

digiTES User Manual

Author	Carlo Tintori
Creation Date	22/10/15
Last Update	16/10/17
Document Revision	2.3
digiTES revision	4.5.8
Number of pages	30
File Name	digiTES.docx

Index

Index	1
1. Author's note	3
2. What is digiTES.....	3
3. Architecture of digiTES.....	5
4. Quick start guide to installation and first run	6
4.1. SW package	6
4.2. System Requirements.....	6
4.3. Installation	7
4.4. Run digiTES	7
4.5. Some tips	10
5. Source Files description and notes for the compilation	10
6. Configuration File.....	11
6.1. Directives.....	12
6.1.1. [COMMON].....	12
6.1.2. [BOARD b].....	12
6.1.3. [CHANNEL c]	12
6.1.4. @OFF	12
6.1.5. @ON	12
6.1.6. #.....	12
6.1.7. Load filename	12
6.2. Connection to the boards.....	12
6.2.1. Open	12
6.3. Acquisition Modes	14
6.3.1. AcquisitionMode.....	14
6.3.2. EventBuildMode	14
6.3.3. WaveformProcessor	14
6.3.4. RecordLength.....	14
6.3.5. PreTrigger	14
6.3.6. EventBuffering	14
6.3.7. RunNumber	14
6.4. Input Data Files.....	14
6.4.1. InputFileType	14
6.4.2. InputDataFilePath.....	14
6.4.3. InputDataFileName.....	15
6.4.4. LoopInputFile	15
6.5. Output Data Files.....	15
6.5.1. SaveLists	15
6.5.2. SaveRawData	15
6.5.3. SaveHistograms	15
6.5.4. SaveWaveforms	15
6.5.5. HistoOutputFormat.....	15
6.5.6. OutFileFormat.....	15
6.5.7. OutFileTimeStampUnit	16
6.5.8. DataFilePath.....	16
6.5.9. HeaderInListFiles.....	16
6.5.10. ConfirmFileOverwrite	16
6.6. Sync and Triggers.....	16
6.6.1. FPIOtype	16
6.6.2. StartMode.....	16
6.6.3. SyncinMode	16
6.6.4. TrginMode	17
6.6.5. VetoWindow	17
6.6.6. TrgoutMode.....	17
6.6.7. TrgoutMask.....	17
6.7. Coincidences in hardware	18
6.7.1. CoincMode.....	18
6.7.2. MajorityLevel	18
6.7.3. CoincWindow.....	18
6.8. Input signals.....	18
6.8.1. EnableInput.....	18
6.8.2. PulsePolarity	18

6.8.3.	BaselineDCOffset	18
6.8.4.	ZeroVoltLevel	19
6.8.5.	InputDynamicRange	19
6.9.	Discriminator	19
6.9.1.	DiscrMode	19
6.9.2.	TriggerThreshold	19
6.9.3.	TrgHoldOff	19
6.9.4.	TTFsmoothing	20
6.9.5.	TTFdelay	20
6.9.6.	CFDDelay	20
6.9.7.	CFDfraction	20
6.9.8.	EnableZCcal	20
6.10.	Gated Charge Integration (DPP PSD/CI)	20
6.10.1.	GateWidth	20
6.10.2.	ShortGateWidth	20
6.10.3.	PreGate	21
6.10.4.	ChargeSensitivity	21
6.10.5.	NSBaseline	21
6.10.6.	FixedBaseline	21
6.10.7.	PileUpMode	21
6.10.8.	PurGap	21
6.10.9.	ChargeLLD	22
6.11.	Trapezoidal Filter (DPP PHA)	22
6.11.1.	TrapRiseTime	22
6.11.2.	TrapFlatTop	22
6.11.3.	TrapPoleZero	22
6.11.4.	PeakingTime	22
6.11.5.	NSPeak	22
6.11.6.	PeakHoldOff	22
6.11.7.	TrapNSBaseline	22
6.11.8.	Decimation	22
6.12.	Energy Spectrum	23
6.12.1.	EHnbin	23
6.12.2.	EnergyCoarseGain	23
6.12.3.	EnergyFineGain	23
6.12.4.	ECalibration	23
6.12.5.	EnableEnergyFilter	23
6.12.6.	EnergyLCut	23
6.12.7.	EnergyUCut	23
6.12.8.	EnableEnergySkim	23
6.13.	Time Spectrum	24
6.13.1.	TspectrumMode	24
6.13.2.	THnbin	24
6.13.3.	THmin	24
6.13.4.	THmax	24
6.13.5.	TOFstartChannel	24
6.13.6.	TOFstartBoard	24
6.13.7.	TstampOffset	24
6.13.8.	TimeCorrelWindow	24
6.14.	PSD Spectrum	24
6.14.1.	EnablePSDFilter	24
6.14.2.	PsdLCut	25
6.14.3.	PsdULD	25
6.14.4.	ScatterPlotMode	25
6.15.	MCS Spectrum	25
6.15.1.	MCSHnbin	25
6.15.2.	DwellTime	25
6.16.	Stop Criteria	25
6.16.1.	StopOnTime	25
6.16.2.	StopOnLiveTime	25
6.16.3.	StopOnTotalEvents	25
6.16.4.	StopOnEnergyEvents	25
6.16.5.	StopOnTimeEvents	26
7.	Command line options	27
8.	Online Keyboard Menu	27

1. Author's note

I started to develop the program **digiTES** for my personal tests and measurements with the digitizers and the DPP algorithms, but also to quickly enable the use of our digital solutions in a variety of physics applications and experiments in collaboration with the customers. I have been using **digiTES** in the last 2-3 years in different conditions and every time I added new features or fixed bugs. It is far to be a professional software and I am not a software developer, so the code style is probably ugly and not optimized, but I consider **digiTES** an useful and flexible tool and for this reason I decided to share it to whoever may take advantage from it.

This software is not monkey proof! There are very few checks that prevent the user to do wrong settings or operations that make the program to crash. I tested it with many board models and firmware types, but not all of the available combinations, so I don't guarantee that it works in any conditions. Please report bugs, comments and suggestions for improvements to me by email; I will appreciate your feedback, but I don't promise that I will follow the suggestions, especially promptly.

*The program **digiTES** is a free software distributed 'as is' in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation.*

Carlo Tintori (c.tintori@caen.it)

2. What is **digiTES**

The program **digiTES**, in conjunction with CAEN's waveform digitizers, implements a **Multi Parametric DAQ for Physics Applications**, where the detectors can be connected directly to the digitizers inputs and the software acquires energy, timing (TAC), MCS and PSD spectra and lists. Such a digital system makes it possible to have in one single board the replacement of almost any conventional analog electronics (QDC, peak sensing ADC, TDC, discriminators, scalers, coincidence units).

The name **digiTES** is an abbreviation of "digital acquisition of Time, Energy and Shape", meaning that the triplet **Time, Energy, PSD** is the basic *fragment* of information acquired, processed and saved by the program. This fragment is represented in the software as an '**event**', which is a formatted data packet containing the triplet T,E,S, some flags and, optionally, a waveform of programmable length.

Typically, the digitizers run a **DPP firmware** (Digital Pulse Processing) and the parameters of the triplet are calculated on-board by the FPGA algorithms. Starting from **digiTES** rev 3.2, the program also supports the acquisition with boards running the standard firmware (pure waveform mode); in this case, the software processes the waveforms to extract the desired parameters (T, E, S) and to complete the event data structure.

DigiTES can manage multiple board acquisitions (currently up to 8 boards), provided that they are the same model and run the same firmware.

digiTES abilities:

- manage a multi-parametric, multi-board readout system for a wide variety of detectors
- synchronize multiple boards in different modes
- control the HV channels in the modules DT5780 and DT5790
- select the configuration file between a set of preconfigured files (e.g. HPGe, NaI, LaBr3...)
- apply correlation between channels at hardware level: coincidence and anti-coincidence, majority, trigger propagation, external Veto or Gate, etc...
- push events into queues and extract them in tuples according to event selection and building criteria:
 - energy and PSD windowing (cuts)
 - coincidence between a reference channel (common start or stop) and any other channel

- Majority of M channels (or just a logic OR) in a group of N (e.g. Clover detectors)
- Paired (couples of channels)
- create virtual channels to collect data produced by a combination of other channels; for instance, the add-back energy of a clover detector
- calculate, display and save the statistics (trigger rate, data throughput, coincidence matching ratio, dead time and others)
- read raw waveforms (Std FW) and apply basic DPP algorithm to calculate charge and time stamp in the software
- plot waveforms (oscilloscope mode): input signal, internal filters (CFD, baseline, trapezoid...), FFT
- acquire and plot spectra: energy, PSD, 2D scatter plot with energy and PSD, deltaT between two channels, deltaT between consecutive triggers (events distribution), MCS (counts in dwell time bins)
- waveform skimming: define a ROI in a spectrum (energy or PSD) and plot the relevant waveforms (pulses that produced events in the selected ROI)
- save output data files (binary or ASCII):
 - raw data
 - lists: T, E, S of individual channels, merged list with all channels, built list with correlated events
 - waveforms (T, E, S + samples)
 - energy, timing and PSD spectra
- use output data files (raw and lists) for off-line runs, with the possibility to apply a different event selection and building
- use the run number (manual set or automatic increment) to define the output file names and to generate a "run_info" text report
- manage job scheduling: it is possible to program a certain number of runs with different settings and conditions
- manage the zero crossing calibration algorithm if the digital CFD to improve the timing resolution of fast signals
- run in emulation mode, where digiTES produces "fake" data emulating exponential pulses
- register read/write access with a manual controller; save register image to compare board settings

digiTES inabilities:

- heterogeneous multiple board systems
- support for models x740, x742, x743, x731
- GUI to set the parameters and control the acquisition

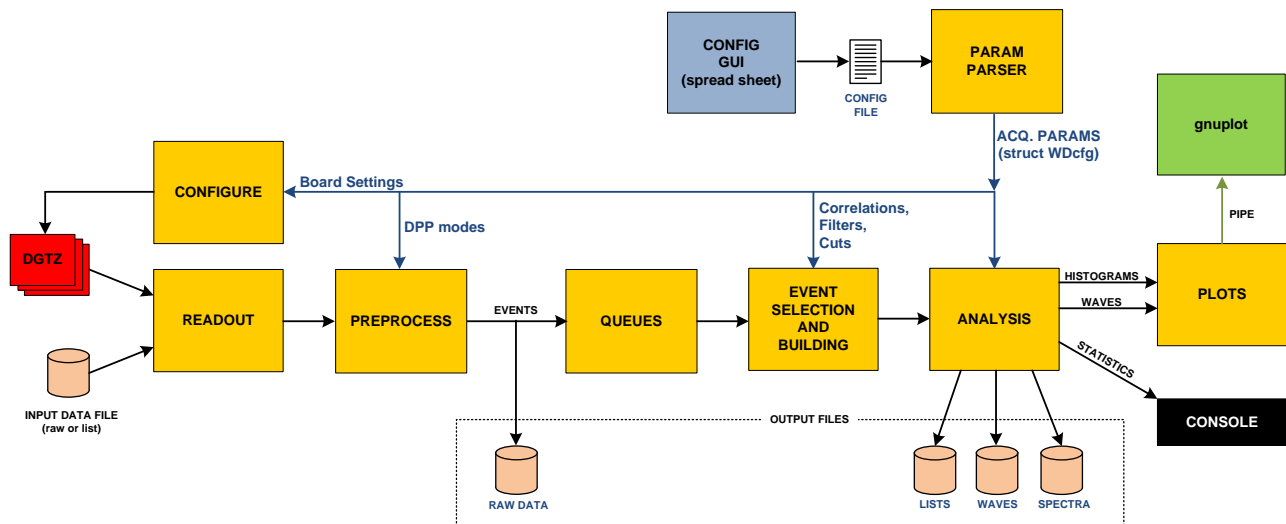
DigiTES supports the following models and firmware:

Model	MS/s	Bit	StdFW	DPP_PSD	DPP_CI	DPP_PHA	DPP_nPHA ⁽¹⁾	DPP_QDC
x730	500	14	✓	✓		✓		
x725	250	14	✓	✓		✓		
x751	1000	10	✓	✓				
x761	4000	10	✓					
x720	250	12	✓	✓	✓			
x724	100	14	✓			✓	✓	
x780	100	14				✓	✓	
x781	100	14				✓	✓	
x740	65	12	✗					✗
x742	5000	12	✗					
x743	3200	12	✗					
x731	1000	8	✗					

⁽¹⁾ DPP_nPHA is the new DPP_PHA of the MCAs (revisions with minor number >= 64)

3. Architecture of digiTES

In the following picture, the yellow blocks are parts of the digiTES programs. The parameters for the acquisition, either hardware settings or parameters for the software data processing and analysis, are all contained in a textual **configuration file**. This can be directly edited by the user or generated by an Excel spread sheet.



Once started, digiTES parses the configuration file and reads the parameters (storing them in the C *struct* called **WDcfg**). In it is an on-line run, the program opens and configures the boards according to the parsed parameters. When the acquisition starts, the raw data coming from the readout module are preprocessed and re-formatted. After the preprocessor, the events have a unique data format that is independent of the hardware and firmware type. They contain the following fields (see the struct *GenericDPPEvent_t* in digiTES.h):

- **TimeStamp** (64 bit) expressed in ns
- **FineTimeStamp** (16 bit) obtained by the samples interpolation of the digital discriminator (CFD or LED); the timing interpolator can be hardware (only for models x730, x725, x751) or software (in this case it is necessary to read the waveforms from the board). The FineTimeStamp is expressed in ps.
- **Energy** (16 bit); this is the pulse height in the DPP_PHA or the pulse charge in the DPP_PSD
- **PSD** (float in the range 0.0 – 1.0); this is a pulse shape discrimination factor. In the DPP_PSD, it is the ratio between the charge in the tail of the signal (slow component) and the total energy:

$$PSD = (Q_L - Q_S) / Q_L$$

In the DPP_PHA, this field is normally meaningless (set to 0), but it would be possible to use another definition of PSD, such as the pulse rise time, calculate it in the software waveform processor and put the result into this field.

- **Flags**: (16 bits) these are various flags indicating pile-up, overflow, etc... There are flags that are not implemented in some versions of the firmware.
 [bit 0] = dead time: a memory full condition or an input saturation occurred between this event and the previous one, so there is a gap where it might be a loss of events [bit 1] = Saturation
 [bit 1] = Saturation: amplitude over/under range (either input signal or energy filters).
 [bit 3] = Fake Event; this is not a real event, it is artificially added by the board, e.g. to track the timestamp rollover. This event is not passed to the data analysis.
 [bit 4] = Pile-up: the energy is not calculated correctly.
 [bit 6] = Time stamp cleared by SIN
 [bit 8] = Energy Over-range
 [bit 9] = Energy Under-range

- **Waveform:** (pointer to the waveform data structure); this field is *NULL* if the waveform readout is not enabled, otherwise it points to a memory buffer where the waveform data can be retrieved.

The formatted events are all pushed into the queues (one per channel) that are able to store a big number of events before they are extracted by the **event builder** and passed to the analysis. If the event building is disabled (no timing correlation between channels), then the events are extracted from the queues in parallel, one per channel, until all the queues are empty. Instead, if the event builder need to group events belonging to a given time window (timing correlation), it is necessary to keep events in the queues until all the correlated channels have data to compare. It may happen that a high counting rate channel accumulates many events in its queue while another low rate channel doesn't give any data for a certain lapse of time. The queues are allocated large enough to compensate for significant disparities in the readout latency of different channels. In case one channel becomes inactive or very slow, it is necessary to prevent the queues to overflow while the event builder waits for data from the inactive channel. There are two mechanisms: the time stamp of the oldest event in the queues is compared with the youngest time stamp read from the board; if the difference is higher than a given timeout, then the event builder goes on in applying the correlation criteria, considering the inactive channels as empty and consuming the events from the other queues. Of course, if the coincidence with an inactive channel is requested, the event builder will reject the events. In parallel to the coincidence timeout, the occupancy of the queue is continuously monitored and the oldest data are discarded from the queues exceeding the almost full level until the occupancy returns below the threshold. Discarded events are always counted and reported in the statistics of the acquisition.

The built events (array of N events, max. one per channel) are read by the block "Analysis" (implemented in the *main* of *digiTES*) that builds the histograms, update the statistics and save the enabled output data files. It also plots the spectra and/or the waveforms and prints the some statistics to the console.

The plots are currently done with *gnuplot*. This is a portable command-line driven graphing utility for Linux and Windows (<http://www.gnuplot.info/>). *DigiTES* opens *gnuplot* as a pipe (*popen*) and uses it to send the commands.

4. Quick start guide to installation and first run

4.1. SW package

There is not any installer for *digiTES*; it comes as a ZIP file that contains the following directories and files:

- **Readme.txt:** readme file with basic information
- **digiTES_UserManual:** (this file) User Manual.
- **ReleaseNotes.txt:** Release Notes of the last software release
- **src:** directory containing the source files (*.cpp); also the Makefile for Linux version
- **include:** directory containing the header files (*.h)
- **build:** directory containing the MS Visual C++ project files (only for Windows version)
- **lib:** directory containing precompiled libraries for Windows (ZCcal.lib) or objects for Linux (ZCcal.o)
- **bin:** working directory containing executables, configuration files, Excel GUI (for Windows only), etc..
- **bin/DataFiles:** Default destination directory (empty) for the output data files

Windows installation may have an additional directory containing *gnuplot*; this folder can be deleted if *gnuplot* (ver. 5.0 or higher) is already installed, but it is necessary to set the correct path to the installed *gnuplot* in the system variables (see file *sysvars.txt* in *digiTES/bin*)

4.2. System Requirements

For Windows:

- Windows 7 or above, 32 or 64 bit
- Microsoft Visual C++ (2010 edition or above); this is only required in case the user wants to modify the program and recompile it

For Linux:

- kernel 2.4 or 2.6 and GNU C/C++ compiler
- glibc version 2.11.1 or above
- gnuplot version 5.0 or above (www.gnuplot.info)

For both:

- CAENVME library version 2.5 or above (<http://www.caen.it/csite/CaenProd.jsp?idmod=689&parent=38>)
- CAENComm library version 1.2 or above (<http://www.caen.it/csite/CaenProd.jsp?parent=38&idmod=684>)
- CAENDigitizer library version 2.8.0 or above (<http://www.caen.it/csite/CaenProd.jsp?parent=38&idmod=717>)
- Drivers (at least one, depending on the connection type)
 - DT57xx-N67xx-DT55xx-V1718-N957 USB Driver (direct USB to Desktop/NIM or VME through V1718)
 - A3818 Driver (direct Optical Link to Desktop/NIM or VME through V2718)
 - A2818 Driver (same as A3818)

4.3. Installation

Once unzipped the software package, go to into the *bin* directory. For Windows 32 bit systems, replace **diGiTES.exe** with **diGiTES_32bit.exe**. For Linux, type 'make' and press enter (the Makefile in *bin* will call the main Makefile for the compilation in *src*); check that the executable file 'diGiTES' is created in *bin*.

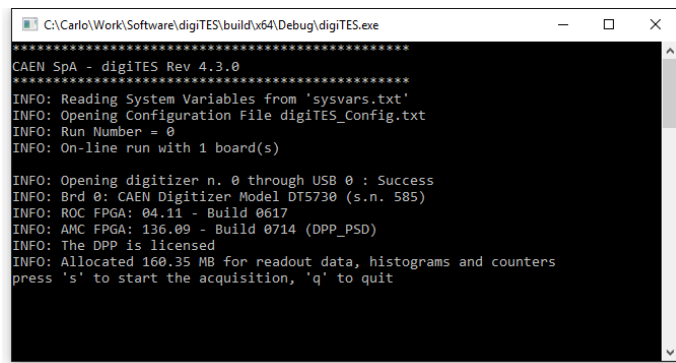
Edit the default configuration file *diGiTES_Config.txt*: by default, the connection parameters are for a Desktop digitizer connected to the computer through the USB port. If you have a different connection, you must change the *Open* command. Some examples are reported below:

```
[BOARD 0] Open USB 0 0:          computer => USB => DT57xx or N67xx (default setting)
[BOARD 0] Open USB 0 3210000:    computer => USB => V1718 => VMEbus => V17xx (Base Address = 0x32100000)
[BOARD 0] Open PCI 0 0 0:        computer => A3818 (or A2818) => CONET Optical Link => DT57xx, N67xx or
                                V17xx
[BOARD 0] Open PCI 0 0 3210000:  computer => A3818 (or A2818) => CONET Optical Link => V2718 => VMEbus =>
                                V17xx (Base Address = 0x32100000)
```

See par. 6.2 for more information about the connection parameters.

4.4. Run diGiTES

To run the program, double click on *diGiTES.exe* for Windows or type *./diGiTES* for Linux. The program opens the configuration file and parses the parameter settings, then tries to open the connection to the board and gives information about the board type, serial number, firmware revision, etc..., as showed in the screen shot below:



```

C:\Carlo\Work\Software\digiTES\build\vx64\Debug\digiTES.exe
*****
CAEN SpA - digiTES Rev 4.3.0
*****
INFO: Reading System Variables from 'sysvars.txt'
INFO: Opening Configuration File digiTES_Config.txt
INFO: Run Number = 0
INFO: On-line run with 1 board(s)

INFO: Opening digitizer n. 0 through USB 0 : Success
INFO: Brd 0: CAEN Digitizer Model DT5730 (s.n. 585)
INFO: ROC FPGA: 04.11 - Build 0617
INFO: AMC FPGA: 136.09 - Build 0714 (DPP_PSD)
INFO: The DPP is licensed
INFO: Allocated 160.35 MB for readout data, histograms and counters
press 's' to start the acquisition, 'q' to quit
  
```

If the connection fails, the program gives an error message *Can't open digitizer n. 0*. If so, check that you have installed the proper driver for the USB device or for the PCI/PCIe card. Use the Windows *device manager* or the command *lsmod* in Linux to verify that the driver is correctly installed. Check also the Connection parameters in the config file, as described above.

If the board is successfully connected, press 's' to start the acquisition. The light 'run' on the front panel of the board goes on. By default, the configuration file is set for the Mixed Acquisition mode, that is List + Waveforms, so the program is able to acquire and plot both waveforms (oscilloscope mode) and spectra (T, E, S). It may happen that the parameters for the discriminator (self-trigger), the pulse polarity, the baseline, etc... are not properly set for your setup, so the digitizer may not be able to trigger and to acquire events. You will see "0 Hz" on each channel, meaning "no data"; the trigger and data ready lights on the front panel are both off. You can "force" triggers in the board by pressing 't' (one single trigger) or 'T' (enable/disable continuous triggers); for each software trigger, the board acquires one event in each channel, so you should be able to see at least some data and the baseline of the signal in the waveforms plot. Please note that the plot pops up in a different window (it is actually the window of *gnuplot*) and this new window takes the "focus"; you must click on the console window and move the focus there if you want to send keyboard commands (for instance 'q' to quit). Some versions of *gnuplot* takes the focus again every time a new plot is drawn and doesn't allow you to operate on the console window; if so, open the configuration dialog of *gnuplot* (icon with a wrench) and disable the option "plot the window at the top of your desktop after each plot (raise)".

The console window, when on focus, has a set of binbkeys (keyboard commands) described in chap. 8. Press the space bar to have an on-line help. Also the plot window, when on focus, has some bindkeys: 'a' to autoscale (both X and Y), 'y' to autoscale on Y only, 'l' to toggle between log and linear scale, 'p' to go to the previous zoom, 'r' to enable/disable the ruler. To zoom in *gnuplot*: right-click on 1st corner, release, left-click on 2nd corner.

The plots are referred to one channel only (0 by default); you can change the plotted channel by pressing 'c' in the console window, then entering the channel number. In case of multiple boards, you can change the board by pressing 'b'.

Now it is time to set parameters suitable for your signals. The directory *bin* contains some preconfigured files: *digiTES_Config_HPGe.txt* for germanium detector or in general for any signal coming from a charge sensitive preamplifier with resistive feedback (boards running the PHA firmware) and *digiTES_Config_LaBr3.txt* suitable for most scintillation detectors with PMT (boards running the PSD or CI firmware). To use one of these files, rename it as *digiTES_Config.txt* (that is the default name) or pass it as a parameter on the command line: *digiTES config_file_name*. Type *digiTES -h* for help about the command line options.

Edit the config file you have chosen and check at least the following important parameters:

EnableInput: enable/disable channels. You must have at least one channel enabled. If this setting is in the [COMMON] section of the config file, then it applies to all channels (all enabled or all disabled). Otherwise, you can put it in the [BOARD b] [CHANNEL c] section to apply to board *b*, channel *c* only.

PulsePolarity: NEGATIVE or POSITIVE

BaselineDCoffset: offset level of the signal baseline expressed in percent of the full scale. Typically 10 is a good setting (10% of the scale). If you have doubt about the position of the baseline, try 50% to see where it falls.

CoincMode: for now, set it to DISABLED (no coincidence is applied in the hardware)

DiscrMode: use LED (Leading Edge Discriminator) for PSD firmware

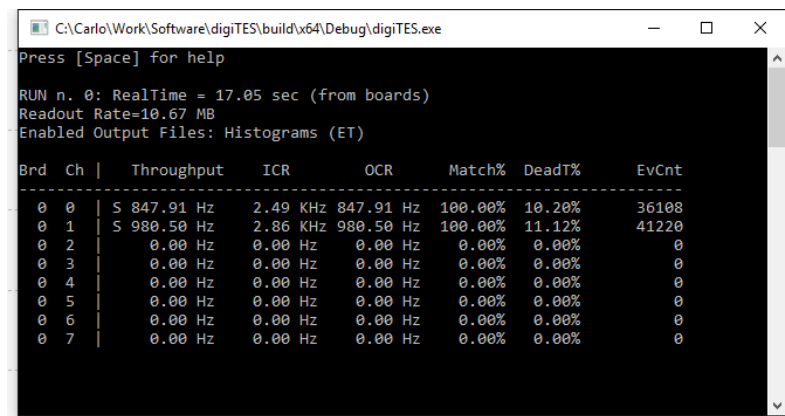
TriggerThreshold: digital discriminator threshold (for the self-trigger) expressed in ADC channels. Start with values around 100 for x730, x724, 20 for x720, x725, 5 for x751.

NSBaseline: option that defines the number of points for the baseline. Try with options 2 or 3.

There is also a configuration file (*digiTES_Config_with_comments.txt*) where each parameter is briefly described with a comment reporting also allowed values and options. An extensive description of the parameters is in chap. 6.

When the parameters in the configuration file are properly set for the application, you should see the waveforms of the pulses in the oscilloscope plot; you can change the length of the waveform and the pre-trigger size by means of the parameters **RecordLength** and **PreTrigger** in the config file. Every time you change a parameter in the config file, you need to restart digiTES.

While the acquisition is running, the console window shows some run information and statistics; these are updated every second. The **RealTime** (from the start of the acquisition) is given by the youngest time stamp of the events (from boards) or by the computer time when no data is available (from computer). The **Readout Rate** is the number of bytes per second transferred from the board(s) to the computer. **Enabled Output** tells you which output files are saved to the disk (Histograms of T, E, S, Waveforms, Lists). For each board/channel, the reported information is the **Throughput** (number of events per second transferred from the boards to the computer), the **ICR** (input count rate), the **OCR** (output count rate), the **Matching Ratio** (ratio between the counting rate before and after the event selection applied by the software), the **Dead Time** (estimated ratio between lost events and acquired events) and **EvCnt** (the total number of output counts).



You can change the reported info by pressing 'Tab', having statistics about **saturated** events (clipped pulses that exceeded the input dynamic range), estimation of the **Busy** time (the board is busy when it is not able to accept a trigger, typically because of a memory full condition) and the **Queue Occupancy**.

During the acquisition, digiTES acquires the events and builds the energy, timing and PSD spectra for each channel. If you press 'e', the Energy Spectrum (of the active channel) will be plotted in a new window. Use 'p' to plot the PSD spectrum, 'P' (capital p) to make the 2D scatter plot with PSD on the y-axis and Energy on the x-axis. Use 'd' to plot the timing spectrum (TAC). This can be the distribution of the intervals between two triggers in the same channel (typically a poissonian) or the Start-Stop measurement between a common start (reference channel) and the other channels. Use 'm' to plot the MCS spectrum (Multi Channel Scaler), which is the number of counts (y-axis) as a function of the time (x-axis); each channel (bin) of the spectrum represents a given counting period called dwell time.

Press 'q' to quit.

4.5. Some tips

It is possible to split the configuration file in several files, each containing different settings. There is a main file in which you can call other files by means of the command **Load**. Nested calls are allowed. If one parameter has multiple settings, the last one will be applied. Splitting the configuration file is useful in many cases: for instance, if you want need to change frequently between runs with waveforms (mixed) and runs in list mode and you don't want to edit the configuration file every time, you can create three files:

Wmode.txt:

```
AcquisitionMode MIXED
Load GeneralConfig.txt
```

Lmode.txt:

```
AcquisitionMode LIST
Load GeneralConfig.txt
```

GeneralConfig.txt (contains all settings but AcquisitionMode)

Now you can execute *digiTES Wmode* to run with waveforms and *digiTES Lmode* to run in list mode. Another possibility is to use a dedicated file (loaded by the main config file) for the channel enable mask:

EnableMask.txt:

```
[BOARD 0] [CHANNEL 0] EnableInput 1
[BOARD 0] [CHANNEL 1] EnableInput 1
[BOARD 0] [CHANNEL 2] EnableInput 0
[BOARD 0] [CHANNEL 3] EnableInput 0
[BOARD 1] [CHANNEL 0] EnableInput 0
[BOARD 1] [CHANNEL 1] EnableInput 1
[BOARD 1] [CHANNEL 2] EnableInput 1
[BOARD 1] [CHANNEL 3] EnableInput 0
```

In general, you can make dedicated config files for the most important parameters that you change more frequently (e.g. energy calibration, thresholds, etc...) and leave the other "stable" parameters in a single file that you normally don't touch.

5. Source Files description and notes for the compilation

DigiTES is a C (and not C++) program; the reason why the source files are *.cpp* is that in some cases it has been necessary to use the C++ compiler to link digiTES with other C++ libraries.

The program is almost platform independent; the few functions that differ from Window to Linux are concentrated in *console.cpp*.

To compile digiTES, you need the CAEN libraries reported in par. 4.2.

This is the list of the *.cpp* source file (each having the relevant *.h* header file):

- **digiTES**: contains the main (initialization, start of run, data analysis and output file saving of processed data), the keyboard menu manager and other functions for the variables and hardware initialization.
- **Configure**: contains functions that configure the digitizers with the parameters taken from the *WDcfg* structure. It also defines the waveform names for the traces in the plot, according to the board model and firmware type.

- **BoardUtils**: contains some utility functions for the low level access to the digitizer registers and for the acquisition control (start, stop). Furthermore, it contains the Manual Controller that is an on-line tool that allows the user to read/write registers, save the image of all of the registers and other utilities.
- **ParamParser**: initializes the structure WDcfg to the default values, then it parses the configuration file to read the parameters and overwrite the relevant fields in the WDcfg struct.
- **DataFiles**: manages the input/output files (raw data, lists, histograms and waveforms).
- **Plots**: waveforms and histograms plotter (using 'gnuplot' as a plotting engine).
- **Histograms**: manages the histograms allocation, initialization and accumulation
- **Statistics**: calculates counting rates, total counts, real and dead time, saturation ratio, busy time, etc...
- **fft**: calculates the FFT of a waveform
- **Queues**: allocates the memory buffers for the event queues, then provides the functions to push and pop events. The data transfer is driver by the main: the analysis process requests an event array; this request is managed in the *queues* block that tries to pop events from the queues according to the selection criteria. If the queues are empty or there are events waiting for matching events from correlated channels, then the selector calls the readout function to get new data from the boards (or from the input file for off-line runs).
- **Readout**: allocates memory buffers for the readout and the decoding, than reads data (from boards or from files), decodes the raw data into events (including the preprocessing) and pushes them into the queues.
- **PreProcess**: this is the event and waveform data pre-processor; it gets events from the boards (having different formats according to the firmware and model type) and returns reformatted events according to the *GenericDPPEvent_t* struct. This file contains also the **WaveformProcessor** that can be enabled to process the incoming event waveform and calculate one or more parameters of the triplet "T, E, PSD"; currently, it performs a timing interpolation to get the FineTimeStamp and also a dual gate charge integration to get Energy and PSD.
- **Console**: contains the functions the management of the low level console input (keyboard) and output (terminal). These functions are normally different for Windows and Linux.

6. Configuration File

By default, the program open a configuration file with the name "*digITES_Config.txt*". It contains a list of settings in the form "*KEYWORD value*", where *KEYWORD* identifies a specific parameter and *value* is the relevant setting. The value can be an integer a float or a string depending of the *KEYWORD*.

Besides the *keywords* for the parameters, there are also some *directives*. The lines starting with # are comments and will be ignored by the parser. It is also possible to exclude a complete block of text by including it between the directives *@ON* and *@OFF* (similar to *#ifdef*, *#endif* in C).

The configuration file contains different sections: the parameters that follow the directive *[COMMON]* are parameters that cannot be applied individually or that are broadcasted to all of the channels and to all of the boards; the parameters that follow the directive *[BOARD b]* are apply to all of the channels of board 'b'; the parameters that follow the directive *[CHANNEL c]* (that must stay within a *[BOARD b]* section) apply to channel 'c' of board 'b' only. The lines of the config file are parsed sequentially; is the same parameter is set in different parts, the last setting overwrites the previous ones; for this reason, it makes sense to put the common settings first, thus defining a default setting to all of the channels/boards, then put the individual settings that overwrite the default ones where needed. An simple example should make the directives clear:

```
# BOARD 0: open the digitizer at VME address 0x32100000 with a V1718 USB bridge
[BOARD 0] Open USB 0 0x32100000
# BOARD 1: open the digitizer at VME address 0x32110000 with a V1718 USB bridge
[BOARD 1] Open USB 0 0x32110000

[COMMON]
GateWidth      100      # default gate width is 100 ns
TrgThreshold    30      # default trigger threshold is 30 ADC counts
```

```
[BOARD 0]
GateWidth      80      # all channels in board 0 have GateWidth = 80 ns

[BOARD 1]
[CHANNEL 0]
GateWidth      110     # channel 0 in board 1 has GateWidth = 110 ns
# individual threshold settings for board 1:
[CHANNEL 0]    TrgThreshold  33 # threshold for brd1 - ch0
[CHANNEL 1]    TrgThreshold  34 # threshold for brd1 - ch1
[CHANNEL 2]    TrgThreshold  30 # threshold for brd1 - ch2
[CHANNEL 3]    TrgThreshold  38 # threshold for brd1 - ch3
```

The configuration file can also load another configuration file by using the *LOAD* directive. It may be convenient to have a common config file (the one opened by the program when started) in which individual board config file are loaded. The common config file contains default settings, common settings (such as acquisition mode, trigger mode, etc... and the list of the individual config file to be loaded (one per board). The individual config files contain the connection parameters (USB or PCI Express optical link, link number, etc...) and the individual channel settings.

NOTE: time values (e.g. gate width) can be expressed as a pure number (taking the default unit, usually ns) or with a specific time unit. For instance, the three settings below are all equivalent:

```
GateWidth      1100     # 1100 ns (the default unit for the gate setting is ns)
GateWidth      1100 ns  # 1100 ns
GateWidth      1.1 us   # 1.1 us = 1100 ns
```

The list of all of Keywords and Directives is reported below divided in functional sections:

6.1. Directives

6.1.1. [COMMON]

Start of the Common Section. The parameters in this section will be applied to all of the boards and channels. The Common section ends when a Board or Channel section is found. It is possible to restart another Common section at any place in the configuration file just writing the directive [COMMON] again.

6.1.2. [BOARD b]

Start of the Board Section. The parameters in this section will be applied to board 'b' only.

6.1.3. [CHANNEL c]

Start of the Channel Section. The parameters in this section will be applied to channel 'c' only.

6.1.4. @OFF

Text following this directive will be ignored by the parser until @ON is reached.

6.1.5. @ON

Restart to parse the configuration file.

6.1.6.

Start of a comment on the line.

6.1.7. Load filename

Another configuration file called *filename* is parsed. At the end of that configuration file, the parser returns back to this one and continues the parsing of parameters. This directive is similar to *#include* in C.

6.2. Connection to the boards

6.2.1. Open

This is the keyword that defines the physical layer for the connection of the board(s) to the computer. There must be an Open command per board and it must stay in the [BOARD b] section. There are different cases:

PC directly connected to the digitizer through USB

This is the case of the Desktop and NIM version.

Syntax: Open USB *pn* 0

Where *pn* is the USB port number and it is 0 for the 1st board enumerated by the PC, 1 for the 2nd board, etc. Unfortunately there is not a fixed relationship between the physical ports of the computer and the port number that the operating system assigns to them. The only guarantee is that if you have N boards connected via USB, the parameter *pn* will go from 0 to N-1.

Example: 2 desktop digitizers connected to the PC

```
[BOARD 0] Open USB 0 0 # open 1st digitizer
[BOARD 1] Open USB 1 0 # open 2nd digitizer
```

PC directly connected to the digitizer through PCI/PCle - Optical Link

The PC, that houses one or more A3818 (PCle) or A2818 (PCI), is connected to the digitizers (any model) through the front panel optical link. The connection can be point to point (one link per board, using a crossed duplex fiber optic cord) or daisy chained. In the latter case, one link of the A3818/A2818 is connected to multiple digitizers (up to 8) using single fiber optic cords to connect the TX port of one board to the RX port of the next one and then a duplex "Y shaped" fiber optic to connect the RX of the 1st board and the TX of the last one to the link of the A3818/A2818. The command is the same for PCI and PCle boards.

Syntax: Open PCI *ln* *cn* 0

Where *ln* is the link number and *cn* is the conet node, that is the position of the board in the daisy chain (zero for the 1st or unique board). The link number of the A3818C (4 link card) goes from 0 (top) to 3 (bottom). In case of multiple A3818C cards in the PC, the link number continues sequentially.

Example 1: four digitizers connected in daisy chain to one A3818 or A2818 link :

```
[BOARD 0] Open PCI 0 0 0 # open 1st digitizer
[BOARD 1] Open PCI 0 1 0 # open 2nd digitizer
[BOARD 2] Open PCI 0 2 0 # open 3rd digitizer
[BOARD 3] Open PCI 0 3 0 # open 4th digitizer
```

Example 2: two digitizers connected point-to-point to link 0 and 1 of the A3818C:

```
[BOARD 0] Open PCI 0 0 0 # open 1st digitizer
[BOARD 1] Open PCI 1 0 0 # open 2nd digitizer
```

PC connected to the digitizer through USB-V1718-VME bus

In this case, the digitizers are accessed through the VME backplane and the CAEN VME master Mod. V1718 (VME to USB interface) is used to access the VME bus through the USB.

Syntax: Open USB *pn* *BA*

Where *pn* is the USB port number (see direct USB connection for the details on the port number) and *BA* is the 32 bit Exadecimal Base Address of the VME digitizer.

Example: 3 VME digitizers and a V1718 connected to the PC via USB

```
[BOARD 0] Open USB 0 AA000000 # open 1st digitizer (Rotary Settings = AA00)
[BOARD 1] Open USB 0 AA010000 # open 2nd digitizer (Rotary Settings = AA01)
[BOARD 2] Open USB 0 AB040000 # open 3rd digitizer (Rotary Settings = AB04)
```

PC connected to the digitizer through PCI/PCle-V2718-VME bus

In this case, the digitizers are accessed through the VME backplane and the CAEN VME master Mod. V2718 (VME to CONET interface) is used to access the VME bus through the PCI/PCle cards and the optical link.

Syntax: Open PCI *ln* 0 *BA*

Where *ln* is the link number and *BA* is the 32 bit Exadecimal Base Address of the VME digitizer. The link number of the A3818C (4 link card) goes from 0 (top) to 3 (bottom). In case of multiple A3818C cards in the PC, the link number continues sequentially.

Example: 3 VME digitizers on the backplane and a V2718 connected to the link 0 of the A3818/A2818

```
[BOARD 0] Open PCI 0 0 AA000000 # open 1st digitizer (Rotary Settings = AA00)
[BOARD 1] Open PCI 0 0 AA010000 # open 2nd digitizer (Rotary Settings = AA01)
[BOARD 2] Open PCI 0 0 AB040000 # open 3rd digitizer (Rotary Settings = AB04)
```

6.3. Acquisition Modes

6.3.1. AcquisitionMode

Parameter type: string

Allowed values: MIXED, LIST, OFF_LINE, EMULATOR_LIST, EMULATOR_MIXED

Description: MIXED and LIST modes are for on-line acquisition (data from the boards); in MIXED mode the waveform readout from the digitizers is enabled, in LIST mode it is disabled. The OFF_LINE mode takes data from the input file (no board connection is required). In EMULATOR mode, digiTES emulates pulses and produce relevant data (MIXED or LIST mode, that is with or without waveforms), so that you can run digiTES without any board or data file, just to play with it.

6.3.2. EventBuildMode

Parameter type: string

Allowed values: NONE, CHREF_AND_ANYOTHER, CLOVER N M

Description: event building mode (= correlation between channels); NONE means no correlation (take events from queue as they come), CHREF_AND_ANYOTHER: one timing reference channel (defined by TOFstartChannel and TOFstartBoard) in coincidence with at least another channel. CLOVER N M: groups of N channels (typ. 4) are acquired if at least M are fired within the coincidence window. A virtual channel is then used to take the AddBack energy.

6.3.3. WaveformProcessor

Parameter type: unsigned integer

Allowed values: 0 = disabled, [bit 0]: Fine Time Interpolation [bit 1]: Energy and PSD (dual gated integrator)

Description: in MIXED mode (waveforms enabled), this processor can apply software digital algorithms to the acquired waveform and put the result in the event triplet (replacing the parameters calculated on-board by the FPGA if present).

6.3.4. RecordLength

Parameter type: unsigned integer

Allowed values: depends on the board/firmware type (see the relevant user manuals)

Description: number of samples in the waveforms; this parameter is ignored in LIST mode. NOTE: multiply by the sampling period to get the record length in ns.

6.3.5. PreTrigger

Parameter type: unsigned integer

Allowed values: depends on the board/firmware type (see the relevant user manuals)

Description: number of samples before the trigger in the waveforms; this parameter is ignored in LIST mode. NOTE: multiply by the sampling period to get the pre trigger in ns.

6.3.6. EventBuffering

Parameter type: unsigned integer

Allowed values: 0 to 255. 0 means automatic (decided by digiTES)

Description: the memory buffers in the digitizers can contain N events; one buffer (or aggregate) cannot be read out until it is complete (some digitizer models have an automatic flush after a timeout). High values of N allow for an increased readout efficiency because the size of each buffer is larger and so the payload fills better the data blocks. On the other hand, for low counting rates (< 10 Hz) it is better to set N=1 to avoid latency of the events in the internal memory. This is particularly important when the event builder must find the coincidences between channels. In general, the automatic setting (EventBuffering = 0) works fine.

6.3.7. RunNumber

Parameter type: unsigned integer

Allowed values: any positive integer or AUTO

Description: set run number; if AUTO, the run number is automatically increased after each run. The run number is used in the output file names, so that you can create a new set of files for each run.

6.4. Input Data Files

6.4.1. InputFileType

Parameter type: string

Allowed values: RAW, BINARY_LIST, ASCII_LIST

Description: defines the type of file for the off-file run. RAW is a single file (saved with the option SaveRawData during an on-line run) that contain the whole information read from the digitizers; the LIST (either ASCII or binary) contains the triplet T, E, S of a specific channel; during the off-line run with list file, the data are associated to board 0, channel 0 regardless the actual physical channel from which the data have been acquired.

6.4.2. InputDataFilePath

Parameter type: string

Allowed values: any valid path (you can use both ‘\’ and ‘/’ as path separator, the SW will replace it with the correct one)

Description: Path to the input file for off-line runs. By default, this path is the same as the output files (*DataFilePath*); take care of not overwriting input and output files if you use the same run number.

6.4.3. InputDataFileName

Parameter type: string

Allowed values: any valid file name

Description: when set, defines the Input File name for off-line runs (data from input files instead of from digitizers). Don’t put this parameter in the config file (or comment it out) to run on-line.

6.4.4. LoopInputFile

Parameter type: bit

Allowed values: 0, 1

Description: enables cycling over the data contained in the input data file; when the end of file is reached, the program restarts reading data from the beginning. This mode is mainly used to test the program and make demos because it allows a continuous run cycling a few data set. Of course, the results and the statistics doesn’t have any sense for the physics.

6.5. Output Data Files

6.5.1. SaveLists

Parameter type: unsigned integer (3 bits)

Allowed values: 0 to 7: bit[0]=individual channel list, bit[1]=merged list, bit[2]=built events list

Description: enables lists saving to output files. There are 3 types of list: individual channel list (one file per channel); merged file list (one file for all channels); built events list (one file per group of channels, e.g. crystals of a clover). The file name of the individual list is *RunX_List_Brd_Ch.txt* (or .dat for binary).

6.5.2. SaveRawData

Parameter type: bit

Allowed values: 0, 1

Description: enables the raw data saving to output files. There is one single output file for the raw data and it is always binary. It contains the complete event information as available after the preprocess. The filename is *RunX_raw.dat*. This file can be used as Input Data File for off-line runs to exactly “replay” an on-line run previously saved, with the possibility to change some parameters in the analysis (e.g. the cut ranges or the time correlation window) and see the effect on the same data set.

6.5.3. SaveHistograms

Parameter type: unsigned integer

Allowed values: 0 to 7. 0=all disabled. Bit0 = Energy, Bit1 = Timing, Bit2 = PSD

Description: defines which histograms are saved. The histogram files are always ASCII but can have 3 different formats (see *HistoOutputFormat*). The histograms are saved at the end of run or when the ‘h’ key is pressed.

6.5.4. SaveWaveforms

Parameter type: bit

Allowed values: 0, 1

Description: enables the waveform saving to output files. There is a waveform file per channel, ASCII or binary (see *OutFileFormat*), and the file name is *RunX_Wave_Brd_Ch.txt* (or .dat for binary). In the ASCII file there is one line per waveform: *uint64_t* Time Stamp in ps (that includes also the FineTimeStamp), *uint16_t* Energy, *uint16_t* N (= numSamples), then a list of N *uint16_t* samples. For the ASCII format, the columns are separated by tabs; in the binary file there is no separation between columns and lines. The events saved to the waveform file are those after the selector; if you want to save all of the acquired events, you must disable the correlation and the cuts.

The waveform files cannot be used as Input Data File for off-line runs.

6.5.5. HistoOutputFormat

Parameter type: string

Allowed values: ASCII_1COL, ASCII_2COL, ANSI_42

Description: file format for the histograms (spectra): ASCII_1COL is a text file with a single column (counts); ASCII_2COL includes also the channel number, ANSI-42 is a standard for spectroscopy output files (used by MC2Analyzer) that includes also further information such as parameters, date, operator, etc. but none of them are actually used in digITES. It just uses a fixed template.

6.5.6. OutFileFormat

Parameter type: string

Allowed values: ASCII, BINARY

Description: file format for the LIST and WAVEFORM files; see SaveWaveform and SaveList options for more details on the data format.

6.5.7. OutFileTimeStampUnit

Parameter type: unsigned integer

Allowed values: 0 = ps (default), 1 = ns, 2 = us, 3 = ms, 4 = s

Description: by default, the time stamp in the list files is expressed in *ps* as a 64 bit unsigned integer. By this option, it is possible to express it as a *double* with different time units (ns, us, ms or s)

6.5.8. DataFilePath

Parameter type: string

Allowed values: any valid path name (you can use both ‘\’ and ‘/’ as path separator, the SW will replace it with the correct one)

Description: path to the folder where output and input files are saved and loaded.

6.5.9. HeaderInListFiles

Parameter type: bit

Allowed values: 0, 1

Description: when 1, a header is added in the list files

6.5.10. ConfirmFileOverwrite

Parameter type: bit

Allowed values: 0, 1

Description: when set (=1), the program asks the user to confirm if the output file already present can be overwritten

6.6. Sync and Triggers

6.6.1. FPIOtype

Parameter type: string

Allowed values: NIM, TTL

Description: defines the electrical level of the front panel LEMO I/Os (TRGIN, TRGOUT/GPO, SIN/GPI).

6.6.2. StartMode

Parameter type: string

Allowed values: see the list in the description

Description: defines the way digiTES manages the start of the acquisition in the digitizers:

- **INDEP_SW:** the start is given by a software command to each board individually. In this mode, no external connection between the boards is required, but the boards won't be synchronized because there is no control on the exact time the run starts in each board.
- **SYNCIN_1ST_SW:** this mode configures the 1st board in the chain (defined as [BOARD 0] in the config file) to have a software controlled start/stop and the other boards (from the 2nd to the last one) to have SIN controlled start/stop (SIN=1 → run; SIN=0 → stop). The boards must be connected in daisy chain using the TRGOUT-SIN connectors. To start the acquisition synchronously, the software arms all boards, then starts the run of the 1st one by a software command. The TRGOUT of the 1st board goes high, and so the SIN of the 2nd board that starts the run and so on until the last board in the chain. When the software stops the run in the 1st board, the daisy chain TRGOUT/SIN goes low and stops the run in all of the boards synchronously.
The propagation delay of the SIN/TRGOUT chain causes a fixed offset in the time stamps; this is compensated by the “run delay” setting that is automatically managed by digiTES; it sets a different run delay board by board to compensate for the external delay. Of course, it is necessary to synchronize also the clocks of the digitizers to guarantee that the time stamps don't drift. **WARNING:** in this mode, it is necessary to set SyncinMode = RUN_CTRL.
- **SYNCIN_1ST_HW:** this mode is similar to SYNCIN_1ST_SW with the difference that also the 1st board is programmed to have SIN controlled start/stop, so it is possible to control the run with an external signal connected to the SIN of the 1st board. The software arms the boards, then waits for the external signal going up to start the data readout. If the external signal goes down, the boards stop the acquisition (RUN led on the front panel going off), but the software continues to read data; at some point, it will find all the boards empty. If the external signal returns high, the acquisition restarts and the software takes new data. The time stamp is cleared at the start of each run, so you can see where the run was stopped and restarted in the acquired lists. This is a way to have hardware controlled multiple runs within a single run in the software. **WARNING:** in this mode, it is necessary to set SyncinMode = RUN_CTRL.
- **TRGIN_1ST_SW:** this mode uses the TRGIN/TRGOUT chain instead of the SIN/TRGOUT chain described above. The 1st trigger is used to start the acquisition. To start the run, the software arms all boards, then sends a software trigger to the 1st board in the chain; this trigger pulse is propagated through the chain to start all boards synchronously. The advantage of the TRGIN/TRGOUT is that it is possible to use it also to propagate acquisition triggers or vetoes after the 1st trigger is passed. The disadvantage is that it is not possible to stop the acquisition synchronously; indeed, the stop of the run is controlled by the software sending commands board by board.
- **TRGIN_1ST_HW:** same as TRGIN_1ST_SW but now the start of run comes from an external signal (trigger pulse to TRGIN of the 1st board).

6.6.3. SyncinMode

Parameter type: string

Allowed values: see the list in the description

Description: defines how the boards use the SIN (or GPI for Desktop/NIM modules) input.

- **DISABLED:** SIN is not used
- **TSTAMP_RESET:** a positive pulse into SIN (rising edge) clears the time stamp in the board, without affecting the acquisition
- **RUN_CTRL:** SIN is used to start/stop the acquisition (see parameter StartMode at par. 6.6.2)

6.6.4. TrginMode

Parameter type: string

Allowed values: see the list in the description

Description: defines how the boards use the TRGIN input.

- **DISABLED:** TRGIN is not used
- **EXTTRG_ONLY:** TRGIN is only used to send a common trigger to the board, that is a simultaneous trigger to all channels in the board. This common trigger is ORed with the individual self-trigger of each channel, so the name “EXTTRG_ONLY” doesn’t mean that the board is taking only the external trigger, it means that the TRGIN input is only used as a common trigger. If you want to disable the individual self-triggers and use only the external common trigger, you must disable the digital discriminator (DiscrMode = DISABLED) in all channels. If you have multiple boards, you can propagate the common trigger from board to board through the TRGIN/TRGOUT daisy chain (you must set TrgoutMode = PROP_TRGIN)
- **EXTTRG_START:** TRGIN is used to send common triggers to the board as well as to start the acquisition on the 1st trigger after the board have been armed (see StartMode)
- **VETO:** TRGIN acts as a VETO for the self-triggers of the channels: if the veto is active, the self-triggers are ignored. There are two operating modes: if VetoWindow = 0, then TRGIN is “level sensitive” and the veto remains active as long as the TRGIN is high, otherwise TRGIN is “edge sensitive” and the veto starts with the leading edge of TRGIN and remains active for the time set by VetoWindow. NOTE: the VetoWindow is currently implemented in the x730/x725 models only.
- **GATE:** same as VETO but with reversed polarity: the self-triggers are only accepted when the signal is active. The VetoWindow works in the same way also for the GATE mode.
- **COINC:** the TRGIN signal is used by the channels to make a coincidence within the CoincWindow. If the TRGIN signal (leading edge) arrives before/after the self-trigger with a distance smaller than the coincidence window, the self-trigger is accepted.

6.6.5. VetoWindow

Parameter type: unsigned integer

Allowed values: from 0 to (???)

Description: defines the duration (in ns) of the Veto window. The VETO has different sources: it can come from the external trigger (TRGIN) when TrginMode = VETO as well as from some internal conditions, such as the input saturation, but they are not documented here. In any case, if the Veto Window is zero, then the veto remains active as long as the source is high, otherwise the veto starts with the leading edge of the source and remains active for the time set by VetoWindow. NOTE: the VetoWindow is currently implemented in the x730/x725 models only.

6.6.6. TrgoutMode

Parameter type: string

Allowed values: see the list in the description

Description: defines how the boards use the TRGOUT output.

- **DISABLED:** TRGOUT is always low
- **CHANNEL_TRIGGERS:** logic masked OR of the channel trigger outputs; the mask is set by the parameter TrgoutMask. The width of each trigger output feeding the OR is equal to the parameter CoincWindow for the DPP firmware, while it corresponds to the time the signal stays above the threshold for the Standard Firmware.
- **PROP_TRGIN:** propagate the trgin signal.
- **SYNC_OUT:** this option is required for multi-board synchronization (see StartMode). It applies to both SIN/TRGOUT and TRGIN/TRGOUT chains. In the former case, TRGOUT propagates the run status (TRGOUT=1 when the board is running, 0 when it is stopped). In the latter case, TRGOUT propagates the trigger pulses, including the 1st trigger used to start the acquisition.
- **SQR_WAVE_1KHZ:** it generates a 1 KHz square wave
- **PULSES_1KHZ:** it generates a 1 KHz pulses with 8 ns width
- **SQR_WAVE_10KHZ:** it generates a 10 KHz square wave
- **PULSES_10KHZ:** it generates a 10 KHz pulses with 8 ns width

6.6.7. TrgoutMask

Parameter type: unsigned hexadecimal

Allowed values: 0x00 to 0xFF

Description: when TrgoutMode = TRIGGERS, the TRGOUT (GPO in the desktop models) connector outputs the masked OR of the channel self-trigger. The trgout enable mask is set by this parameter; it doesn’t affect the input channel enable mask for the acquisition.

6.7. Coincidences in hardware

6.7.1. CoincMode

Parameter type: string

Allowed values: see the list in the description

Description: defines HW settings for the "on-board" coincidence and trigger propagation logic (between the channels of the board). It is applied equal to all of the boards (no board to board coincidence is managed here).

- **DISABLED:** no HW coincidence
- **MAJORITY:** all channels are validated when there are at least N channels firing in the coincidence window, where N is defined by the parameter MajorityLevel
- **MINORITY:** all channels are validated when there are less than N channels firing in the coincidence window, where N is defined by the parameter MajorityLevel
- **PAIRED_AND:** coincidence between couples (0 and 1, 2 and 3, etc...)
- **ANTI_PAIRED_AND:** anti-coincidence between couples (0 and 1, 2 and 3, etc...)
- **PAIRED_OR:** trigger propagation within the couples: each channel in the couple acquires when one of the two has a self-trigger.
- **COMMON_REFCH:** all channels go in coincidence with one reference channel, typically for TOF measurements with a common start and multiple stops (NOTE: for the moment, the ref channel is hardcoded to channel zero)
- **ANTI_COMMON_REFCH:** all channels go in anti-coincidence with one reference channel (NOTE: for the moment, the ref channel is hardcoded to channel zero)
- **AND_ALL:** all the enabled channels must be in coincidence
- **OR_ALL:** trigger propagation from any to all. All channels in the board acquire when there is at least a self-trigger in one channel.
- **CH0_TO_ALL:** trigger propagation from channel 0 to all. All channels in the board acquire when there is a self-trigger in channel 0.

6.7.2. MajorityLevel

Parameter type: unsigned integer

Allowed values: 1 to 7

Description: minimum number of channel that must fire within the coincidence window to have the validation of the coincidence logic. NOTE: in the x730 and x725 boards, this number refers to the couples and not to the single channels; for instance, if MajorityLevel = 2 and channels 0 and 2 fire, then the coincidence is valid (two different couples firing); if channel 0 and 1 fire, the coincidence is not valid (one couple only).

6.7.3. CoincWindow

Parameter type: unsigned integer

Allowed values: any positive

Description: defines the coincidence window (in ns). Note that the CoincWindow is a hardware setting (register of the digitizer) while the parameter TimeCorrelWindow is used in the software event correlation. The two parameters can be different.

6.8. Input signals

6.8.1. EnableInput

Parameter type: bit

Allowed values: 0, 1

Description: enables one input channel for the acquisition. Typically, in the config file there is a default setting (in the [COMMON] section) to keep all channels enabled, then each channel can be individually disabled (or the other way around).

6.8.2. PulsePolarity

Parameter type: string

Allowed values: NEGATIVE, POSITIVE

Description: defines the polarity of the input pulses; notice that in the DPP_PHA algorithms, negative pulses are inverted at the input of the FPGA and so they always appear as positive pulses. In the DPP_PSD the signal is instead plotted with the actual polarity.

6.8.3. BaselineDCOffset

Parameter type: float

Allowed values: 0 to 100

Description: the analog input stage of the digitizers has a 16 bit DAC that generates a DC level to be added to the input signal, so that it is possible to move the baseline of the signal over the full input dynamic range of the ADC. The value of this parameter defines the position of the baseline in percent of the full scale, taking into account the pulse polarity. For instance, if you set BaselineDCOffset = 10, the baseline will be at about 10% of the FSR for a positive signal and about 90% for a negative one. Because of the tolerance of the components, the actual position can be significantly different from the theoretical value, so it is recommended to make a check the position in oscilloscope mode. It may happen that the digitizer doesn't trigger because the value of the DC offset is wrong and the baseline is out of range; in this case, we suggest to use the software trigger (keys 't' or 'T' in the on-line menu) to force the acquisition and to plot the waveforms, so that it is possible to see the position of the baseline and check if it is out of range.

6.8.4. ZeroVoltLevel

Parameter type: unsigned integer

Allowed values: 0 to $2^{N_{bit}} - 1$

Description: In general, the “zero volt” of the input signal (baseline in the analog signal) doesn’t correspond to zero ADC counts; there is an offset that is determined by the BaselineDCOffset (programmable DAC voltage summed to the input signal). The DPP firmware is able to calculate the baseline of the input and subtract it from the signal automatically, so there is no need to set this parameter. However, there are cases where the firmware is not able to do this:

- Standard FW (non DPP)
- PHA for 725/730 models when LED or CFD discriminators are used; in fact, the PHA calculates the baseline of the trapezoid, but not that one of the input.

In such cases, the user must check the position of the baseline (in oscilloscope mode) and set this parameter (channel by channel) accordingly. This allows, for instance, the leading edge discriminator of the standard firmware to have a trigger threshold relative to the baseline and not to the absolute ADC count.

6.8.5. InputDynamicRange

Parameter type: unsigned integer

Allowed values: 0, 1 for x725, x730; 0 to 3 for x780/x781

Description: some digitizers have the possibility to change the gain (and so the full scale range) in the analog input stage; this acts as a coarse gain in the acquisition of the energy. It is worth noticing that some digitizers (e.g. the x724) have different purchasing options for the input range, but this is a fixed factory setting that cannot be controlled by this option. This parameter can only be applied in these cases:

x730: 0 = 2 Vpp
 1 = 0.5 Vpp
 x780: 0 = 0.6 Vpp
 1 = 1.4 Vpp
 2 = 3.7 Vpp
 3 = 9.5 Vpp

6.9. Discriminator

6.9.1. DiscrMode

Parameter type: string

Allowed values:

- **LED_PSD** (or just **LED**): Leading Edge Discriminator in DPP_PSD
- **CFD_PSD** (or just **CFD**): Constant Fraction Discriminator in DPP_PSD
- **RCCR2_PHA**: RC-RC² TT filter in the PHA (2nd derivative + smoothing)
- **CFD_PHA**: Constant Fraction Discriminator in DPP_PHA
- **LED_PHA**: Leading Edge Discriminator in DPP_PHA

Description: Set the discriminator type. This option is available for the DPP_PSD and PHA of the x730/x725 only (soon on the x751 PSD too). The LED mode uses the point where the signal crosses the trigger threshold to generate the self-trigger; in CFDF mode, the threshold is only used to arm the CFD, then the self-trigger is issued on the zero crossing of the CFD signal. Normally in CFD mode the delay between the input pulses and the self-trigger is higher, so it is important to review the timing settings associated to the trigger (e.g. the PreTrigger and the PreGate) every time the DiscrMode is toggled between LED and CFD. It is recommended to use the CFD mode for a good timing resolution (FineTimeStamp interpolation).

6.9.2. TriggerThreshold

Parameter type: unsigned integer

Allowed values: 0 to $2^{N_{bit}} - 1$, where Nbit is the number of bit if the ADC

Description: Trigger Threshold in ADC counts. The table below shows approximately how much is an ADC count in voltage:

- | | |
|-------------------------------------|----------|
| • x724 (standard model): | 0.14 mV |
| • x724 (customized to 0.5Vpp): | 0.03 mV |
| • x751/x761: | 1 mV |
| • x720 | 0.5 mV |
| • x730/x725 (InputDynamicRange = 0) | 0.12 mV |
| • x730/x725 (InputDynamicRange = 1) | 0.03 mV |
| • x780 (InputDynamicRange = 0) | 0.036 mV |
| • x780 (InputDynamicRange = 1) | 0.085 mV |
| • x780 (InputDynamicRange = 2) | 0.22 mV |
| • x780 (InputDynamicRange = 3) | 0.58 mV |

In the DPP firmware, the threshold is referred to the baseline, so it has to be intended as the minimum amplitude of the input pulses that make the trigger to occur. Conversely, in the standard firmware the trigger threshold is an absolute value; it is therefore important to know the exact position of the baseline before setting it.

6.9.3. TrgHoldOff

Parameter type: unsigned integer

Allowed values: ? (rounded to multiples of 10 ns for the x724, x780, x781, 8 ns for the x730, x725, x751, x720)

Description: this is the protection time (in ns) that prevents the channel to retrigger after the previous trigger; it is typically used to avoid false triggers on after pulses, ringing, overshoot, etc..

6.9.4. TTFsmoothing

Parameter type: unsigned integer

Allowed values: 0 = disabled, 1 = 2 samples, 2 = 4 samples, 3 = 8 samples, 4 = 16 samples, 5 = 32 samples

DPP_PSD x730/x725: 0 to 4

DPP_PHA (all): 0 to 5

Not allowed for other firmware types

Description: the Trigger and Timing Filter (RC-CR2 in the PHA, CFD or LED in the PSD) is a fast digital filter that processes the input pulses, generates the self-trigger and applies the timing interpolation. When the signal presents high frequency noise or fast spikes, it is convenient to smooth it before processing the trigger. The smoothing is a moving averaging window of N samples, as reported in the allowed values.

6.9.5. TTFdelay

Parameter type: unsigned integer

Allowed values: ? (expressed in ns)

Description: this parameter is used in the RC-CR2 filter of the DPP_PHA (it is meaningless in the PSD) and defines the delay for the digital derivative filter. It must be set about equal to the rise time (from the baseline to the peak) of the input signals plus the smoothing factor.

6.9.6. CFDDelay

Parameter type: unsigned integer

Allowed values: ? (expressed in ns)

Description: it defines the delay of the digital CFD (DPP_PSD only). The typical setting is $Delay = (1 - f) * SignalRiseTime$, where $f = CFDFraction$

Sometimes values greater than the theoretical one give better results. See the CFD waveform to find the best shape (maximizing the slope in the zero crossing).

6.9.7. CFDFraction

Parameter type: unsigned integer

Allowed values: 0 = 25%, 1 = 50%, 2 = 75%, 3 = 100%

Description: fraction used in the digital CFD.

6.9.8. EnableZCcal

Parameter type: bit

Allowed values: 0 = disabled, 1 = enabled

Description: the zero crossing of the CFD signal is calculated on-line by the FPGA that uses the linear interpolation between two points to increase the timing resolution beyond the sampling rate. The result of the interpolation is expressed as a 10 bit number that represents the position of the zero crossing within the sampling period. However, this algorithm is affected by systematic errors that can be corrected after a calibration based on the statistics. It is possible to make a calibration run using the -zcc option on the command line or pressing 'z' after the start of the run to restart in calibration mode. During the calibration run, the program acquires a given number of events for each channel (the percent of completion is showed in the console) and calculates the calibration table. This is saved to the file `zcc.dat` when quitting. If `EnableZCcal = 1` and the calibration file is present, the calibration table is used by the program to correct the ZC error. Only the channels having the relevant table in the `zcc.dat` file are calibrated. If `EnableZCcal = 0`, the calibration is disabled; however, it is possible to make a calibration run and generate the `zcc.dat` file. **WARNING:** every time a calibration run is performed, the `zcc.dat` file is overwritten, without any warning message. Please rename the file if you don't want to lose it. Otherwise, you can change the file name in the file `sysvars.txt` (system variables).

6.10. Gated Charge Integration (DPP PSD/CI)

6.10.1. GateWidth

Parameter type: unsigned integer

Allowed values: range depends on firmware type; expressed in ns

Description: gate width for the total charge integration (= Energy) in the DPP_PSD and DPP_CI. This parameter is also used by the WaveformProcessor for the software DPP.

6.10.2. ShortGateWidth

Parameter type: unsigned integer

Allowed values: range depends on firmware type; expressed in ns

Description: short gate width for the charge integration of the fast component in the DPP_PSD (not used by the DPP_CI). The parameter PSD in the event data is calculated by the software as $PSD = (Q_{LONG} - Q_{SHORT})/Q_{LONG}$. This parameter is also used by the WaveformProcessor for the software DPP.

6.10.3. PreGate

Parameter type: unsigned integer

Allowed values: range depends on firmware type; expressed in ns

Description: portion of the gate (both short and long) before the trigger.

6.10.4. ChargeSensitivity

Parameter type: unsigned integer

Allowed values: 0 to 4 for x730/x725, 0 to 5 for x751, 0 to 3 for x720

Description: the digital gated integrator makes the sum of the samples (baseline subtracted) within the integration gate; this sum is accumulated over a given number of bit (typically 24); however it doesn't make sense to provide the result over so many bits. Depending on the signal amplitude and duration, we can define a dividing factor to apply to the sum before the charge is put in the output data. Therefore, this parameter defines the conversion between 1 LSB (or channel in the energy value) and the charge expressed in fC and acts as a coarse gain in the energy spectrum. Starting from the rev. 4.2.6 of digiTES, the ChargeSensitivity is actually automatically set by the software according to the *EnergyCoarseGain* setting in the energy histogram (this parameter is more familiar to most users), so it is recommended not to set the ChargeSensitivity. However, it is still possible to force a specific ChargeSensitivity just keeping the explicit setting in the configuration file; in this case, the EnergyCoarseGain will be ignored. See par. 6.12.2 for the correspondence table between EnergyCoarseGain and ChargeSensitivity.

The table below reports the charge sensitivity expressed in fC for the different models and input ranges.

Sensitivity	x730 - 2Vpp	x730 - 0.5Vpp	x725 - 2Vpp	x725 - 0.5Vpp	x751 - 1Vpp	x720 - 2Vpp
0	5	1.25	10	2.5	20	40
1	20	5	40	10	40	160
2	80	20	160	40	80	640
3	320	80	640	160	160	2560
4	1280	320	2560	640	320	not allowed
5	not allowed	not allowed	not allowed	not allowed	640	not allowed

6.10.5. NSBaseline

Parameter type: unsigned integer

Allowed values: 0 to 4 for x725, x730 and x720; 0 to 7 for x751

Description: number of samples averaged on the input signal to calculate the baseline; this parameter applies to DPP_PSD and DPP_CI only; for TrapNSBaseline for the DPP_PHA firmware. When NSBaseline = 0, the dynamic baseline calculation is disabled and a fixed value (set by the parameter FixedBaseline) is used.

NSBaseline	x730/x725	x751	x720
0	fixed	fixed	fixed
1	16	8	8
2	64	16	32
3	256	32	128
4	1024	64	512
5	-	128	-
6	-	256	-
7	-	512	-

6.10.6. FixedBaseline

Parameter type: unsigned integer

Allowed values: 0 to $2^{N_{bit}-1}$

Description: when NSBaseline = 0, the baseline is set the a fixed value equal to the content of this register.

6.10.7. PileUpMode

Parameter type: unsigned integer

Allowed values: 0 = tagged, 1 = rejected, 2 = retrigger (x751 only)

Description: the pile-up condition occurs when there are multiple pulses within the integration gate, also in the case where the signal doesn't cross the trigger threshold more than once. Assuming a positive pulse, to FPGA rises the pile-up flag when it finds a peak-valley-peak sequence: after the trigger (1st edge of the pulse crossing the threshold), the logic searches for the 1st peak, then for a valley (signal falling down from the peak for more than a given threshold called *PurGap*) and finally for a 2nd peak (signal rising up again from the valley for more than the same threshold). If this condition is met, then the event is tagged as "pile-up". The *PileUpMode* option tells the logic what to do in this case: 0 means "tag and keep in the memory", 1 means "discard the event", 2 means "make a new event starting from the pile up", thus creating 2 separate events. The latter option is only available for the x751 digitizers.

6.10.8. PurGap

Parameter type: unsigned integer

Allowed values: 1 to 2^N-1 , with N = number of bit of the ADC

Description: threshold used to search the peak-valley-peak sequence (pile-up identification). See par. 6.10.7.

6.10.9. ChargeLLD

Parameter type: unsigned integer

Allowed values: 0 to 65535. 0 means disabled.

Description: hardware zero suppression based on collected charge ($= Q_{LONG}$). Events with charge $< Q_{thr}$ are discarded (not saved to the memory board). This option is only available on DPP_PSD for x725 and x730 with Fw Rev ≥ 136.9 .

6.11. Trapezoidal Filter (DPP PHA)

6.11.1. TrapRiseTime

Parameter type: unsigned integer

Allowed values: expressed in ns; current maximum range is 10000 ($= 10 \mu s$) in PHA for x724, x780, x781, 20000 ($= 20 \mu s$) in nPHA, 8000 ($= 8 \mu s$) in PHA for x730 and 16000 ($= 16 \mu s$) in PHA for x725. Values can be increased by a factor 2, 4, 8 by decimation.

Description: Rise Time of the trapezoid; this parameters plays the same role of the shaping time in the analog spectroscopy amplifier. Typically, to have the same energy resolution, the rise time of the digital filter is set to about 2.5 times the shaping time of the analog chain.

6.11.2. TrapFlatTop

Parameter type: unsigned integer

Allowed values: expressed in ns. (range ?)

Description: Flat top of the trapezoid. Typically it is set in the range 500 ns - 1 us, but the value can be higher in case of large size detectors where the ballistic deficit due to the slow charge collection time is significant.

6.11.3. TrapPoleZero

Parameter type: unsigned integer

Allowed values: expressed in ns in the range 0 to 650 μs

Description: Pole Zero compensation of the digital trapezoidal filter; it must be set equal to the decay time constant of the pre-amplifier (Typ 50 μs).

6.11.4. PeakingTime

Parameter type: unsigned integer

Allowed values: expressed in ns. (range ?)

Description: it defines the starting point in the flat top where the peak is sampled to get the energy value. It is expressed as the time from the beginning of the flat top. Normally, the peaking time is at $\sim 80\%$ of the flat top. Check that the end of the peaking region (see NSpeak) is before the end of the flat top.

6.11.5. NSPeak

Parameter type: unsigned integer

Allowed values: 0 = 1 sample, 1 = 4 samples, 2 = 16 samples, 3 = 64 samples

Description: number of samples of the flat top that are averaged to calculate the trapezoid height (=energy).

6.11.6. PeakHoldOff

Parameter type: unsigned integer

Allowed values: expressed in ns; (range ?)

Description: defines the minimum distance that separates two consecutive flat tops (that is from the end of the 1st to the start of the 2nd) to consider them as pile-up. The minimum pulse separation for not piled-up events is approximately $TrapRiseTime + TrapFlatTop + PeakHoldOff$. To avoid border effect, it is recommended to set PeakHoldOff at least greater than 100ns. To guarantee a "fresh" baseline recalculated for each trapezoid, it is necessary to set $PeakHoldOff > (TrapRiseTime + BaselineEveragingTime)$, where BaselineEveragingTime is given by the number of samples used for the baseline (see TrapNSBaseline) multiplied by the sampling period. In cases where the counting rate is very high and it is necessary to get a good energy resolution, it is recommended to increase PeakHoldOff to at least $2 * TrapRiseTime$.

6.11.7. TrapNSBaseline

Parameter type: unsigned integer

Allowed values: 0=fixed, 1=16, 2=64, 3=256, 4=1024, 5=4096, 6=16384

Description: defines the number of samples in the moving window used to calculate the baseline of the trapezoid.

6.11.8. Decimation

Parameter type: unsigned integer

Allowed values: 0 to 3, being the decimation equal to the relevant power of two (that is 1, 2, 4, 8)

Description: in the DPP_PHA firmware it is possible to apply a decimation to the input signal. This consists in averaging a certain number of samples to make one sample at lower frequency but higher resolution. For instance, if you make a decimation of 4 in the x724, the sampling rate becomes 25 MHz (instead of 100). Besides the reduction of the noise and improvement of the effective number of bits, the decimation makes the range of the timing parameters (mainly trapezoidal rise time and flat top) N times bigger, where N is the decimation.

6.12. Energy Spectrum

6.12.1. EHnbin

Parameter type: unsigned integer or string

Allowed values: any power of 2 in the range 256 - 32K; strings 1K, 2K, 4K, 8K, 16K and 32K are also accepted

Description: number of channels (bins) in the energy spectrum

6.12.2. EnergyCoarseGain

Parameter type: integer and fractions (as float or strings)

Allowed values: 1, 2, 4, 8, 16, 32; attenuations: "1/16" or 0.0625, "1/8" or 0.125, "1/4" or 0.25, "1/2" or 0.5.

Description: the energy gain is a multiplying factor (or even attenuation) that is applied by the algorithm that calculates the energy, either in PHA (trapezoid height) or PSD/CI (gated integrator). It is a pure digital gain and has no effect on the A/D conversion and input dynamic range of the analog signal. Both PHA and PSD algorithms use accumulators to sum a series of samples and obtain the energy value represented over a large number of bit; the final energy information transferred from the board to the computer is then rescaled to a given number of bits by a shift and truncate operation. The Coarse Gain takes place at this stage and it determines the weight of the bits used while rescaling the energy. In addition to that, the software can add one more energy divider (Ediv) to fill the gaps in the available options of the firmware. In the PSD or CI firmware, the EnergyCoarseGain takes the place of the ChargeSensitivity when this is not specifically set.

The aim of the Coarse Gain is to hide the different mechanisms that regulate the calculation of the energy in the firmware and have an unique, intuitive parameter that controls the X axis of the energy spectrum, acting as a gain factor in the conversion from KeV to channels and allowing the user to expand or shrink the spectrum.

The table below reports, for different boards, the correspondence between Coarse Gain and the ChargeSensitivity (and Ediv) setting as well as the position (in channels) of the 662 KeV of the Cs127 source for typical NaI and LaBr3 detectors.

[illegible]

6.12.3. EnergyFineGain

Parameter type: float

Allowed values: any positive float

Description: the fine gain operates in combination with the coarse gain; it is typically used to precisely expand the energy spectrum and bring the centroid of an energy peak into a given channel. This is very useful when you want to compare (overlap) two different spectra, channel by channel. In the PHA firmware, the fine gain is applied in the trapezoidal filter (in the FPGA) while in the PSD and CI firmware it is applied in the software.

6.12.4. ECalibration

Parameter type: 2, 3, or 4 float numbers

Allowed values: any value

Description: The syntax is ECalibration c0 c1 c2 c3 (c2 and c3 are optional). These are the parameters for the energy calibration (conversion from channels to keV): $E_{keV} = c_3 * E_{ch}^3 + c_2 * E_{ch}^2 + c_1 * E_{ch} + c_0$.

6.12.5. EnableEnergyFilter

Parameter type: bit

Allowed values: 0 = disabled, 1 = enabled

Description: this filter (fully implemented in the software) applies an *energy gating* to the events read from the boards, removing from the queues the events whose Energy value is not between the lower and the upper threshold (LCut and UCut). It doesn't affect the acquisition at board level.

6.12.6. EnergyLCut

Parameter type: unsigned integer

Allowed values: 0 to 32K

Description: Lower Threshold for the Energy Filter (energy gating); when the filter is enabled, events with Energy < EnergyLCut will be discarded from the queues.

6.12.7. EnergyUCut

Parameter type: unsigned integer

Allowed values: 0 to 32K

Description: Upper Threshold for the Energy Filter (energy gating); when the filter is enabled, events with Energy > EnergyUCut will be discarded from the queues.

6.12.8. EnableEnergySkim

Parameter type: bit

Allowed values: 0=disabled, 1=enabled

Description: this parameter is available for the DPP_nPHA only (DPP_PHA of the x724/x780/x781 with minor number > 64). It operates at hardware level, similarly to the Energy Filter described above but with the advantage that the selection is done before reading the events out from the board, thus reducing the data throughput. It uses that same thresholds of the SW energy filter. This filter is mainly intended for debugging, for instance when there is an unexpected peak or structure in the energy spectrum and the user wants to see the waveforms that generate those energies.

6.13. Time Spectrum

6.13.1. TspectrumMode

Parameter type: string

Allowed values: INTERVALS, START_STOP

Description: there are two different types of time spectrum: INTERVALS is the distribution of the deltaT between two consecutive events in the same channel; typically this timing distribution is a poissonian. It is very useful to analyze this histogram because it tells you the average counting rate ($= 1/\text{Tau}$, being Tau the time constant of the exponential) as well as the minimum distance between events, corresponding to the initial gap in the poissonian curve. This is a way to estimate the dead time in the event acquisition. Do not confuse this dead time with that one of the PHA firmware that takes into account also the dead time due to the pile-up of the trapezoids. Typically, in INTERVALS mode, THmin=0 and THmax is quite high (ms or even seconds if the expected rate is low).

The START_STOP mode is instead the measurement of the time between a pulse in the common reference channel (defined by the TOFstartChannel/Board) and the pulses in the other channels. Sometimes this is called TAC spectrum. The histogram can include also negative values (that is a stop arrived before the common start). THmin and THmax define the range for the start-stop measurement and are typically set equal to +/- TimeCorrelWindow.

6.13.2. THnbin

Parameter type: unsigned integer or string

Allowed values: any power of 2 in the range 256 - 32K; strings 1K, 2K, 4K, 8K, 16K and 32K are also accepted

Description: number of channels (bins) in the time spectrum

6.13.3. THmin

Parameter type: float

Allowed values: any value (expressed in ns)

Description: minimum value of the timing spectrum. Values smaller than THmin won't be histogrammed.

6.13.4. THmax

Parameter type: float

Allowed values: any value (expressed in ns)

Description: maximum value of the timing spectrum. Values greater than THmax won't be histogrammed.

6.13.5. TOFstartChannel

Parameter type: unsigned integer

Allowed values: 0 to 15

Description: identifies the channel used for the one-to-all coincidence applied in the software (when TimeCorrelFilter is enabled) and for the Start-Stop spectra (deltaT). This channel becomes the common start for the other channels.

6.13.6. TOFstartBoard

Parameter type: unsigned integer

Allowed values: 0 to 7

Description: identifies the board where TOFstartChannel is taken (in case of multiple boards)

6.13.7. TstampOffset

Parameter type: integer

Allowed values: any value in ns

Description: Adds a delay to the time stamps of the channel. Can be positive or negative. This parameter can be used to compensate for the cable delays in the Start-Stop measurements.

6.13.8. TimeCorrelWindow

Parameter type: positive float

Allowed values: any time expressed in ns.

Description: coincidence window used to build events in the correlated mode (CHREF_AND_ANYOTHER, CLOVER)

6.14. PSD Spectrum

6.14.1. EnablePSDFilter

Parameter type: bit

Allowed values: 0 = disabled, 1 = enabled

Description: this filter (fully implemented in the software) applies a *PSD gating* to the events read from the boards, removing from the queues the events whose PSD value is not between the lower and the upper threshold (LCut and UCut). It doesn't affect the acquisition at board level.

6.14.2. PsdLCut

Parameter type: float

Allowed values: 0.0 to 1.0

Description: Lower Threshold for the PSD Filter (PSD gating); when the filter is enabled, events with $PSD < PsdLCut$ will be discarded from the queues.

6.14.3. PsdULD

Parameter type: float

Allowed values: 0.0 to 1.0

Description: Upper Threshold for the PSD Filter (PSD gating); when the filter is enabled, events with $PSD > PsdUCut$ will be discarded from the queues.

6.14.4. ScatterPlotMode

Parameter type: string

Allowed values: PSD_HORIZONTAL, PSD_DIAGONAL, E_VS_DELTAE

Description: 2D scatter plot:

PSD_HORIZONTAL: X-axis = Total energy = Q_{LONG} , Y-axis = $PSD = (Q_{LONG} - Q_{SHORT}) / Q_{LONG}$

PSD_DIAGONAL: X-axis = Total energy = Q_{LONG} , Y-axis = Fast component = Q_{SHORT}

E_VS_DELTAE: X-axis = Energy even channels, Y-axis = Energy odd channels (delta E detector)

6.15. MCS Spectrum

6.15.1. MCSHnbin

Parameter type: unsigned integer or string

Allowed values: any power of 2 in the range 256 - 32K; strings 1K, 2K, 4K, 8K, 16K and 32K are also accepted

Description: number of channels (bins) in the MCS spectrum

6.15.2. DwellTime

Parameter type: unsigned integer

Allowed values: any positive value (expressed in μs)

Description: time interval for the MCS spectrum. The N^{th} bin of the MCS histogram represents the number of events counted in the time interval that goes from $N * DwellTime$ to $(N+1) * DwellTime$. The MCS histogram starts to be filled at the start of the acquisition (when time stamp = 0) and ends after $MCSHnbin * DwellTime \mu s$. For the moment, the only way to restart it is to stop and start the acquisition.

6.16. Stop Criteria

6.16.1. StopOnTime

Parameter type: unsigned integer

Allowed values: any positive value (expressed in ms); 0 means "no stop"

Description: the acquisition in one channel stops when the time stamp of the events (real time) from that channel reaches the given stop time.

6.16.2. StopOnLiveTime

Parameter type: unsigned integer

Allowed values: any positive value (expressed in ms); 0 means "no stop"

Description: the acquisition in one channel stops when the calculated live time in that channel reaches the given stop time.

6.16.3. StopOnTotalEvents

Parameter type: unsigned integer

Allowed values: any positive value; 0 means "no stop"

Description: the acquisition in one channel stops when the total number of events (after the filters) accumulated from that channel reaches the given threshold.

6.16.4. StopOnEnergyEvents

Parameter type: unsigned integer

Allowed values: any positive value; 0 means "no stop"

Description: the acquisition in one channel stops when the number of events accumulated in the energy spectrum from that channel reaches the given threshold.

6.16.5. StopOnTimeEvents

Parameter type: unsigned integer

Allowed values: any positive value; 0 means “no stop”

Description: the acquisition in one channel stops when the number of events accumulated in the timing spectrum from that channel reaches the given threshold.

7. Command line options

Syntax of the command line:

`digitES [-t] [-i] [-l] [-rRunNumber] [-fInputFileName] [ConfigFileName]`

<code>[-t]</code>	Test Connection: the program tries to open the digitizers as defined in the configuration file, writes a <i>connect log file</i> with the board info (or failure note), then exits.
<code>[-i]</code>	Immediate Start: the run starts without waiting the key 's' pressed by the user
<code>[-l]</code>	Load Runtime settings: the programs retrieve some run time settings from a saved file before starting the acquisition (for instance, the traces settings, the plotted channel, etc...)
<code>[-h]</code>	Help
<code>[-rRunNumber]</code>	Assigns the value RunNumber to the run being launched
<code>[-fInputFileName]</code>	Off-line run using a data file with the given name
<code>[ConfigFileName]</code>	Use the given configuration file instead of the default <i>digitES_Config.txt</i> .

8. Online Keyboard Menu

Once started, the program digitES opens and configures the digitizers, prints some information related to the connected boards, then waits for commands from the keyboard. You must activate the console window to send commands to digitES from the keyboard (typically the focus is on the *gnuplot* windows).

Below the list of commands (bindkeys) in the revision 4.3.0 of digitES; it is not guaranteed that future versions of the program maintain the same bindkeys, so please check them in the source code. You can see this list also on-line by pressing the space bar during the acquisition.

1-6) Change Probe settings for oscilloscope traces

The waveform plot can show up to 6 traces, two analog and 4 digital. Depending on the firmware and board type, each trace can represent a different signal. Pressing the key from 1 to 6 you can select the options available for the relevant trace or switch the trace off (option 'x'). NOTE: when two analog traces coming from the board are active (dual trace mode), there is a down-sampling of 2. You must switch the trace 2 off to have the full sampling rate (so a better waveform resolution) in trace 1. The down-sampling doesn't affect the energy and timing resolution, since it involves the waveform readout only and not the internal DPP algorithms.

b-c) Choose board/channel to plot

Only one channel at time is plotted. You can choose which one by pressing 'b' for board and 'c' for channel.

B) Print Board Info

Prints some board info (for the board being plotted, see command 'b').

C) Open/Save config files

Choose and load a config file from a list of preset or save the current config file with a given name and add it to the list of preset.

d) DeltaT Histogram plot

Plot the deltaT (Start-Stop measurement) histogram. NOTE: for the reference channel (TOFStartChannel) the timing plot represents the statistics of intervals between the pulses (typically a poissonian).

D) Enable/Disable Raw Data dump to files

If the parameter *SaveRawData* is zero, the raw data file is not saved to the disk. It is possible to start (and stop) raw data file saving during the run by pressing 'D'.

e) Energy Histogram plot

Plots the energy spectrum.

E) Edit config file

Open the current config file with a text editor (notepad for Windows and gedit for Linux)

f) Enable/Disable automatic refresh (plots and logs)

Plots and statistic display can slow down the acquisition. You can “freeze” them by pressing ‘f’. Press ‘f’ again to restart. ‘f’ in conjunction with ‘o’ (see below) allows the user to keep the plot still and advance one by one on keyboard command.

F) Plot FFT. Press 'F' again to change FFT mean\n")

Change the waveform plot into FFT mode. It is possible to average n FFT curves to increase the definition of the plot; to do this, press ‘F’ again and set the number N for the mean.

g) Enable/Disable internal pulse emulator

h) Save Histograms to files (one shot)

Saves the spectra files (those that are enabled by the parameter *SaveHistogram*). NOTE: spectra are also saved at the end of the acquisition.

H) HV setting

Open a manual controller for the HV channels (in DT5780, DT5790 and Hexagon)

i) Toggle statistics mode (integral/instantaneous)

Toggle between integral and instantaneous mode for the statistics (rates, event counter, etc...). In integral mode, the statistics are averaged and accumulated from the start of acquisition; in instantaneous mode they are re-calculated every T, where T is 1 second by default, but can be changed by the parameter ‘StatUpdateTime’ or by the option ‘u’ of this menu.

I) Enter run description

Enter a text string that will be saved in the run_info file.

j) Sample histogram plot

Plot the histogram of the raw ADC samples.

l) Enable/Disable List dump to files

If the parameter *SaveList* is zero, the list files are not saved to the disk. It is possible to start (and stop) list file saving during the run by pressing ‘l’.

m) MCS Histogram plot

Plot the MCS histogram.

M) Manual Controller

Opens the Manual Controller for the register access to the boards.

o) One shot refresh (plots and logs)

When the plots and logs are off (see ‘f’ option above), you can force one single shot for the plots and statistics.

p) PSD Histogram plot

Plots the PSD histogram.

P) Energy-PSD scatter 2D-plot

Plots the 3-D scatter plot with Energy on the X-axis (total charge), PSD on the Y-axis and counts on the Z-axis (represented with a color scale).

q, Q) Quit

Quits the program.

R) Reset Histograms and Statistics

Resets all the accumulated histograms (spectra) and statistics.

s) Start acquisition

Start/Stop the acquisition.

S) Stop acquisition

Restart the acquisition with a new run number and description.

t) Force trigger (one shot)

Sends one software trigger to all of the boards, thus forcing the acquisition of one event.

T) Force trigger (enable/disable continuous)

Sends continuous software triggers to all of the boards, thus forcing the acquisition of events even if no signal is present at the input. Press 'T' again to disable the continuous trigger.

u) Change Update Time for the statistics

By default, the statistics are updated every second; press 'u' to change the updating time and make the statistics more stable and precise, especially at low counting rate.

w) Enable/Disable Waveform plot (switching between List-Mixed modes)

If the acquisition started in List mode (no waveform) you can restart a new acquisition in Mixed mode (thus having the waveforms) by pressing 'w'. Likewise, you can stop the waveforms and pass to List Mode when Mixed mode was enabled.

x) Toggle between Channels and Units in the spectra

If the parameters for the energy calibration are set (m and q), it is possible to toggle between channels and KeV on the X-axis of the energy spectrum.

X) Close histogram plot**z) Start a calibration run**

Restart a new run and accumulate statistics to calculate the values for the zero crossing calibration table. It is necessary to accumulate a given number (read from the `sysvars.txt` file) of events for each channel; during the run, the percent of completion is displayed. When 100% is reached, the acquisition can be closed and the calibration tables are saved to the disk. It is also possible to stop earlier: the calibration is saved anyway, but it may be less precise.

Tab) Change Statistics

Toggle between different options for the statistics displayed in the console window.

Space) Print Menu

Prints the on-line keyboard menu.

Besides the digiTES commands (console window), there are some bindkeys in gnuplot. These are the main ones:

- | | |
|----------|--|
| a | Autoscale on both X and Y (zoom full) |
| l | Toggle between log and linear scale on Y axis (notice that if the current zoom includes Y=0, the log scale is not activated; in this case, press 'a' to autoscale) |
| r | enable/disable ruler (X-Y cursors) |
| y | Autoscale on Y axis only (leave X as is) |
| p | previous zoom |

The zoom in gnuplot is done by selecting a window with the mouse: right click on one corner of the wanted window, release the mouse button, move the mouse pointer to the opposite corner, left click.