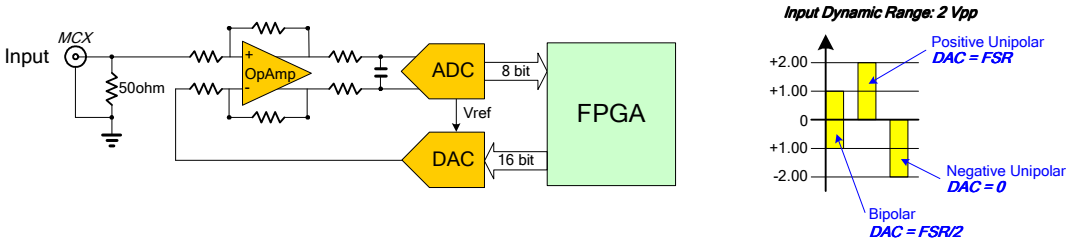



**Fig. 3.1: Input diagram**

## 3.2. Clock Distribution

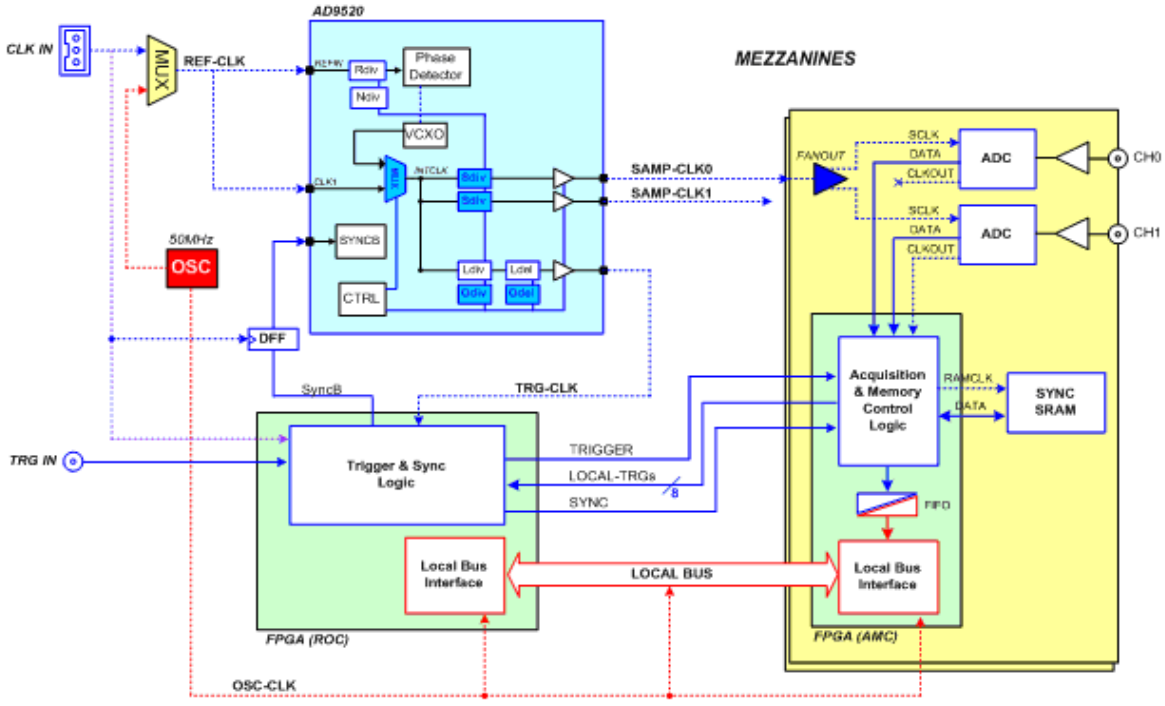



**Fig. 3.2: Clock distribution diagram**

The module clock distribution takes place on two domains: OSC-CLK and REF-CLK; the former is a fixed 50MHz clock provided by an on board oscillator, the latter provides the ADC sampling clock.

OSC-CLK handles Local Bus (communication between motherboard and mezzanine boards; see red traces in the figure above).
REF-CLK handles ADC sampling, trigger logic, acquisition logic (samples storage into RAM, buffer freezing on trigger) through a clock chain. Such domain can use either an external (via front panel signal) or an internal (via local oscillator) source, in the latter case OSC-CLK and REF-CLK will be synchronous (the operation mode remains the same anyway).
DT5720 uses an integrated phase-locked-loop (PLL) and clock distribution device (AD9520). It is used to generate the sampling clock for ADCs (SAMP-CLK0/SAMP-CLK1) and trigger logic synchronization clock (TRG-CLK).
Both clocks can be generated from the internal oscillator or from external clock input (CLK IN). By default, board uses the internal clock as PLL reference (REF-CLK). External clock can be selected by register access. AD9520 configuration can be changed and stored into non-volatile memory. AD9520 configuration change is primarly intended to be used for external PLL reference clock frequency change:
DT5720 locks to an external 50 MHz clock with default AD9520 configuration.

Please contact CAEN (support.frontend@caen.it) for more information and configuration tools.

Refer also to AD9520 data sheet for more details:
http://www.analog.com/UploadedFiles/Data_Sheets/AD9520.pdf

### 3.2.1. Trigger Clock

TRG-CLK signal has a frequency equal to ½ of SAMP-CLK; therefore a 2 samples “uncertainty” occurs over the acquisition window.

## 3.3. Acquisition Modes

### 3.3.1. Acquisition run/stop

The acquisition can be started in two ways, according to Acquisition Control register bit 0 setting (see § 5.17):
- setting the RUN/STOP bit (bit 2) in the Acquisition Control register (bit 0 of Acquisition Control must be set to REGISTER-CONTROLLED RUN MODE)
- driving GPI signal high (bit 0 of Acquisition Control must be set to 1, GPI CONTROLLED RUN MODE)

Subsequently acquisition is stopped either:
- resetting the RUN/STOP bit (bit 2 in the Acquisition Control register (bit 0 of Acquisition Control must be set to REGISTER-CONTROLLED RUN MODE)
- driving GPI signal low (bit 0 of Acquisition Control set to 1, GPI CONTROLLED RUN MODE)

### 3.3.2. Acquisition Triggering: Samples and Events

When the acquisition is running, a trigger signal allows to:
- store the 32 bit counter (represents a time reference) of the Trigger Time Tag (TTT), that runs at 1/2 of the sampling clock frequency; actually the capture of the trigger takes place every 2 clock cycles, thus the time resolution of the Trigger Time Tag is 2 clock cycles (20 ns). This means that the LSB of the TTT is always 0.

– increment the EVENT COUNTER (see § 5.26)
– fill the active buffer with the pre/post-trigger samples, whose number is programmable (Acquisition window width, § 5.22), freezing then the buffer for readout purposes, while acquisition continues on another buffer

**Table 3.1: Buffer Organization**

| REGISTER (see § 5.15) | BUFFER NUMBER | SRAM 1.25 MB/ch | |
|---|---|---|---|
| | | Std. | Pack2.5 |
| 0x00 | 1 | 1M | 1.25M |
| 0x01 | 2 | 512K | 640K |
| 0x02 | 4 | 256K | 320K |
| 0x03 | 8 | 128K | 160K |
| 0x04 | 16 | 64K | 80K |
| 0x05 | 32 | 32K | 40K |
| 0x06 | 64 | 16K | 20K |
| 0x07 | 128 | 8K | 10K |
| 0x08 | 256 | 4K | 5K |
| 0x09 | 512 | 2K | 2.5K |
| 0x0A | 1024 | 1K | 1.25K |

An event is therefore composed by the trigger time tag, pre- and post-trigger samples and the event counter.
Overlap between “acquisition windows” may occur (a new trigger occurs while the board is still storing the samples related to the previous trigger); this overlap can be either rejected or accepted (programmable via software).
If the board is programmed to accept the overlapped triggers, as the “overlapping” trigger arrives, the current active buffer is filled up, then the samples storage continues on the subsequent one.

In this case events will not have all the same size (see figure below).

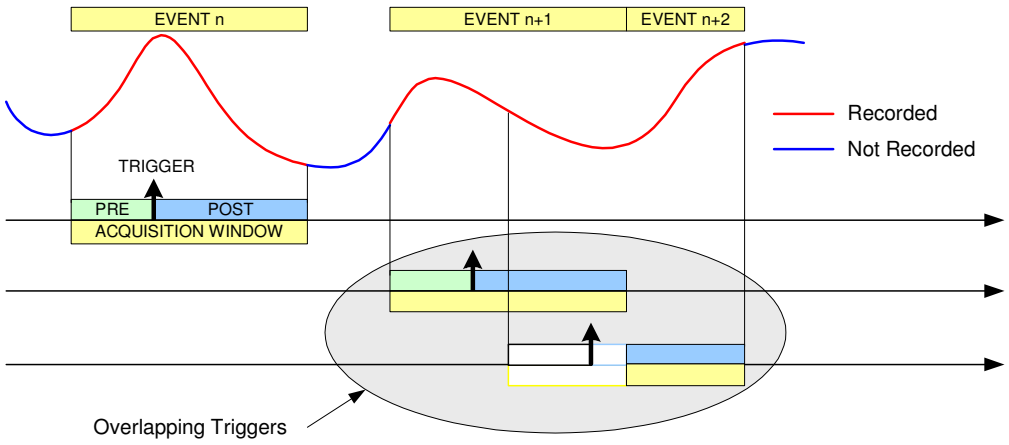



**Fig. 3.3: Trigger Overlap**

A trigger can be refused for the following causes:
– acquisition is not active

- memory is FULL and therefore there are no available buffers
- the required number of samples for building the pre-trigger of the event is not reached yet; this happens typically as the trigger occurs too early either with respect to the RUN_ACQUISITION command (see § 3.3.1) or with respect to a buffer emptying after a MEMORY_FULL status
- the trigger overlaps the previous one and the board is not enabled for accepting overlapped triggers

As a trigger is refused, the current buffer is not frozen and the acquisition continues writing on it. The Event Counter can be programmed in order to be either incremented or not. If this function is enabled, the Event Counter value identifies the number of the triggers sent (but the event number sequence is lost); if the function is not enabled, the Event Counter value coincides with the sequence of buffers saved and readout.

#### 3.3.2.1. Custom size events

It is possible to make events with a number of Memory locations, which depends on Buffer Organization register setting (see § 5.15) smaller than the default value. One memory location contains two ADC samples and the maximum number of memory locations $N_{LOC}$ is therefore half the maximum number of samples per block NS = 512K/Nblocks (640K/Nblocks when Pack2.5 mode is used).
Smaller $N_{LOC}$ values can be achieved by writing the number of locations $N_{LOC}$ into the Custom Size register (see § 5.16).
$N_{LOC}$ = 0 means “default size events”, i.e. the number of memory locations is the maximum allowed.
$N_{LOC}$ = N1, with the constraint 0<N1<½NS (0<N1<2/5NS with Pack2.5), means that one event will be made of 2·N1 samples (2.5·N1 samples with Pack2.5).

### *3.3.3. Event structure*

An event is structured as follows:
- Header (4 32-bit words)
- Data (variable size and format)

The event can be readout via Optical Link and/or USB; data format is 32 bit long word.

#### 3.3.3.1. Header

It is composed by four words, namely:
- Size of the event (number of 32 bit long words)
- Bit24; data format: 0= normal format; 1= *Zero Length Encoding* data compression method enabled; Channel Mask (=1: channels participating to event; ex CH2 and CH3 participating→Ch Mask: 0xC, this information must be used by the software to acknowledge which channel the samples are coming from)
- Event Counter: It is the trigger counter; it can count either accepted triggers only, or all triggers (see § 5.16).
- Trigger Time Tag: It is a 32 bit counter (31 bit count + 1 overflow bit), which is reset as acquisition starts and is incremented at each sampling clock hit. It is the trigger time reference.

#### 3.3.3.2. Samples

Stored samples; data from masked channels are not read.

### 3.3.3.3. Event format examples

The event format is shown in the following figure (case of 4 channels enabled, with *Zero Length Encoding* disabled and enabled respectively; see § 3.3.3.1 and § 3.4.1.2):
An event is structured as follows:

– identifier (Trigger Time Tag, Event Counter)
– samples caught in the acquisition windows

The event can be stored in the board memories (and can be readout via Optical Link) in two ways: data format is 32 bit long word, and each long_word may contain 2 samples (Standard mode) or “two and a half” (Pack2.5 mode), depending on Channel Configuration register setting (see § 5.12).
The event format is therefore one of the following:

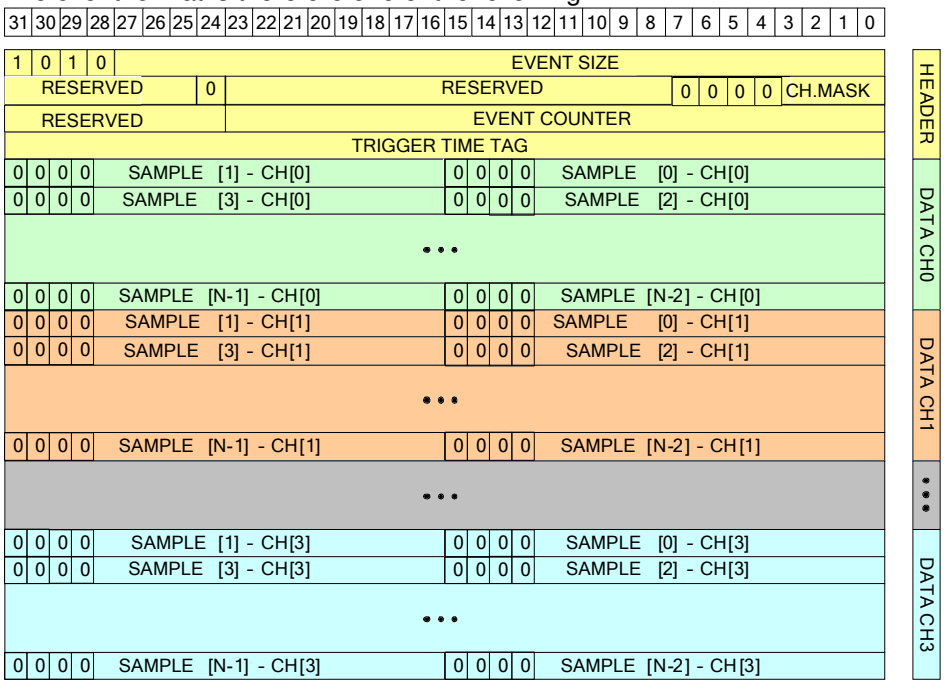



**Fig. 3.4: Event Organization (standard mode), normal format**

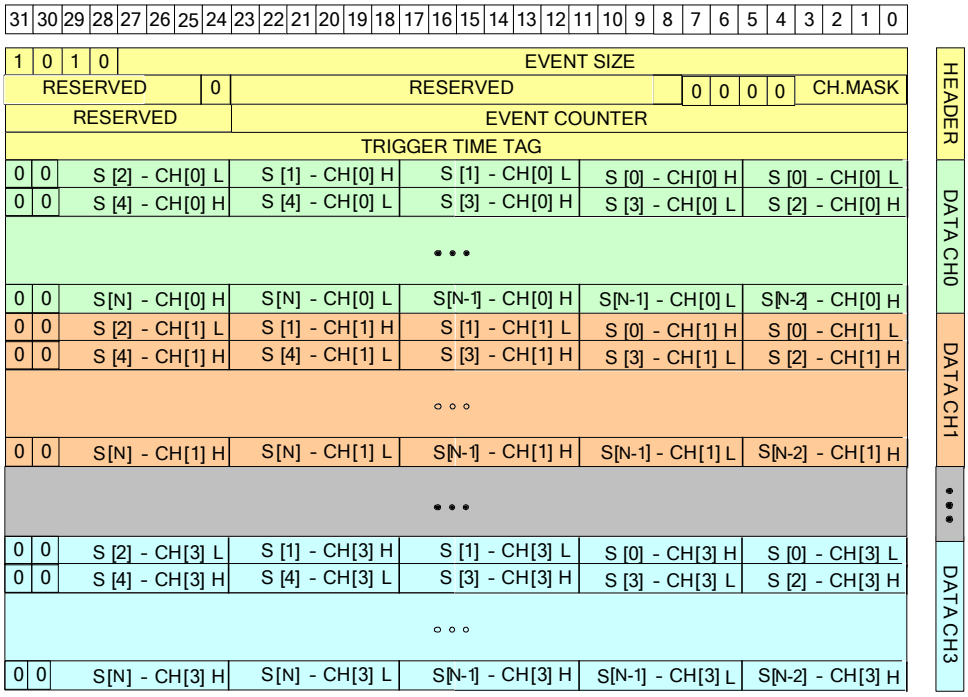



**Fig. 3.5:Event Organization (Pack2.5 mode), normal format**

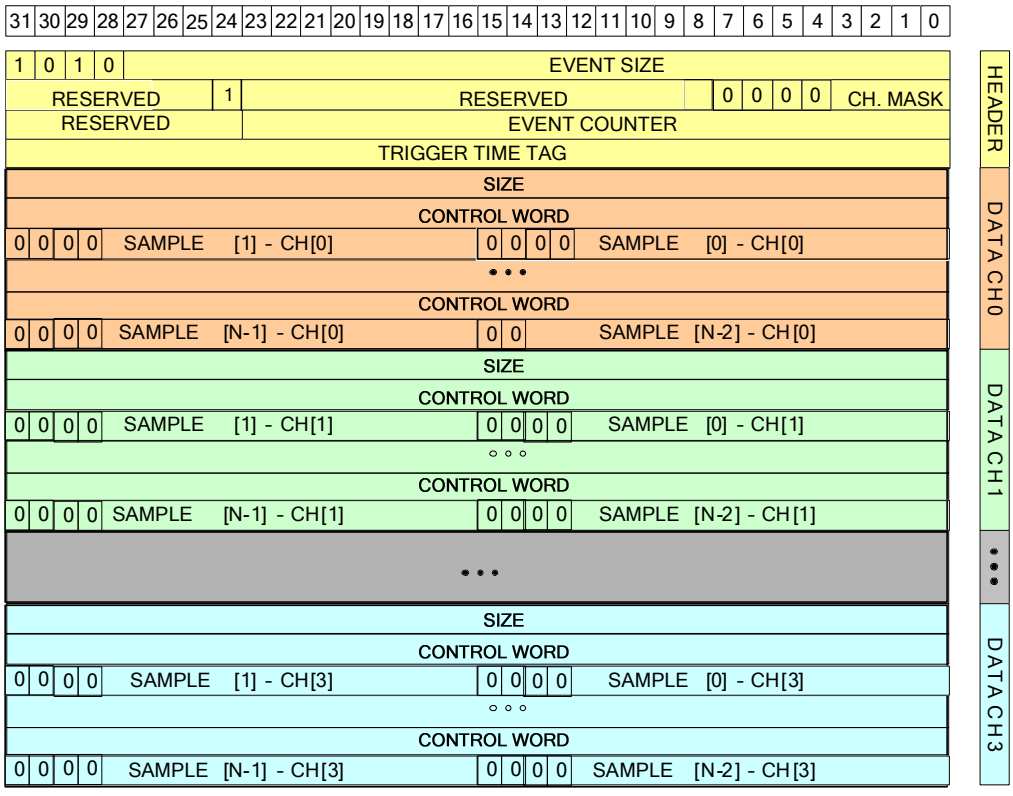



**Fig. 3.6: Event Organization (standard mode), Zero Length Encoding**

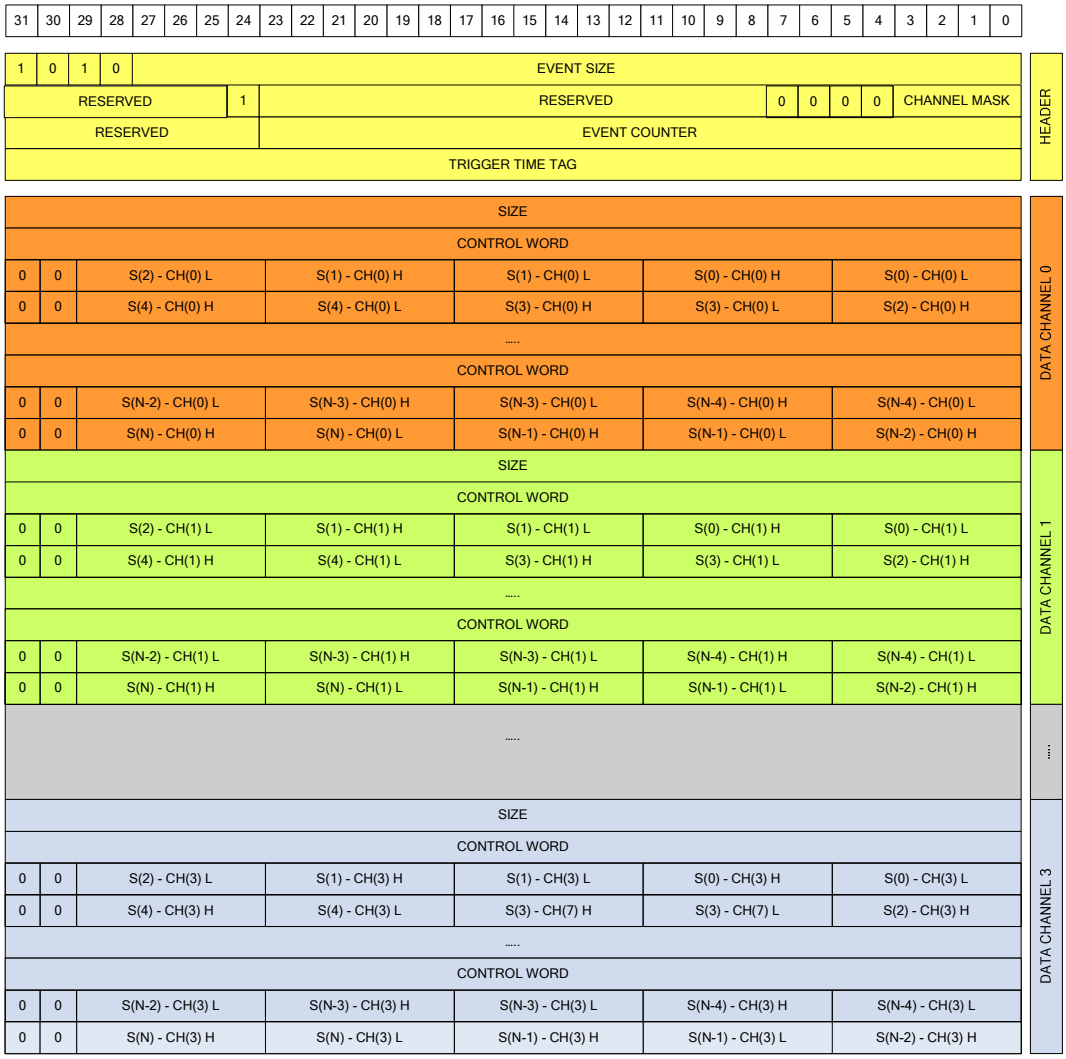



**Fig. 3.7: Event Organization (Pack2.5 mode), Zero Length Encoding**

# 3.4. Zero suppression

The board implements two algorithms of "Zero Suppression" and "Data Reduction"[1]
- Full Suppression based on the signal amplitude (ZS_AMP),
- Zero Length Encoding (ZLE),

The algorithm to be used is selected via Control register, and its configuration takes place via two more registers (CHANNEL n ZS_THRES and CHANNEL n ZS_NSAMP).
When using these algorithms, it must be noticed that that one datum (64bit long word) contains 4 samples (5 samples with Pack2.5 mode): therefore, depending also on trigger polarity (settings of bit31 of Channel n ZS_THRES register), threshold is crossed if:

- Positive Logic: one datum is considered **OVER** threshold if at least one sample is higher or equal to threshold.
- Negative Logic: one datum is considered **UNDER** threshold if at least one sample is lower than threshold.

## *3.4.1. Zero Suppression Algorithm*

### 3.4.1.1. Full Suppression based on the amplitude of the signal

Full Suppression based on the signal amplitude allows to discard a full event if the signal does not exceed the programmed threshold for Ns subsequent data at least (Ns is programmable, see § 5.4).
It is also possible to configure the algorithm with "negative" logic: in this case the event is discarded if the signal does not remain under the programmed threshold for Ns subsequent data at least.

### 3.4.1.2. Zero Length Encoding ZLE

Zero Length Encoding allows to transfer the event in compressed mode, discarding either the data under the threshold set by the User (positive logic) or the data over the threshold set by the User (negative logic).

With Zero Length Encoding it is also possible to set $N_{LBK}$ (LOOK BACK), the number of data to be stored before the signal crosses the threshold and/or, $N_{LFWD}$ (LOOK FORWARD), the number of data to be stored after the signal crosses the threshold (see § 5.3).
In this case the event of each channel has a particular format which allows the construction of the acquired time interval:

- **Total size of the event (total number of transferred 32bit data words)**
- **Control word**
- [stored valid data, if control word is "good"]
- **Control word**
- [stored valid data, if control word is "good"]
- ...

The total size is the number of 32 bit data that compose the event (including the size itself).
The control word has the following format:

[1] Available with Piggy Back Rev. 0.5 and Firmware Rev. 0.5

| Bit | Function |
|---|---|
| [31] | 0: skip<br>1: good |
| [30] | 0: AMC FPGA FW rev 0.5 and earlier<br>1: AMC FPGA FW rev 0.6 and later |
| [29:21] | 0 |
| [20:0] | stored/skipped words |

If the control word type is "good", then it will be followed by as many 32bit data words as those indicated in the "stored/skipped words" field; if the control word type is "skip" then it will be followed by a "good" control world, unless the end of event is reached.
**IMPORTANT NOTE: the maximum allowed number of control words is 62 (14 for AMC FPGA release 0.5 and earlier); therefore the ZLE is active within the event until the 14$^{th}$ transition between a "good" and a "skip" zone (or between a "skip" and a "good" zone). All the subsequent samples are considered "good" and stored.**

### *3.4.2. Zero Suppression Examples*

If the input signal is the following ($N_1$ , $N_2$ .. $N_n$, $N_{LBK}$ , $N_{LFWD}$ are 64bit longwords = 4 samples each[2]):

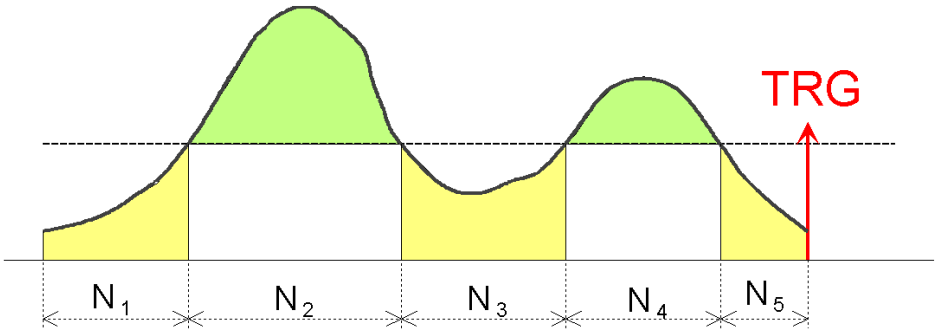



**Fig. 3.8: Zero Suppression example**

If the algorithm works in positive logic, and
**$N_{LBK} < N_1$** ;
**$N_{LFWD} < N_5$** ;
**$N_{LBK} + N_{LFWD} < N_3$** ;

[2] 5 samples each with Pack2.5 mode

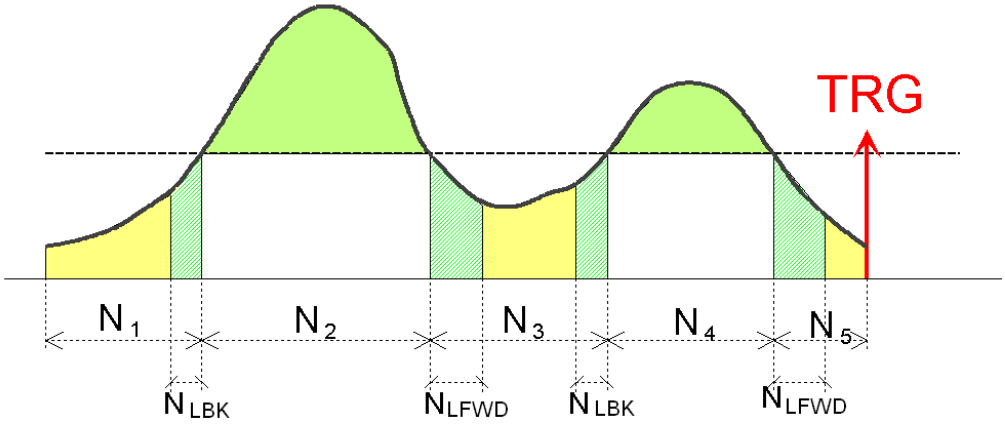



**Fig. 3.9: Example with positive logic and non-overlapping $N_{LBK}$ / $N_{LFWD}$**

then the readout event is:
$N'_2 + N'_4 + 5$ (control words) + 1 (size)
Skip $N'_1 = 2(N_1 - N_{LBK})$
Good $N'_2 = 2(N_{LBK} + N_2 + N_{LFWD})$
... $N'_2$ words with samples over threshold
Skip $N'_3 = 2(N_3 - N_{LFWD} - N_{LBK})$
Good $N'_4 = 2(N_{LBK} + N_4 + N_{LFWD})$
... $N'_4$ words with samples over threshold
Skip $N'_5 = 2(N_5 - N_{LFWD})$

If the algorithm works in negative logic, and
**$N_{LBK} + N_{LFWD} < N_2$** ;
**$N_{LBK} + N_{LFWD} < N_4$** ;

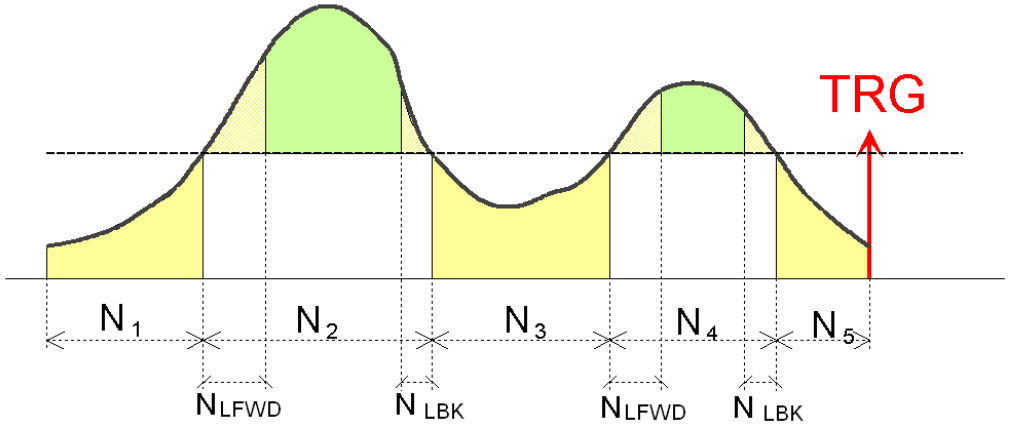



**Fig. 3.10: Example with negative logic and non-overlapping $N_{LBK}$ / $N_{LFWD}$**

then the readout event is:
$N'_1 + N'_3 + N'_5 + 5$ (control words) + 1 (size)
Good $N'_1 = 2(N_1 + N_{LFWD})$
... $N'_1$ words with samples under threshold
Skip $N'_2 = 2(N_2 - N_{LFWD} - N_{LBK})$
Good $N'_3 = 2(N_{LBK} + N_3 + N_{LFWD})$
... $N'_3$ words with samples under threshold

Skip $N'_4=2(N_4 - N_{LFWD} - N_{LBK})$
Good $N'_5 = 2(N_{LBK} + N_5)$
... $N'_5$ words with samples under threshold

In some cases the number of data to be discarded can be smaller than $\mathbf{N_{LBK}}$ and $\mathbf{N_{LFWD}}$ :

**1)** If the algorithm works in positive logic, and
$\mathbf{N_1 \leq N_{LBK} < N_3}$ ;
$\mathbf{N_{LFWD} = 0}$ ;

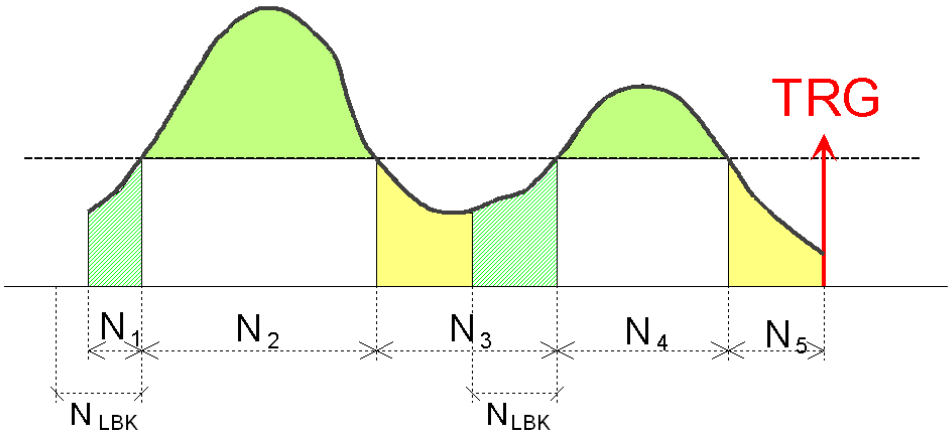



**Fig. 3.11: Example with positive logic and non overlapping $N_{LBK}$**

then the readout event is:
$N'_1 + N'_2 + N'_4 + 5$ (control words) + 1 (size)
Good $N'_1 + N'_2 = 2(N_1 + N_2)$
... $N'_1 + N'_2$ words with samples over threshold
Skip $N'_3 = 2(N_3 - N_{LBK})$
Good $N'_4 = 2(N_{LBK} + N_4)$
... $N'_4$ words with samples over threshold
Skip $N'_5 = 2N_5$

**2)** If the algorithm works in positive logic, and
$\mathbf{N_{LBK} = 0}$ ;
$\mathbf{N_5 \leq N_{LFWD} < N_3}$ ;

then the readout event is:
$N'_2 + N'_4 + N'_5 + 5$ (control words) + 1 (size)
Skip $N'_1 = 2N_1$
Good $N'_2 = 2(N_2 + N_{LFWD})$
... $N'_2$ words with samples over threshold
Skip $N'_3 = 2(N_3 - N_{LFWD})$
Good $N'_4 + N'_5$ ($N'_5 = 2N_5$...)
... $N'_4 + N'_5$ words with samples over threshold

**3)** If the algorithm works in positive logic, and
$\mathbf{N_{LBK} = 0}$ ;
$\mathbf{N_3 \leq N_{LFWD} < N_5}$ ;

then the readout event is:
$N'_2 + 3$ (control words) + 1 (size)

Skip $N'_1 = 2N_1$
Good $N'_2 = 2(N_2 + N_3 + N_4 + N_{LFWD})$
... $N'_2$ words with samples over threshold
Skip $N'_5 = 2(N_5 - N_{LFWD})$

**4)** If the algorithm works in positive logic, and
$\mathbf{N_3 \leq N_{LBK} < N_1}$;
$\mathbf{N_{LFWD} = 0}$;

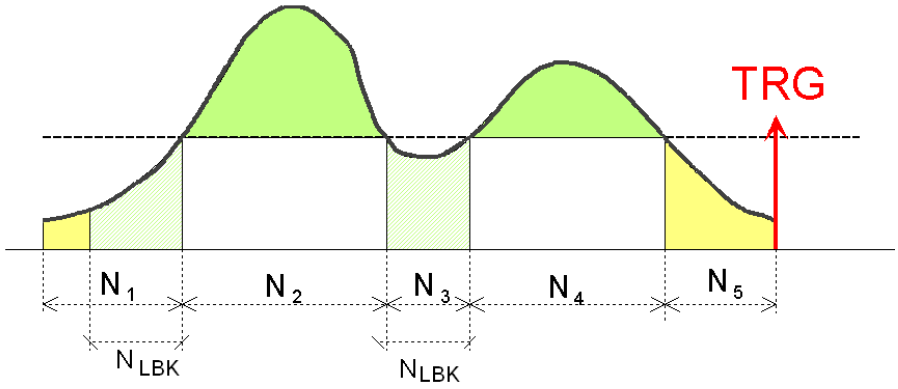



**Fig. 3.12: Example with positive logic and overlapping $N_{LBK}$**

then the readout event is:
$N'_2 + N'_4 + 4$ (control words) + 1 (size)
Skip $N'_1 = 2(N_1 - N_{LBK})$
Good $N'_2 = 2(N_{LBK} + N_2)$
... $N'_2$ words with samples over threshold
Good $N'_4 = 2(N_3 + N_4)$
... $N'_4$ words with samples over threshold
Skip $N'_5 = 2N_5$
N.B: In this case there are two subsequent “GOOD” intervals.

**5)** If the algorithm works in positive logic, and
$\mathbf{0 < N_{LBK} < N_1}$;
$\mathbf{N_{LFWD} < N_5}$;
$\mathbf{N_{LBK} + N_{LFWD} \geq N_3}$.
then the readout event is:
$N'_2 + N'_4 + 4$ (control words) + 1 (size)
Skip $N'_1 = 2(N_1 - N_{LBK})$
Good $N'_2 = 2(N_{LBK} + N_2 + N_{LFWD})$
... $N'_2$ words with samples over threshold
Good $N'_4 = 2(N_3 - N_{LFWD}) + 2N_4 + 2N_{LFWD}$
... $N'_4$ words with samples over threshold
Skip $N'_5 = 2(N_5 - N_{LFWD})$
N.B: In this case there are two subsequent “GOOD” intervals.

These examples are reported with positive logic; the compression algorithm is the same also working in negative logic.

## 3.5. Trigger management

All the channels in a board share the same trigger: this means that all the channels store an event at the same time and in the same way (same number of samples and same position with respect to the trigger); several trigger sources are available.

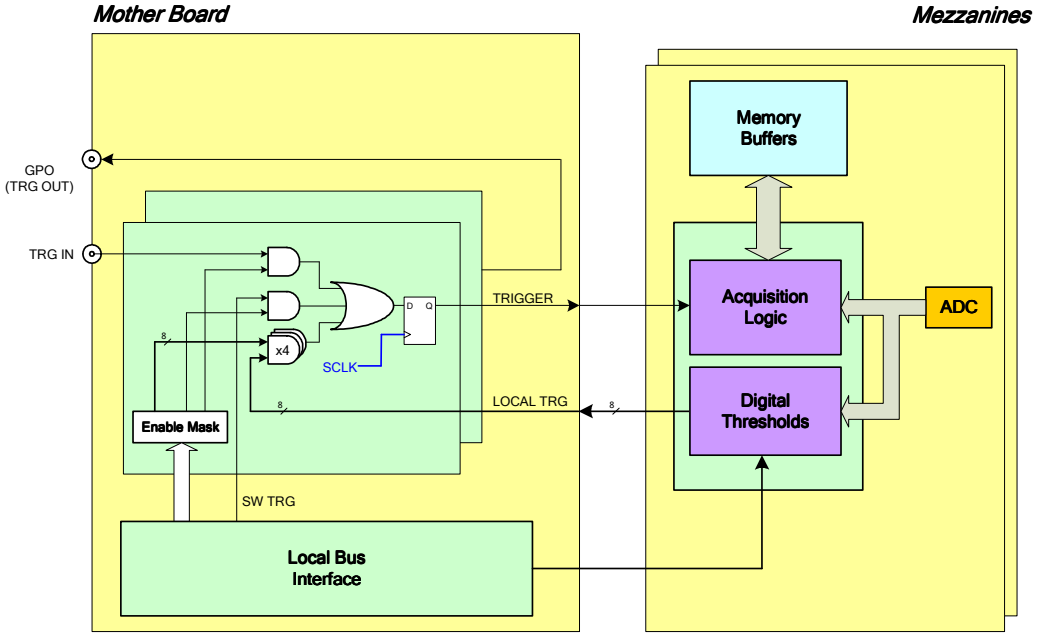



**Fig. 3.13: Block diagram of Trigger management**

### 3.5.1. External trigger

External trigger can be NIM/TTL signal on LEMO front panel connector, 50 Ohm impedance. The external trigger is synchronised with the internal clock (see § 3.2.1); if External trigger is not synchronised with the internal clock, a one clock period jitter occurs.

### 3.5.2. Software trigger

Software trigger are generated INTERNALLY (write access in the relevant register, see § 5.19).

### 3.5.3. Local channel auto-trigger

Each channel can generate a local trigger as the digitised signal exceeds the Vth threshold (ramping up or down, depending on register settings), and remains under or over threshold for Nth “4/5 samples groups” (depending on selected storage mode, see § 3.3.3) at least (Nth is programmable via register setting). The Vth digital threshold, the edge type, and the minimum number Nth of [4/5 samples] are programmable via register accesses, see § 5.3 and § 5.6; actually local trigger is delayed of Nth [4/5 samples] with respect to the input signal.

N.B.: the local trigger signal does not start directly the event acquisition on the relevant channel; such signal is propagated to the central logic which produces the global trigger, which is distributed to all channels (see § 3.5.4).

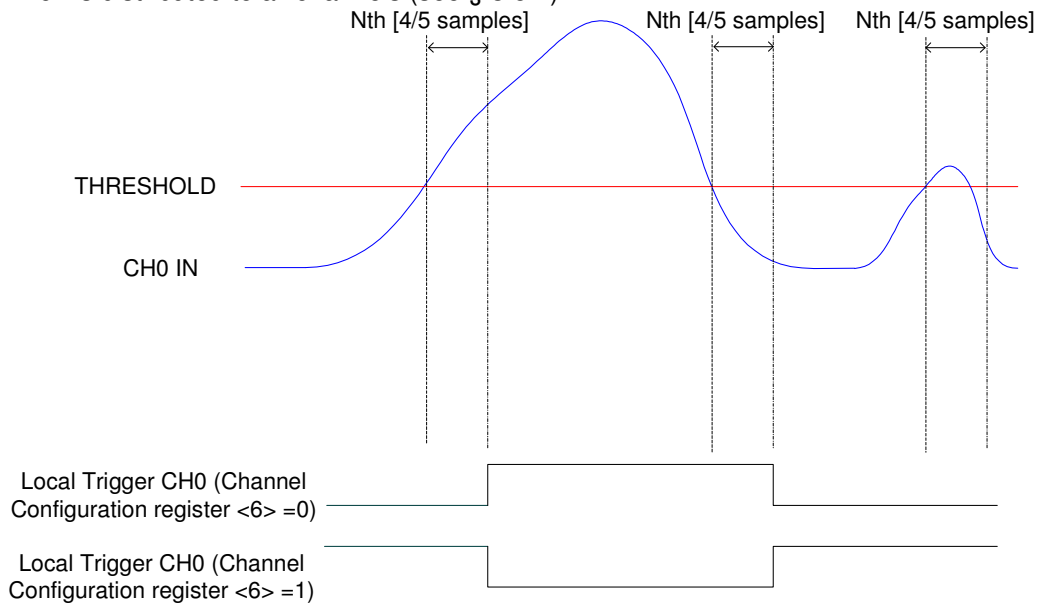



**Fig. 3.14: Local trigger generation**

### 3.5.3.1. Trigger coincidence level

It is possible to set the minimum number of channels that must be over threshold, beyond the triggering channel, in order to actually generate the local trigger signal. If, for example, Trigger Source Enable Mask (see § 5.20) bits[3:0]=F (all channels enabled) and Local trigger coincidence level = 1 (bits [26:24]), whenever an enabled channel exceeds the threshold, the trigger will be generated only if at least another channel is over threshold at that moment. Local trigger coincidence level must be smaller than the number of channels enabled via bit[3:0] mask. The following figure shows examples with Local trigger coincidence level = 1 and = 0.

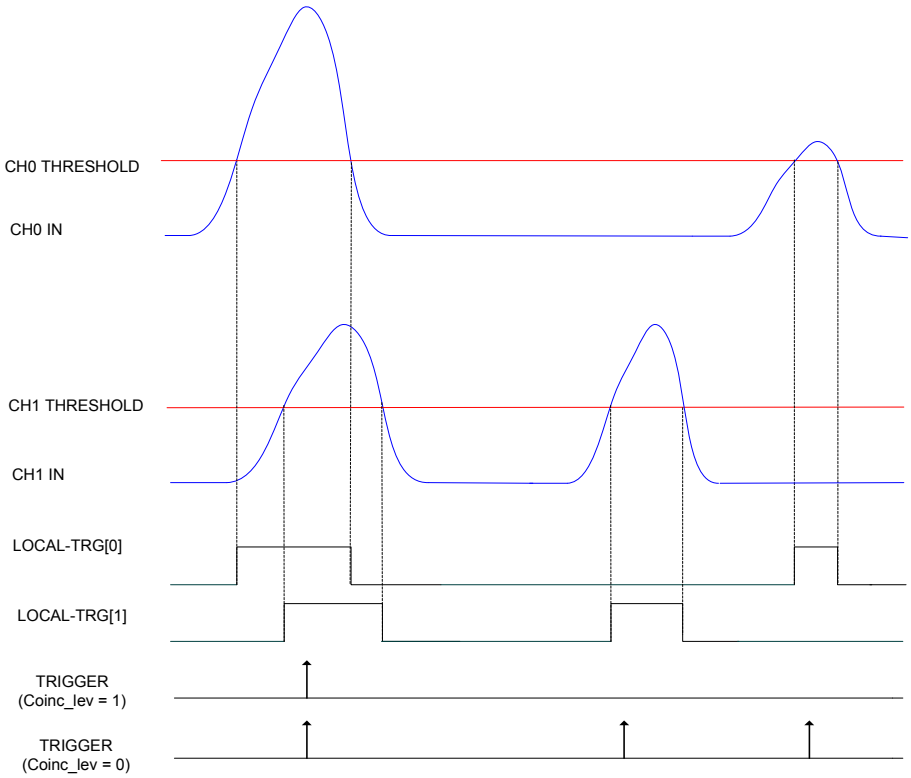



**Fig. 3.15: Local trigger relationship with Coincidence level**

### *3.5.4. Trigger distribution*

The OR of all the enabled trigger sources, after being synchronised with the internal clock, becomes the global trigger of the board and is fed in parallel to all the channels, which store an event.
A Trigger Out is also generated on the relevant front panel GPO connector (NIM or TTL), and allows to extend the trigger signal to other boards.
For example, in order to start the acquisition on all the channels in the crate, as one of the channels ramps over threshold, the Local Trigger must be enabled as Trigger Out, the Trigger Out must then be fed to a Fan Out unit; the obtained signal has to be fed to the External Trigger Input of all the boards in the crate (including the board which generated the Trigger Out signal).

## 3.6. Data transfer capabilities

The board can be accessed by using software drivers and libraries developed by CAEN. Single 16/32 register read/write cycles, multi read cycles and block transfers are supported by the provided library (please consult the relevant documentation for details)
Sustained readout rate is up to 60 MB/s for optical link, using block transfers, and up to 30 MB/s for a USB 2.0 link, using block transfers as well.

## 3.7. Events readout

Event readout is done by accessing the Event Readout Buffer (see § 5.1), a FIFO (First-In First-Out) memory that can be accessed into the 0x0000-0x0FFC address space.
Data transfer is always aligned to the programmed number N of events; let X the size of the event expected or read from dedicated register:

- If the event size is known, a read cycle equal to N*X will return all data without interruptions.
- If the number of data read from the Event Readout Buffer is higher than N*X, transfer will be terminated anyway by DT5720 at the end of N*X data.
- If the event size X is unknown (for example in case of overlapping triggers), there are two cases:
  - data transfer ≤ N*X : all data will be returned.
  - data transfer > N*X : only N*X data will be returned.

Once an event is read, the corrisponding acquisition buffers are available to store new data.
During readout, the board can continue to store events in memory up to the maximum number of programmed buffers available; the acquisition process is therefore "dead-timeless": event storage is only interrupted if the combination of trigger and readout rate causes a memory full situation: all acquisition buffers are used and they have not been read yet.
In order to exploit the maximum readout rate allowed by the communication path (USB or optical link), it is suggested to perform block transfer read cycles of at least N*X data with N set to its maximum value, whether possible.

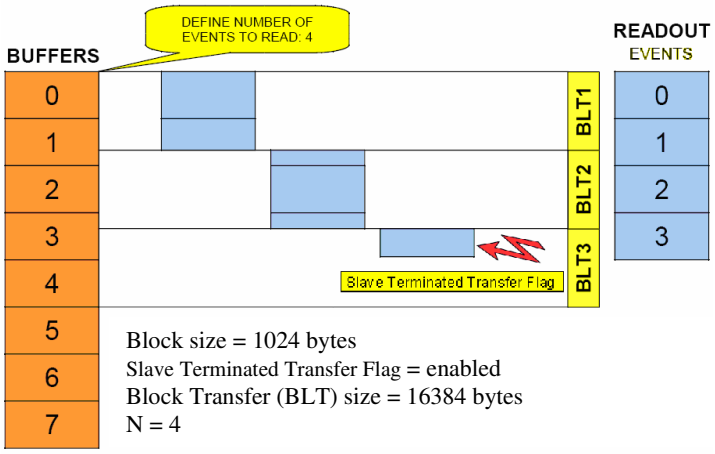



**Fig. 3.16: Example of block transfer readout**

## 3.8. Optical Link and USB access

The board houses a USB2.0 compliant port, providing a transfer rate up to 30 MB/s, and a daisy chainable Optical Link able to transfer data at 80 MB/s; the latter allows to connect up to eight DT5720 to a single Optical Link Controller: for more information, see www.caen.it (path: Products / Front End / PCI/PCIe / Optical Controller)
The parameters for read/write accesses via optical link are Address Modifier, Base Address, data Width, etc; wrong parameter settings cause Bus Error.

Control Register bit 3 allows to enable the module to broadcast an interrupt request on the Optical Link; the enabled Optical Link Controllers propagate the interrupt on the PCI bus as a request from the Optical Link is sensed.

# 4. Software tools

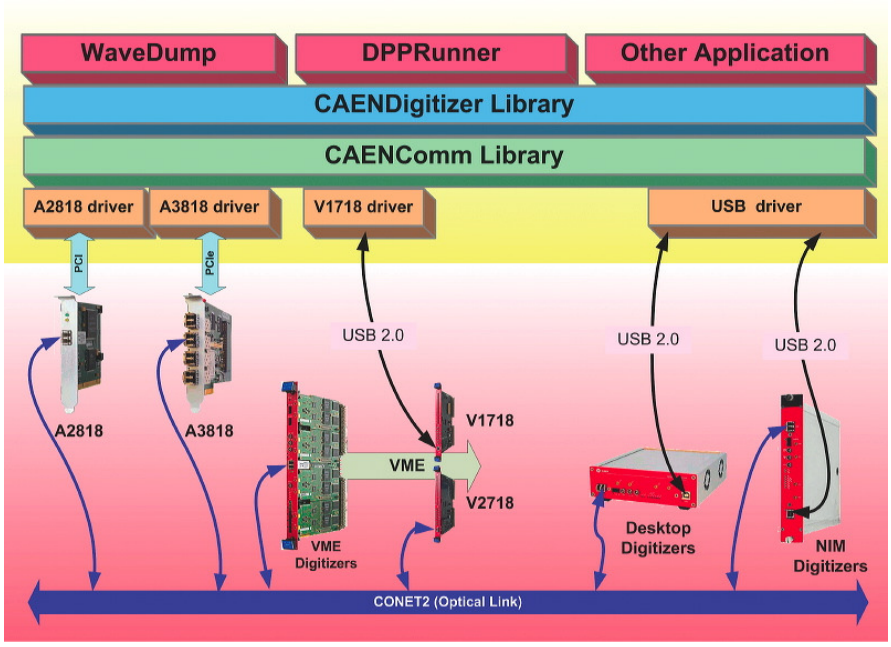



**Fig. 4.1: Block diagram of the software layers**

CAEN provides drivers for both the physical communication channels (USB and the proprietary CONET Optical Link managed by the A2818 PCI card or A3818 PCIe cards; see § 6.4), a set of C and LabView libraries, demo applications and utilities. Windows and Linux are both supported. The available software is the following:

- **CAENComm** library contains the basic functions for access to hardware; the aim of this library is to provide a unique interface to the higher layers regardless the type of physical communication channel. The CAENComm requires the CAENVMELib library to be installed even in the cases where the VME is not used.
- **CAENDigitizer** is a library of functions designed specifically for the digitizer family and it supports also the boards running special DPP (Digital Pulse Processing) firmware. The purpose of this library is to allow the user to open the digitizer, program it and manage the data acquisition in an easy way: with few lines of code the user can make a simple readout program without the necessity to know the details of the registers and the event data format. The CAENDigitizer library implements a common interface to the higher software layers, masking the details of the physical channel and its protocol, thus making the libraries and applications that rely on the CAENDigitizer independent from the physical layer. The library is based on the CAENComm library that manages the communication at low level (read and write access). CAENVMELib and CAENComm libraries must be already installed on the host PC before installing the CAENDigitizer; however, both CAENVMELib and CAENComm libraries are completely transparent to the user.
- **WaveDump** is a Console application that allows to program the digitizer (according to a text configuration file that contains a list of parameters and instructions), to start the acquisition, read the data, display the readout and trigger rate, apply some post processing (such as FFT and amplitude histogram), save data to a file and also plot the waveforms using the external plotting tool “gnuplot”, available on internet for free. This program is quite basic and has no graphics but it is an excellent example of C

code that demonstrates the use of libraries and methods for an efficient readout and data analysis. **NOTE:** WaveDump does not work with digitizers running DPP firmware. The users who intend to write the software on their own are suggested to start with this demo and modify it according to their needs. For more details please see the WaveDump User Manual and Quick Start Guide (Doc nr.: UM2091, GD2084 ).

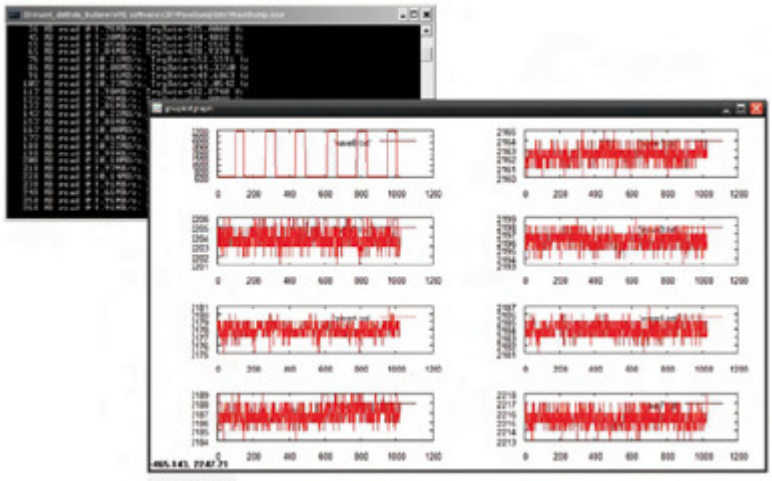


**Fig. 4.2: WaveDump output waveforms**

- **CAENScope** is a fully graphical program that implements a simple oscilloscope: it allows to see the waveforms, set the trigger thresholds, change the scales of time and amplitude, perform simple mathematical operations between the channels, save data to file and other operations. CAENscope is provided as an executable file; the source codes are not distributed. **NOTE:** CAENScope does not work with digitizers running DPP firmware and it is not compliant with x742 digitizer family. For more details please see the CAENScope Quick Start Guide GD2484.

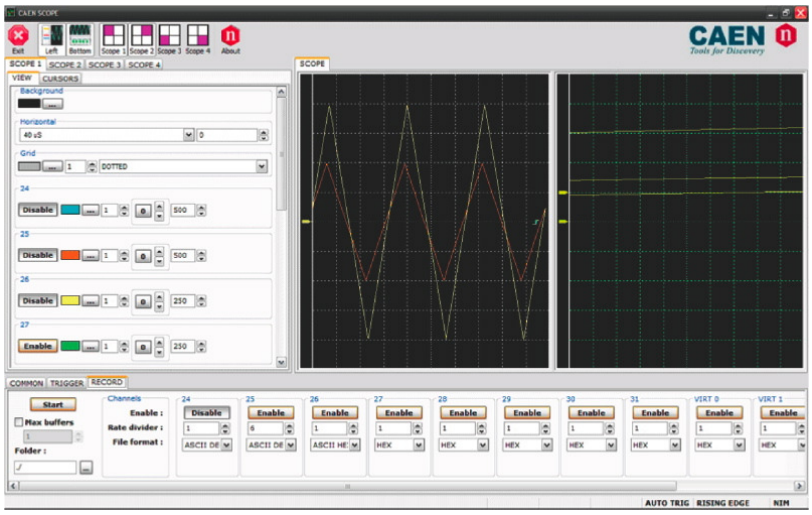



**Fig. 4.3: CAENScope oscilloscope tab**

- **CAENUpgrader** is a software composed of command line tools together with a Java Graphical User Interface (for Windows and Linux OS). CAENUpgrader allows in few easy steps to upload different firmware versions on CAEN boards, to upgrade the VME digitizers PLL, to get board information and to manage the firmware license. CAENUpgrader requires the installation of 2 CAEN libraries (CAENComm,

CAENVMELib) and Java SE6 (or later). CAENComm allows CAENUpgrader to access target boards via USB or via CAEN proprietary CONET optical link.

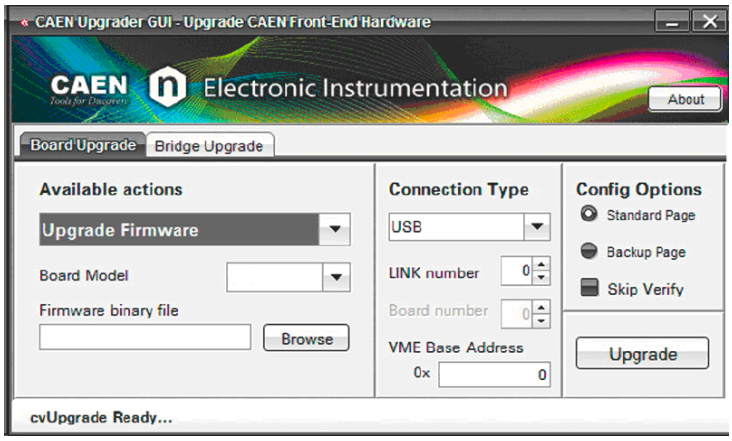



**Fig. 4.4: CAENUpgrader Graphical User Interface**

- **DPP Control Software** is an application that manages the acquisition in the digitizers which have DPP firmware installed on it. The program is made of different parts: there is a GUI whose purpose is to set all the parameters for the DPP and for the acquisition; the GUI generates a textual configuration file that contains all the parameters. This file is read by the Acquisition Engine (**DPPrunner)**, which is a C console application that programs the digitizer according to the parameters, starts the acquisition and manage the data readout. The data, that can be waveforms, time stamps, energies or other quantities of interest, can be saved to output files or plotted using gnuplot as an external plotting tool, exactly like in WaveDump. NOTE: so far DPP Control Software is developed for Mod. x724 and Mod. x720 digitizer series.

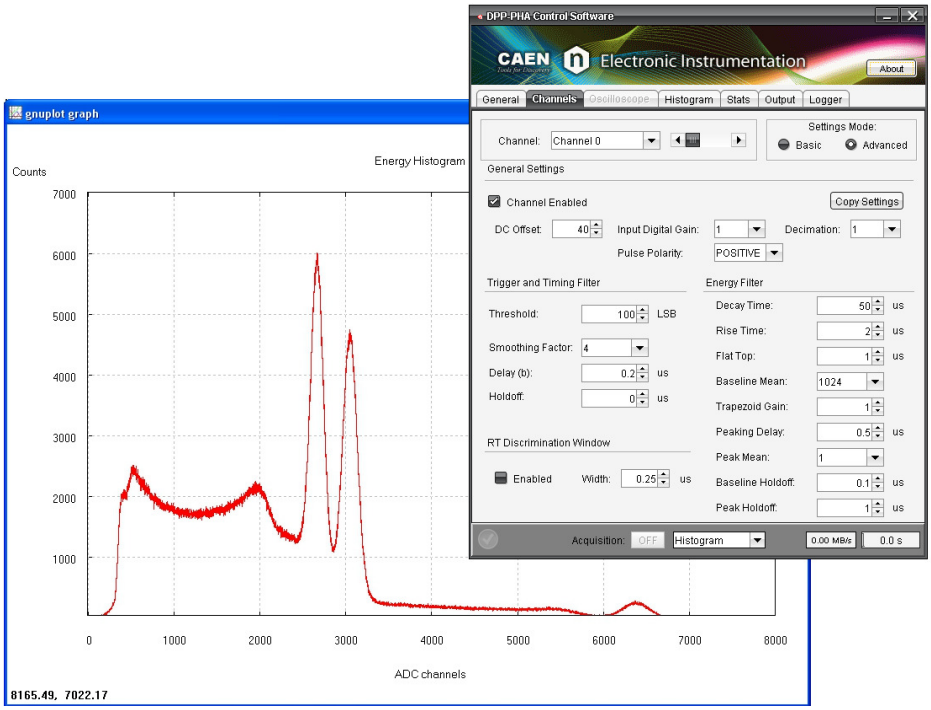



**Fig. 4.5: DPP Control Software Graphical User Interface and Energy plot**

# 5. Board internal registers

The following sections will describe in detail the registers (accessible via software in D32 mode) content.

## 5.1. Registers address map

**Table 5.1: Address Map for the Model DT5720**

| REGISTER NAME | ADDRESS | MODE | H_RES | S_RES | CLR |
|---|---|---|---|---|---|
| EVENT READOUT BUFFER | 0x0000-0x0FFC | R | X | X | X |
| Channel n ZS_THRES | 0x1n24 | R/W | X | X | |
| Channel n ZS_NSAMP | 0x1n28 | R/W | X | X | |
| Channel n THRESHOLD | 0x1n80 | R/W | X | X | |
| Channel n TIME OVER/UNDER THRESHOLD | 0x1n84 | R/W | X | X | |
| Channel n STATUS | 0x1n88 | R | X | X | |
| Channel n AMC FPGA FIRMWARE REVISION | 0x1n8C | R | | | |
| Channel n BUFFER OCCUPANCY | 0x1n94 | R | X | X | X |
| Channel n DAC | 0x1n98 | R/W | X | X | |
| Channel n ADC CONFIGURATION | 0x1n9C | R/W | X | X | |
| CHANNEL CONFIGURATION | 0x8000 | R/W | X | X | |
| CHANNEL CONFIGURATION BIT SET | 0x8004 | W | X | X | |
| CHANNEL CONFIGURATION BIT CLEAR | 0x8008 | W | X | X | |
| BUFFER ORGANIZATION | 0x800C | R/W | X | X | |
| CUSTOM SIZE | 0x8020 | R/W | X | X | |
| ACQUISITION CONTROL | 0x8100 | R/W | X | X | |
| ACQUISITION STATUS | 0x8104 | R | | | |
| SW TRIGGER | 0x8108 | W | | | |
| TRIGGER SOURCE ENABLE MASK | 0x810C | R/W | X | X | |
| FRONT PANEL TRIGGER OUT ENABLE MASK | 0x8110 | R/W | X | X | |
| POST TRIGGER SETTING | 0x8114 | R/W | X | X | |
| FRONT PANEL I/O CONTROL | 0x811C | R/W | X | X | |
| CHANNEL ENABLE MASK | 0x8120 | R/W | X | X | |
| ROC FPGA FIRMWARE REVISION | 0x8124 | R | | | |
| EVENT STORED | 0x812C | R | X | X | X |
| BOARD INFO | 0x8140 | R | | | |
| EVENT SIZE | 0x814C | R | X | X | X |
| CONTROL | 0xEF00 | R/W | X | | |
| STATUS | 0xEF04 | R | | | |
| INTERRUPT STATUS ID | 0xEF14 | R/W | X | | |
| INTERRUPT EVENT NUMBER | 0xEF18 | R/W | X | X | |
| BLT EVENT NUMBER | 0xEF1C | R/W | X | X | |
| SCRATCH | 0xEF20 | R/W | X | X | |
| SW RESET | 0xEF24 | W | | | |
| SW CLEAR | 0xEF28 | W | | | |
| FLASH ENABLE | 0xEF2C | R/W | X | | |

| REGISTER NAME | ADDRESS | MODE | H_RES | S_RES | CLR |
|---|---|---|---|---|---|
| FLASH DATA | 0xEF30 | R/W | X | | |
| CONFIGURATION RELOAD | 0xEF34 | W | | | |
| CONFIGURATION ROM | 0xF000-0xF088 | R | | | |

## 5.2. Configuration ROM (0xF000-0xF088; r)

The following registers contain some module's information, they are D32 accessible (read only):

- **OUI**: manufacturer identifier (IEEE OUI)
- **Version**: purchased version
- **Board ID**: Board identifier
- **Revision**: hardware revision identifier
- **Serial MSB**: serial number (MSB)
- **Serial LSB**: serial number (LSB)

**Table 5.2: ROM Address Map for the Model DT5720**

| Description | Address | Content |
|---|---|---|
| checksum | 0xF000 | *0xA4* |
| checksum_length2 | 0xF004 | 0x00 |
| checksum_length1 | 0xF008 | 0x00 |
| checksum_length0 | 0xF00C | 0x20 |
| constant2 | 0xF010 | 0x83 |
| constant1 | 0xF014 | 0x84 |
| constant0 | 0xF018 | 0x01 |
| c_code | 0xF01C | 0x43 |
| r_code | 0xF020 | 0x52 |
| oui2 | 0xF024 | 0x00 |
| oui1 | 0xF028 | 0x40 |
| oui0 | 0xF02C | 0xE6 |
| vers | 0xF030 | 0x30 |
| board2 | 0xF034 | 0x02 |
| board1 | 0xF038 | 0x16 |
| board0 | 0xF03C | 0x58 |
| revis3 | 0xF040 | *0x00* |
| revis2 | 0xF044 | *0x00* |
| revis1 | 0xF048 | *0x00* |
| revis0 | 0xF04C | *0x00* |
| sernum1 | 0xF080 | |
| sernum0<br>VCXO type | 0xF084<br>0xF088 | <br>0x00 (AD9520-3) |

These data are written into one Flash page; at Power ON the Flash content is loaded into the Configuration RAM, where it is available for readout.

## 5.3. Channel n ZS_THRES (0x1n24; r/w)

| Bit | Function |
|---|---|
| [31] | 0 = Positive Logic<br>1 = Negative Logic |
| [30:12] | *reserved* |
| [11:0] | With "Full Suppression based on the amplitude", the 12 LSB represent the value to be compared with each sample of the event; and see if it is over/unedr threshold (depending on the used logic).<br>With "Zero Length Encoding", the 12 LSB represent the value to be compared with each sample of the event, and see if it is "good" or "skip" type (see § 3.4 and § 5.12). |

## 5.4. Channel n ZS_NSAMP (0x1n28; r/w)

| Bit | Function |
|---|---|
| [31:0] | With "Full Suppression based on the amplitude" (ZS AMP), bits [20:0] allow to set the number Ns of subsequent data which must be found over/under threshold (depending on the used logic) necessary to validate the event; if this field is set to 0, it is considered "1".<br>With "Zero length encoding" (ZLE) bit [31:16] allows to set/read $N_{LBK}$: the number of data to be stored before the signal crosses the threshold.<br>bit [15:0] allows to set/read $N_{LFWD}$: the number of data to be stored after the signal crosses the threshold (see § 3.4 and § 5.12) |

## 5.5. Channel n Threshold (0x1n80; r/w)

| Bit | Function |
|---|---|
| [11:0] | Threshold Value for Trigger Generation |

Each channel can generate a local trigger as the digitised signal exceeds the Vth threshold, and remains under or over threshold for Nth [4 samples; 5 samples in Pack2.5 mode] at least; local trigger is delayed of Nth [4/5 samples] with respect to input signal. This register allows to set Vth (LSB=input range/12bit); see also § 3.5.3.

## 5.6. Channel n Over/Under Threshold (0x1n84; r/w)

| Bit | Function |
|---|---|
| [11:0] | Number of Data under/over Threshold |

Each channel can generate a local trigger as the digitised signal exceeds the Vth threshold, and remains under or over threshold for Nth [4/5 samples] at least; local trigger is delayed of Nth [4 samples; 5 samples in Pack2.5 mode] with respect to input signal. This register allows to set Nth; see also § 3.5.3.

## 5.7. Channel n Status (0x1n88; r)

| Bit | Function |
|---|---|
| [5:3] | *reserved* |
| [2] | Channel n DAC (see § 5.10) Busy<br>1 = Busy<br>0 = DC offset updated |
| [1] | Memory empty |
| [0] | Memory full |

## 5.8. Channel n AMC FPGA Firmware (0x1n8C; r)

| Bit | Function |
|---|---|
| [31:16] | Revision date in Y/M/DD format |
| [15:8] | Firmware Revision (X) |
| [7:0] | Firmware Revision (Y) |

Bits [31:16] contain the Revision date in Y/M/DD format.
Bits [15:0] contain the firmware revision number coded on 16 bit (X.Y format).
Example: revision 1.3 of 12th June 2007 is: 0x7612103

## 5.9. Channel n Buffer Occupancy (0x1n94; r)

| Bit | Function |
|---|---|
| [10:0] | Occupied buffers (0..1024) |

## 5.10. Channel n DAC (0x1n98; r/w)

| Bit | Function |
|---|---|
| [15:0] | DAC Data |

Bits [15:0] allow to define a DC offset to be added the input signal in the ±1V range. When Channel n Status bit 2 is set to 0, DC offset is updated (see § 5.7).

## 5.11. Channel n ADC Configuration (0x1n9C; r/w)

| Bit | Function |
|---|---|
| [15:0] | T.B.D. |

This register allows to pilot the relevant ADC signals. See the LTC2242-12 - 12-Bit, 250Msps ADC data sheet for details.

## 5.12. Channel Configuration (0x8000; r/w)

| Bit | Function |
|---|---|
| [19:16] | Allows to select Zero Suppression algorithm:<br>0000 = no zero suppression (default);<br>0010 = zero length encoding (ZLE);<br>0011 = full suppression based on the amplitude (ZS AMP) |
| [18:12] | *reserved* |
| [11] | 0 = Pack2.5 disabled<br>1 = Pack2.5 enabled |
| [10:7] | *reserved* |
| [6] | 0 = Trigger Output on Input Over Threshold<br>1 = Trigger Output on Input Under Threshold<br>allows to generate local trigger either on channel over or under threshold (see § 5.3 and § 5.6) |
| [4] | Reserved (Must be always set to 1) |
| [3] | 0 = Test Pattern Generation Disabled<br>1 = Test Pattern Generation Enabled |
| [1] | 0 = Trigger Overlapping Not Enabled<br>1 = Trigger Overlapping Enabled<br>Allows to handle trigger overlap (see § 3.3.2) |
| [0] | *reserved* |

This register allows to perform settings which apply to all channels.

It is possible to perform selective set/clear of the Channel Configuration register bits writing to 1 the corresponding set and clear bit at address 0x8004 (set) or 0x8008 (clear) see the following § 5.13 and 5.14. Default value is 0x10.

## 5.13. Channel Configuration Bit Set (0x8004; w)

| Bit | Function |
|---|---|
| [7:0] | Bits set to 1 means that the corresponding bits in the Channel Configuration register are set to 1. |

## 5.14. Channel Configuration Bit Clear (0x8008; w)

| Bit | Function |
|---|---|
| [7:0] | Bits set to 1 means that the corresponding bits in the Channel Configuration register are set to 0. |

## 5.15. Buffer Organization (0x800C; r/w)

| Bit | Function |
|---|---|
| [3:0] | BUFFER CODE |

The BUFFER CODE allows to divide the available Output Buffer Memory into a certain number of blocks, according to the table in § 3.3.2.
A write access to this register causes a Software Clear, see § 5.36. This register must not be written while acquisition is running.

## 5.16. Custom Size (0x8020; r/w)

| Bit | Function |
|---|---|
| [31:0] | 0= Custom Size disabled<br>$N_{LOC}$ (≠0) = Number of memory locations per event (1 location = 2 samples or 2 locations = 5 samples when Pack2.5 mode is used, see § 3.3.3 ) |

This register must not be written while acquisition is running.

## 5.17. Acquisition Control (0x8100; r/w)

| Bit | Function |
|---|---|
| [6] | 0 = Internal clock source<br>1 = External clock source |
| [5,4] | Reserved |
| [3] | 0 = COUNT ACCEPTED TRIGGERS<br>1 = COUNT ALL TRIGGERS<br>allows to reject overlapping triggers (see § 3.3.2) |
| [2] | 0 = Acquisition STOP<br>1 = Acquisition RUN<br>allows to RUN/STOP Acquisition |
| [1] | Reserved ( set to 0) |
| [0] | 0 = REGISTER-CONTROLLED RUN MODE<br>1 = GPI CONTROLLED RUN MODE |

Bit [2] allows to Run and Stop data acquisition; when such bit is set to 1 the board enters Run mode and a Memory Reset is automatically performed. When bit [2] is reset to 0 the stored data are kept available for readout. In Stop Mode all triggers are neglected.
Bit [0] descritpion:

0 = REGISTER-CONTROLLED RUN MODE: multiboard synchronisation via GPI front panel signal
- RUN control: start/stop via set/clear of bit[2]
- GATE always active (Continuous Gate Mode)

1 = GPI CONTROLLED RUN MODE: Multiboard synchronisation via GPI front panel signal
- GPI works both as SYNC and RUN_START command
- GATE always active (Continuous Gate Mode)

## 5.18. Acquisition Status (0x8104; r)

| Bit | Function |
|---|---|
| [8] | Board ready for acquisition (PLL and ADCs are synchronised correctly)<br>0 = not ready<br>1 = ready<br>This bit should be checked after software reset to ensure that the board will enter immediatly run mode after RUN mode setting; otherwise a latency between RUN mode setting and Acquisition start might occur. |
| [7] | PLL Status Flag (see § 2.5.1):<br>0 = PLL loss of lock<br>1 = no PLL loss of lock<br>NOTE: flag can be restored to 1 via read access to Status Register (see § 0) |
| [6] | PLL Bypass mode (see § 2.5.1):<br>0 = No bypass mode<br>1 = Bypass mode |
| [5] | Clock source:<br>0 = Internal<br>1 = External |
| [4] | EVENT FULL: it is set to 1 as the maximum nr. of events to be read is reached |
| [3] | EVENT READY: it is set to 1 as at least one event is available to readout |
| [2] | 0 = RUN off<br>1 = RUN on |
| [1:0] | *reserved* |

## 5.19. Software Trigger (0x8108; w)

| Bit | Function |
|---|---|
| [31:0] | A write access to this location generates a trigger via software |

## 5.20. Trigger Source Enable Mask (0x810C; r/w)

| Bit | Function |
|---|---|
| [31] | 0 = Software Trigger Disabled<br>1 = Software Trigger Enabled |
| [30] | 0 = External Trigger Disabled<br>1 = External Trigger Enabled |
| [29:27] | *reserved* |
| [26:24] | Local trigger coincidence level (default = 0) |
| [23:4] | *reserved* |
| [3] | 0 = Channel 3 trigger disabled<br>1 = Channel 3 trigger enabled |

| | |
|---|---|
| [2] | 0 = Channel 2 trigger disabled<br>1 = Channel 2 trigger enabled |
| [1] | 0 = Channel 1 trigger disabled<br>1 = Channel 1 trigger enabled |
| [0] | 0 = Channel 0 trigger disabled<br>1 = Channel 0 trigger enabled |

This register bits[0,3] enable the channels to generate a local trigger as the digitised signal exceeds the Vth threshold (see § 3.5.3). Bit0 enables Ch0 to generate the trigger, bit1 enables Ch1 to generate the trigger and so on.
Bits [26:24] allows to set minimum number of channels that must be over threshold, beyond the triggering channel, in order to actually generate the local trigger signal; for example if bit[3:0]=F (all channels enabled) and Local trigger coincidence level = 1, whenever one channel exceeds the threshold, the trigger will be generated only if at least another channel is over threshold at that moment. Local trigger coincidence level must be smaller than the number of channels enabled via bit[3:0] mask.
EXTERNAL TRIGGER ENABLE (bit30) enables the board to sense TRG-IN signals
SW TRIGGER ENABLE (bit 31) enables the board to sense software trigger (see § 5.19).

## 5.21. Front Panel Trigger Out Enable Mask (0x8110; r/w)

| Bit | Function |
|---|---|
| [31] | 0 = Software Trigger Disabled<br>1 = Software Trigger Enabled |
| [30] | 0 = External Trigger Disabled<br>1 = External Trigger Enabled |
| [29:4] | *reserved* |
| [3] | 0 = Channel 3 trigger disabled<br>1 = Channel 3 trigger enabled |
| [2] | 0 = Channel 2 trigger disabled<br>1 = Channel 2 trigger enabled |
| [1] | 0 = Channel 1 trigger disabled<br>1 = Channel 1 trigger enabled |
| [0] | 0 = Channel 0 trigger disabled<br>1 = Channel 0 trigger enabled |

This register bits[0,3] enable the channels to generate a TRG_OUT front panel signal on GPO output as the digitised signal exceeds the Vth threshold (see § 3.5.3).
Bit0 enables Ch0 to generate the TRG_OUT, bit1 enables Ch1 to generate the TRG_OUT and so on.
EXTERNAL TRIGGER ENABLE (bit30) enables the board to generate the TRG_OUT
SW TRIGGER ENABLE (bit 31) enables the board to generate TRG_OUT (see § 5.19).

## 5.22. Post Trigger Setting (0x8114; r/w)

| Bit | Function |
|---|---|
| [31:0] | Post trigger value |

The register value sets the number of post trigger samples. The number of post trigger samples is :
Npost = PostTriggerValue*4 + ConstantLatency; where:
Npost = number of post trigger samples.
PostTriggerValue = Content of this register.
ConstantLatency = constant number of samples added due to the latency associated to the trigger processing logic in the ROC FPGA; this value is constant, but the exact value may change between different firmware revisions.

## 5.23. Front Panel I/O Control (0x811C; r/w)

| Bit | Function |
|---|---|
| [15:2] | *reserved* |
| [1] | 0= panel output signals (GPO) enabled<br>1= panel output signals (GPO) enabled in high impedance |
| [0] | 0 = GPI/GPO/TRG-IN are NIM I/O Levels<br>1 = GPI/GPO/TRG-IN are TTL I/O Levels |

## 5.24. Channel Enable Mask (0x8120; r/w)

| Bit | Function |
|---|---|
| [7:4] | *reserved* |
| [3] | 0 = Channel 3 disabled<br>1 = Channel 3 enabled |
| [2] | 0 = Channel 2 disabled<br>1 = Channel 2 enabled |
| [1] | 0 = Channel 1 disabled<br>1 = Channel 1 enabled |
| [0] | 0 = Channel 0 disabled<br>1 = Channel 0 enabled |

Enabled channels provide the samples which are stored into the events (and not erased). The mask cannot be changed while acquisition is running.

## 5.25. ROC FPGA Firmware Revision (0x8124; r)

| Bit | Function |
|---|---|
| [31:16] | Revision date in Y/M/DD format |
| [15:8] | Firmware Revision (X) |
| [7:0] | Firmware Revision (Y) |

Bits [31:16] contain the Revision date in Y/M/DD format.
Bits [15:0] contain the firmware revision number coded on 16 bit (X.Y format).

## 5.26. Event Stored (0x812C; r)

| Bit | Function |
|---|---|
| [31:0] | This register contains the number of events currently stored in the Output Buffer |

This register value cannot exceed the maximum number of available buffers according to setting of buffer size register.

## 5.27. Board Info (0x8140; r)

| Bit | Function |
|---|---|
| [23:16] | Number of channels (DT5720: 0x04; DT5720A: 0x02) |
| [15:8] | Memory size code (DT5720: 0x02) |
| [7:0] | Board Type (DT5720: 0x03) |

## 5.28. Event Size (0x814C; r)

| Bit | Function |
|---|---|
| [31:0] | Nr. of 32 bit words in the next event |

## 5.29. Control (0xEF00; r/w)

| Bit | Function |
|---|---|
| [7] | Reserved; must be set to 0, Release On Register Access (RORA) Interrupt mode |
| [6] | Reserved, must be set to 0 |
| [5] | Reserved, must be set to 0 |
| [4] | Reserved, must be set to 1 |
| [3] | 0 = interrupt disabled<br>1 = interrupt enabled |
| [2,1] | Reserved |
| [0] | Reserved (must be set to 0) |

Interrupt request can be removed by accessing this register and disabling the active interrupt level

## 5.30. Status (0xEF04; r)

| Bit | Function |
|---|---|
| [2] | 0 = Slave Terminated Transfer Flag: no terminated transfer<br>1 = Slave Terminated Transfer Flag: one transfer has been terminated by DT5720 (unsupported register access or block transfer prematurely terminated in event aligned reaout) |
| [1] | 0 = The Output Buffer is not FULL;<br>1 = The Output Buffer is FULL. |
| [0] | 0 = No Data Ready;<br>1 = Event Ready |

## 5.31. Interrupt Status ID (0xEF14; r/w)

| Bit | Function |
|---|---|
| [31..0] | This register contains the STATUS/ID that the module places on the data stream during the Interrupt Acknowledge cycle |

## 5.32. Interrupt Event Number (0xEF18; r/w)

| Bit | Function |
|---|---|
| [9:0] | INTERRUPT EVENT NUMBER |

If interrupts are enabled, the module generates a request whenever it has stored in memory a Number of events > INTERRUPT EVENT NUMBER

## 5.33. Block Transfer Event Number (0xEF1C; r/w)

| Bit | Function |
|---|---|
| [15:0] | This register contains the number of complete events which has to be transferred via Block Transfer (see § 3.7). |

## 5.34. Scratch (0xEF20; r/w)

| Bit | Function |
|---|---|
| [31:0] | Scratch (*to be used to write/read words for test purposes*) |

## 5.35. Software Reset (0xEF24; w)

| Bit | Function |
|---|---|
| [31:0] | A write access to this location allows to perform a software reset |

## 5.36. Software Clear (0xEF28; w)

| Bit | Function |
|---|---|
| [31:0] | A write access to this location clears all the memories |

## 5.37. Flash Enable (0xEF2C; r/w)

| Bit | Function |
|---|---|
| [0] | Reserved for Firmware upgrade tool |

## 5.38. Flash Data (0xEF30; r/w)

| Bit | Function |
|---|---|
| [7:0] | Data to be serialized towards the SPI On board Flash |

This register is handled by the Firmware upgrade tool.

## 5.39. Configuration Reload (0xEF34; w)

| Bit | Function |
|---|---|
| [31:0] | A write access to this register causes a software reset, a reload of Configuration ROM parameters and a PLL reconfiguration. |

# 6. Installation

## 6.1. Power ON sequence

To power ON the board follow this procedure:

1. connect the 12V dc power supply to the DT5720
2. power up the DT5720

## 6.2. Power ON status

At power ON the module is in the following status:

- the Output Buffer is cleared;
- registers are set to their default configuration

## 6.3. Firmware upgrade

The DT5720 firmware is stored onto on-board non-volatile memory. CAEN provides a firmware upgrade tool (see § 4) that can be used with either USB or optical link paths. Firmware updates are available in the Digitizer web page, while the software package, application notes and user manual are available in the CAENUpgrader web page at www.caen.it; follow the instructions for installation and usage.
WARNING: in case of programming failures, the board hosts a backup image of factory firmware.
Please contact CAEN at support.frontend@caen.it for instuctions in order to restore the backup image.
Once the board is successfully powered with backup firmware, the standard firmware image can be reprogrammed.

## 6.4. Drivers

DT5720 needs CAEN USB driver to be installed in order to use the USB communication channel:

- **Download** the driver package compliant to your Operating System (Windows or Linux) on CAEN web site in the 'Software/Firmware' area at the digitizer page.
- **Uncompress** the package to your host.
- **For Windows users:**
  - **Installer option:** with the hardware not connected, run the single installer file and complete the installation Wizard. Then, connect the hardware and the driver will be automatically find by the OS.
  - **Drivers option:** connect the hardware; then, perform the driver installation by pointing Windows to the folder where driver files have been extracted.
- **For Linux users:** follow the installation instructions inside the README file in the package.

Concerning the OPTICAL LINK communication channel, please refer to A2818 PCI card or A3818 PCIe cards User Manual for driver installation.