

User Manual UM1935

CAENDigitizer Library

Library of functions for CAEN Digitizers high level
management

Rev. 20 - May 2nd, 2019

Purpose of this User Manual

This User Manual contains the full description of the C version of CAENDigitizer library rel. 2.13.1

For future release compatibility, check in the relevant library Release Notes file.

Change Document Record

Date	Revision	Changes
Feb 16 th , 2012	01	Fully revised and implemented Chap. DPP Specific Functions .
May 5 th , 2012	02	Fully revised to document the software library 2.2.1
Oct 8 th , 2012	03	Removed LabVIEW content
Oct 12 th , 2012	04	Revised functions at pp. 73 – 73
May 8 th , 2013	05	Revised DPP-CI and DPP-PSD digital probes
Mar 7 th , 2014	06	Added preliminary support to 743 digitizer series; added support to 730 digitizer series, to DT5790 board and to DPP-ZLEplus firmware; updated: Sect. Return Codes , functions GetInfo , Set / GetDPP_PHA_VirtualProbe and MallocDPPEvents ; added function GetCorrectionTables and Chap. ZLEplus Specific Functions .
May 5 th , 2014	07	Added a note to Set / GetTriggerPolarity functions
Jul 23 rd , 2014	08	Revised for final 743 support. Added support for x781. Added Chap. x743 Specific Functions . Added functions Set / GetDPP_VirtualProbe and GetDPP_SupportedVirtualProbes
Sep 26 th , 2014	09	Updated functions: SetDPPParameters , MallocDPPEvents , Set / GetNumEventsPerAggregate and Set / GetRecordLength
Feb 2 nd , 2015	10	Updated functions: Set / GetChannelSelfTrigger , Set / GetAcquisitionMode ; updated Chap. Introduction . Added function Calibrate .
Sep 9 th , 2015	11	Updated functions: GetInfo , Set / GetRecordLength , Set / GetChannelSelfTrigger , SetDPPParameters , DPP codes , Calibrate ; added a note to Set / GetDPPAcquisitionMode ; added Set / GetTriggerLogic , ReadTemperature , Set / GetChannelPairTriggerLogic ; added Chap. Technical Support
Nov 9 th , 2015	12	Updated functions: Set / GetRecordLength , Set / GetChannelEnableMask , Set / GetNumEventsPerAggregate , SetDPPEventAggregation , Set / GetDPPAcquisitionMode ; updated Sect. DPP Example Codes
Jan 25 th , 2016	13	Updated functions: Reset , Set / GetRecordLength , Calibrate . Moved function Set / GetChannelPulsePolarity to Chap. DPP Specific Functions .
Mar 9 th , 2016	14	Added note in Reset and OpenDigitizer functions for 743 family. Added support for V1743 in Set / GetAnalogMonOutput function. Added function Set / GetSAMTriggerCountVetoParam . Added note in Sect. DPP Example and Sec. Waveform Recording Firmware Example. Added support for 740D family. Updated Sect. Waveform Recording Firmware Example and Sect. DPP Example Codes
Oct 14 th , 2016	15	Updated MallocDPPEvents function: added <i>uint32_t Format2</i> in <i>CAEN_DGTZ_DPP_PSD_Event_t</i> structure, related to the CFD of the 751 digitizer family.
Feb 17 th , 2017	16	Updated functions: Set / GetChannelDCOffset , Set / GetAnalogMonOutput , Set / GetChannelSelfTrigger , Enable / DisableSAMPulseGen ; added Chap. Applications Notes
July 10 th , 2017	17	Added support for x725/x730 DPP-ZLE firmware. Specified Inputs and Outputs parameters for each described function. Updated Sect. x742 Offline Data Correction Functions and Chap. ZLEplus Specific Functions . Added examples in Sect. Waveform Recording Firmware Example and Sect. DPP Example Codes .
January 25 th , 2018	18	Added note on software compatibility with A3818 driver in Chap. 1 . Updated Fig. 4.1 . Extended 725 support in the Calibrate function. Updated Set / GetRunSynchronizationMode function. Added Set / GetDecimationFactor function.
October 15 th , 2018	19	Added support for x740D DPP-QDC firmware and x725/x730 DPP-DAW firmware.
May 2 nd , 2019	20	Added <i>CAEN_DGTZ_UnsupportedBaseAddress</i> in Sect. Return Codes . Added description of the <i>Pattern</i> field of the <i>CAEN_DGTZ_EventInfo_t</i> structure in GetEventInfo function. Revised Chap. 10 .

Symbols, Abbreviated Terms and Notation

ADC	Analog to Digital Converter
DPP	Digital Pulse Processing
FFT	Fast Fourier Transform
FSR	Full Scale Range
OS	Operating System
SBC	Single Board Computer

Reference Document

- [RD1] UM1934 – CAENComm Library User & Reference Manual
- [RD2] UM2784 - CAENDigitizer LabView User & Reference Manual
- [RD3] GD2783 – First Installation Guide to Desktop Digitizers & MCA
- [RD4] Technical Information Manual of V1718 and VX1718 VME – USB2.0 Bridge
- [RD5] Technical Information Manual of A3818 PCI Express Optical Link Controller
- [RD6] Technical Information Manual of A2818 PCI Optical Link Controller
- [RD7] UM1934 - CAENComm User & Reference Manual
- [RD8] AN2472 - CONET1 to CONET2 migration
- All documents can be downloaded at: www.caen.it/support-services/documentation-area/

CAEN S.p.A.
Via Vetraia, 11 55049 Viareggio (LU) - ITALY
Tel. +39.0584.388.398 Fax +39.0584.388.959
info@caen.it
www.caen.it

© CAEN SpA – 2019

Disclaimer

No part of this manual may be reproduced in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of CAEN SpA.

The information contained herein has been carefully checked and is believed to be accurate; however, no responsibility is assumed for inaccuracies. CAEN SpA reserves the right to modify its products specifications without giving any notice; for up to date information please visit www.caen.it.

MADE IN ITALY: We remark that all our boards have been designed and assembled in Italy. In a challenging environment where a competitive edge is often obtained at the cost of lower wages and declining working conditions, we proudly acknowledge that all those who participated in the production and distribution process of our devices were reasonably paid and worked in a safe environment (this is true for the boards marked "MADE IN ITALY", while we cannot guarantee for third-party manufactures).



Index

Purpose of this User Manual	2
Change Document Record	2
Symbols, Abbreviated Terms and Notation	3
Reference Document	3
Index	4
List of Figures	6
List of Tables	7
1 Introduction	8
Drivers & Libraries	9
Drivers	9
Libraries	9
Installation.....	11
Return Codes.....	12
2 Communication.....	13
OpenDigitizer	13
CloseDigitizer.....	14
WriteRegister.....	14
ReadRegister.....	14
Reset	15
GetInfo	15
Interrupt Configuration	17
Set / GetInterruptConfig.....	18
IRQWait.....	19
VMEIRQWait	20
VMEIRQCheck.....	21
VMEIACKCycle.....	21
RearmInterrupt	21
Data Readout	22
ClearData	22
DisableEventAlignedReadout	22
Set / GetMaxNumEventsBLT	23
MallocReadoutBuffer.....	23
FreeReadoutBuffer	24
ReadData	24
GetNumEvents.....	25
GetEventInfo	25
DecodeEvent	26
AllocateEvent.....	27
FreeEvent	27
Calibrate.....	27
ReadTemperature	28
LoadDRS4CorrectionData	28
Enable/Disable DRS4Correction.....	29
GetCorrectionTables	29
3 Trigger Configuration	30
SendSWtrigger	30
Set / GetSWTriggerMode	31
Set / GetExtTriggerInputMode.....	31
Set / GetChannelSelfTrigger	32
Set / GetGroupSelfTrigger.....	33
Set / GetChannelGroupMask	34
Set / GetChannelTriggerThreshold.....	34

	Set / GetGroupTriggerThreshold	35
	Set / GetRunSynchronizationMode	36
	Set / GetIOLevel	37
	Set / GetTriggerPolarity	38
	Set / GetGroupFastTriggerThreshold	39
	Set / GetGroupFastTriggerDCOffset	39
	Set / GetFastTriggerDigitizing	40
	Set / GetFastTriggerMode	40
	Set / GetDRS4SamplingFrequency	41
	Set / GetOutputSignalMode	41
4	Acquisition.....	42
	Set / GetChannelEnableMask	42
	Set / GetGroupEnableMask	42
	SWStartAcquisition	43
	SWStopAcquisition	43
	Set / GetRecordLength	44
	Set / GetPostTriggerSize	45
	Set / GetAcquisitionMode	46
	Set / GetChannelDCOffset	47
	Set / GetGroupDCOffset	47
	Set / GetDESMODE	48
	Set / GetDecimationFactor	48
	Set / GetZeroSuppressionMode	49
	Set / GetChannelZSParams	50
	Set / GetAnalogMonOutput	51
	Set / GetAnalogInspectionMonParams	52
	Set / GetEventPackaging	53
	Waveform Recording Firmware Example Code.....	54
	x742 Offline Data Correction Functions	56
	ApplyDataCorrection	56
	SaveCorrectionTables	56
	LoadCorrectionTable	57
	GetNumEvents	57
	GetEventPtr	57
	X742_DecodeEvent	58
5	x743 Specific Functions	59
	Set / GetSAMCorrectionLevel	59
	Set / GetSAMPostTriggerSize	60
	Set / GetSAMSamplingFrequency	60
	Read EEPROM	61
	LoadSAMCorrectionData	61
	Enable / DisableSAMPulseGen	62
	SendSAMPulse	62
	Set / GetSAMAcquisitionMode	63
	Set / GetChannelPairTriggerLogic	64
	Set / GetTriggerLogic	65
	Set / GetSAMTriggerCountVetoParam	66
6	DPP Specific Functions.....	67
	DPP codes	67
	Set / GetDPPPreTriggerSize	67
	Set / GetChannelPulsePolarity	68
	GetDPPEvents	68
	MallocDPPEvents	69
	FreeDPPEvents	70
	FreeDPPWaveforms	70
	MallocDPPWaveforms	71
	DecodeDPPWaveforms	72

	SetDPPEventAggregation	72
	Set / GetNumEventsPerAggregate	73
	Set / GetMaxNumAggregatesBLT	73
	SetDPPParameters	74
	Set / GetDPPAcquisitionMode	77
	Set / GetDPPTriggerMode	78
	Set / GetDPP_VirtualProbe	79
	GetDPP_SupportedVirtualProbes	80
	Set / GetDPP_PHA_VirtualProbe	81
	Set / GetDPP_PSD_VirtualProbe	82
	Set / GetDPP_CI_VirtualProbe	84
	DPP Example Codes	85
7	ZLEplus Specific Functions.....	88
	MallocZLEEvents	88
	FreeZLEEvents	89
	GetZLEEvents	89
	MallocZLEWaveforms	90
	FreeZLEWaveforms	90
	DecodeZLEWaveforms	91
	SetZLEParameters	91
8	Applications Notes	92
	Triggering with the x743 Modules	92
	Trigger Modes	92
	Enabling Channels as Trigger Sources	94
	Logic between Channels	94
	Global Trigger Logic	94
	Example of Coincidence between Channel 0 and Channel 5	95
	Setting Charge Mode for the x743 Modules	96
	Variable Definitions	96
	Enable/Disable Charge Mode	96
	Charge Parameters Setting	96
	Allocation on the Readout Buffer	96
	Reading Event Buffer	97
	Decoding Charge Event	97
	Free of the Readout Buffer	97
9	Examples of Communication Settings	98
	Example No.1	98
	Example No.2	99
	Example No.3	100
	Example No.4	102
	Example No.5	104
10	Technical Support	106

List of Figures

Fig. 1.1: Hardware and Software layers.....	10
Fig. 4.1: Acquisition example location in Windows OS	54
Fig. 6.1: DPP examples location in Windows OS	86
Fig. 8.1: The Trigger tab of CAEN WaveCatcher software	92
Fig. 8.2: Charge Mode parameters in WaveCatcher software	96
Fig. 9.1: Connection example no.1	98
Fig. 9.2: Connection example no.2	99
Fig. 9.3: Connection example no.3	100
Fig. 9.4: A2818 network scheme	101
Fig. 9.5: Connection example no.4	102
Fig. 9.6: Connection example no.5	104

List of Tables

Tab. 1.1: Return codes table.....12

1 Introduction

CAEN has developed a complete family of digitizers that consists of several models differing in sampling frequency, resolution, number of channels, form factor (VME, NIM and Desktop), memory size and other parameters. They all provide the possibility to be handled and readout by a host PC via different communication channels. In parallel to the hardware development, CAEN has made a big effort in developing algorithms for Waveform Recording and Digital Pulse Processing (DPP). All CAEN Digitizers support waveform recording firmware and, depending on the model, DPP firmware.

The CAENDigitizer library is specifically designed for CAEN boards like Digitizers, Digital MCAs and Digital Pulse Analyzer, to manage the acquisition, execute the readout, unpack data, send triggers, etc. The library supports the boards running waveform recording firmware as well as boards equipped with DPP firmware. Currently supported DPP firmware are PHA, PSD, QDC, ZLEPlus, DAW and CI.

The purpose of this library is to allow the user to perform the most common operations for data acquisition with a Digitizer **in an easy way**: with few lines of code the user can make a simple readout program with **no need of knowing the details of the registers** and the event data format.

The CAENDigitizer library relies on the CAENComm library **[RD1]**, allowing to implement a common interface to the higher software layers, masking the details of the physical channel and its protocol. Libraries and applications that rely on the CAENDigitizer become, in this way, independent from the physical layer.

The library is provided with examples and demo applications, including source files for custom user development.

Supported platforms are Windows and Linux, both 32 and 64 bits. A specific version of CAENDigitizer library has been developed for LabVIEW and is documented in **[RD2]**.

Drivers & Libraries

Drivers

In order to interface with the hardware, CAEN provides the drivers for all the different types of physical communication channels featured by the specific digitizer and compliant with Windows and Linux OS:

- **USB 2.0 Drivers for NIM/Desktop** boards are downloadable on CAEN website (www.caen.it) in the “Software/Firmware” area of the digitizer web page (**login required**).



Note: USB driver installation with Desktop/NIM digitizers is detailed for Windows OS in **[RD3]**.

- **USB 2.0 Drivers for V1718** CAEN Bridge, required to interface the VME boards, is downloadable on CAEN website (www.caen.it) in the “Software/Firmware” area of the V1718 web page (**login required**).



Note: for the installation of the V1718 USB driver, refer to the User Manual of the Bridge (**[RD4]**).

- **Optical Link Drivers** are managed by the A2818 PCI card or the A3818 PCIe card. The driver installation package is available on CAEN website in the “Software/Firmware” area at the A2818 or A3818 page (**login required**).



Note: for the installation of the Optical Link driver, refer to the User Manual of the specific Controller (**[RD5]**, **[RD6]**).

Libraries

The CAENDigitizer library is based on a set of middleware libraries also required by CAEN software tools for a correct functioning. These libraries, including also demo and example programs, represent a powerful base for users who want to develop customized applications for the digitizer control (communication, configuration, readout, etc.):

- **CAENVMELib** is a set of ANSI C functions which permit a user program to use and configure the CAEN Bridges and Controllers V1718/VX1718 (VME-USB2.0 Bridge), V2718/VX2718 (VME-PCI/PCIe Optical Link Bridge), A2818/A3818 (PCI/PCIe-CONET Controller).

The CAENVMELib installation package is available on CAEN website in the ‘Download’ area at the CAENVMELib Library page. See **[RD4]** for more details.

- **CAENComm** library manages the communication at low level (read and write access). The purpose of the CAENComm is to implement a common interface to the higher software layers, masking the details of the physical channel and its protocol, thus making the libraries and applications that rely on the CAENComm independent from the physical layer. Moreover, the CAENComm is based in turn on CAENVMELib and it requires the CAENVMELib library (access to the VME bus) even in cases where the VME is not used. This is the reason why **CAENVMELib must be already installed on your PC before installing the CAENComm**.

The CAENComm installation package is available on CAEN website in the ‘Download’ area at the CAENComm Library page. See **[RD7]** for more details.

Currently, the CAENComm, and so the CAENDigitizer, supports the following communication channels (see **Fig. 1.1**):

- PC → USB → Digitizer (either Desktop or NIM models)
- PC → USB → V1718 → VME → Digitizers (VME models only)
- PC → PCI (A2818) → CONET → Digitizers (all models)
- PC → PCI (A2818) → CONET → V2718 → VME → Digitizers (VME models only)
- PC → PCIe (A3818) → CONET → Digitizers (all models)
- PC → PCIe (A3818) → CONET → V2718 → VME → Digitizers (VME models only)

CONET (Chainable Optical NETWORK) indicates the CAEN proprietary protocol for communication on Optical Link. Refer to **[RD8]** for useful information.

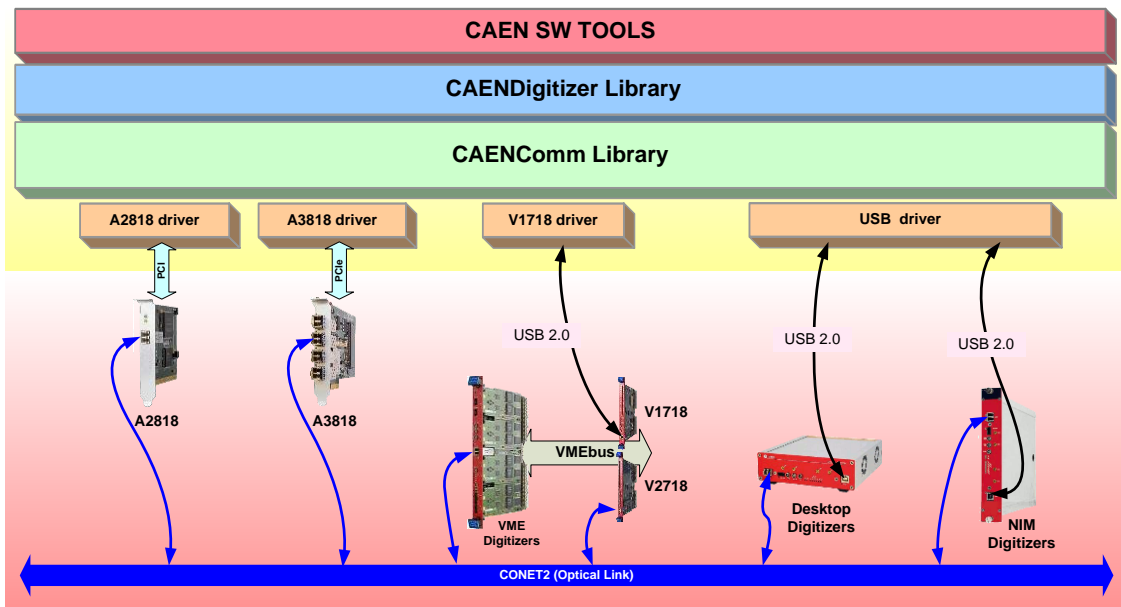


Fig. 1.1: Hardware and Software layers

Installation

The CAENDigitizer library is compliant with both Windows and Linux OS, 32 and 64 bits.

Before installing CAENDigitizer library, perform the following steps:

- **Make sure** that your **hardware** (Digitizer and/or Bridge, or Controller) is **properly installed** (refer to the related User Manual for hardware installation instructions).
- **Make sure** you **have installed the driver** for your OS and the physical communication layer to be used. Driver installation packages are downloadable from the CAEN website (**login required**) as reported in the **Drivers** paragraph.
- **Make sure** you **have installed the required CAEN libraries** CAENVMElib and CAENComm, downloadable from the CAEN website (**login required**).

Then:

- Go to the CAEN web site in the “Download” area of *CAENDigitizer Library* page
- Download the **CAENDigitizer Library installation package** related to your OS (**login required**)



Note: at this stage, if the required libraries still haven't been installed, it is possible to download them by clicking on the red link under the library packet link.

- **Extract files** to your host.

For Windows users:

- **Launch the installer** and **follow** the instructions in the **installation wizard**.

For Linux users:

- **Execute** the instructions in the **README file** within the library package.



Note: installation of the CAENDigitizer library also includes a “Samples” folder with a set of source files and the related Visual Studio projects that can be compiled and used as readout tools for CAEN Digitizers running the waveform recording firmware (see **Waveform Recording** Firmware Example) or a DPP firmware (see **DPP Example Codes**). Functions for the offline data correction of x742 digitizers are also available (see **x742 Offline Data Correction Functions**).

IMPORTANT:

USING THE CONET OPTICAL LINK BY A3818 PCIe CONTROLLER IN WINDOWS OS, DRIVER RELEASE 2.0 (OR HIGHER) STRICTLY REQUIRES CAENVMElib LIBRARY RELEASE 3.0.0 (OR HIGHER)!

Return Codes

Error code	Value	Meaning
CAEN_DGTZ_Success	0	Operation completed successfully
CAEN_DGTZ_CommError	-1	Communication error
CAEN_DGTZ_GenericError	-2	Unspecified error
CAEN_DGTZ_InvalidParam	-3	Invalid parameter
CAEN_DGTZ_InvalidLinkType	-4	Invalid Link Type
CAEN_DGTZ_InvalidHandler	-5	Invalid device handler
CAEN_DGTZ_MaxDevicesError	-6	Maximum number of devices exceeded
CAEN_DGTZ_BadBoardType	-7	Operation not allowed on this type of board
CAEN_DGTZ_BadInterruptLev	-8	The interrupt level is not allowed
CAEN_DGTZ_BadEventNumber	-9	The event number is bad
CAEN_DGTZ_ReadDeviceRegisterFail	-10	Unable to read the registry
CAEN_DGTZ_WriteDeviceRegisterFail	-11	Unable to write into the registry
CAEN_DGTZ_InvalidChannelNumber	-13	The Channel is busy
CAEN_DGTZ_ChannelBusy	-14	The channel number is invalid
CAEN_DGTZ_FPIOModelInvalid	-15	Invalid FPIO Mode
CAEN_DGTZ_WrongAcqMode	-16	Wrong acquisition mode
CAEN_DGTZ_FunctionNotAllowed	-17	This function is not allowed for this module
CAEN_DGTZ_Timeout	-18	Communication Timeout
CAEN_DGTZ_InvalidBuffer	-19	The buffer is invalid
CAEN_DGTZ_EventNotFound	-20	The event is not found
CAEN_DGTZ_InvalidEvent	-21	The event is invalid
CAEN_DGTZ_OutOfMemory	-22	Out of memory
CAEN_DGTZ_CalibrationError	-23	Unable to calibrate the board
CAEN_DGTZ_DigitizerNotFound	-24	Unable to open the digitizer
CAEN_DGTZ_DigitizerAlreadyOpen	-25	The Digitizer is already open
CAEN_DGTZ_DigitizerNotReady	-26	The Digitizer is not ready to operate
CAEN_DGTZ_InterruptNotConfigured	-27	The Digitizer has not the IRQ configured
CAEN_DGTZ_DigitizerMemoryCorrupted	-28	The digitizer flash memory is corrupted
CAEN_DGTZ_DPPFirmwareNotSupported	-29	The digitizer DPP firmware is not supported in this lib version
CAEN_DGTZ_InvalidLicense	-30	Invalid Firmware License
CAEN_DGTZ_InvalidDigitizerStatus	-31	The digitizer is found in a corrupted status
CAEN_DGTZ_UnsupportedTrace	-32	The given trace is not supported by the digitizer
CAEN_DGTZ_InvalidProbe	-33	The given probe is not supported for the given digitizer's trace
CAEN_DGTZ_UnsupportedBaseAddress	-34	The Base Address is not supported, as in case of DT and NIM devices
CAEN_DGTZ_NotYetImplemented	-99	The function is not yet implemented

Tab. 1.1: Return codes table

2 Communication

These functions allow the user to open and close the connection with the digitizer as well as to get board information such as the serial number, the model, the firmware revision, etc. To open one board is necessary to describe the physical communication channel from the PC to the device (as already indicated in the introduction). Once the device is opened, the function returns a handle that becomes the unique identifier of that device; any access operation to the device (except for VME IRQ management) will take place according to its handle, thus making transparent the physical channel.

OpenDigitizer

Description

The function opens the digitizer and gets the device handler. The two connection types are USB or OpticalLink. The various connection modes are described in sect. **Drivers & Libraries**. See chap. **Examples of Communication Settings** for practical examples of the different types of communication modes and the relevant parameters.



Note: in case of 743 family, this function also resets the SAMLONG DLLs.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_OpenDigitizer (CAEN_DGTZ_ConnectionType LinkType,
                        int LinkNum,
                        int ConetNode,
                        uint32_t VMEBaseAddress,
                        int *handle
                        );

//-----
//Types Definition
//-----
typedef enum CAEN_DGTZ_ConnectionType {
    CAEN_DGTZ_USB = 0,
    CAEN_DGTZ_OpticalLink = 1,
    CAEN_DGTZ_PCI_OpticalLink = 1, //Deprecated use 'CAEN_DGTZ_OpticalLink'
    CAEN_DGTZ_PCIE_OpticalLink = 1, //Deprecated use 'CAEN_DGTZ_OpticalLink'
    CAEN_DGTZ_PCIE_EmbeddedDigitizer = 1, //Deprecated use 'CAEN_DGTZ_OpticalLink'
} CAEN_DGTZ_ConnectionType;
```

Arguments

Name	I/O	Description
LinkType	Input	Indicates the physical communication channel. It can be: <ul style="list-style-type: none"> - <i>CAEN_DGTZ_USB</i> (either direct connection or VME through V1718) - <i>CAEN_DGTZ_OpticalLink</i> (either direct connection or VME through V2718) Note: functions <i>CAEN_DGTZ_PCI_OpticalLink</i> , <i>CAEN_DGTZ_PCIE_OpticalLink</i> , and <i>CAEN_DGTZ_PCIE_EmbeddedDigitizer</i> are now deprecated.
LinkNum	Input	Link number: <ul style="list-style-type: none"> - in case of USB, the link numbers are assigned by the PC when you connect the cable to the device; it is 0 for the first device, 1 for the second and so on. There is not a fixed correspondence between the USB port and the link number. - for the OpticalLink, the link number indicates which link of A2818 or A3818 is used; link index start from 0 (1st Optical link port in the 1st slot used). It is not known a priori which is the first slot used (it depends on the motherboard of the PC used.). IMPORTANT Note: if A2818 and A3818 are installed together, the A2818 have the lowest index assigned.
ConetNode	Input	The CONET node identifies which device in the daisy-chain is being addressed. The node is 0 for the first device in the chain, 1 for the second and so on. In case of USB, <i>ConetNode</i> must be 0.
VMEBaseAddress	Input	VME Base Address of the board (rotary switches setting) expressed as a 32-bit number in case you want to access a board through the VME bus. It MUST BE 0 in all other cases.
*handle	Output	Pointer to the handler returned by the open function

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

CloseDigitizer

Description

This function closes the digitizer.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_CloseDigitizer (int handle);
```

Arguments

Name	I/O	Description
handle	Input	Digitizer handler to be closed

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

WriteRegister

Description

Generic write access to one register of the digitizer. The CAENDigitizer library provides specific functions for most of the parameters settings; in the case where there is not a specific function to access a particular register or the user wants to force the writing of a datum, this function makes it possible. It is worth noticing that the overwriting of some settings can cause inconsistency of the operations.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_WriteRegister(int handle,
                        uint32_t Address,
                        uint32_t Data
                        );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
Address	Input	Register address. For the VME access, this is the lower 16-bit part of the VME address bus
Data	Input	32-bit data to write on the addressed register

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

ReadRegister

Description

Generic read access to one register of the digitizer (see **WriteRegister** for more details).

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_ReadRegister(int handle,
                       uint32_t Address,
                       uint32_t *Data
                       );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
Address	Input	Register address. For the VME access, this is the lower 16-bit part of the VME address bus
*Data	Input	Pointer to the 32-bit data read from the addressed register

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Reset

Description

This function resets the Digitizer. All internal registers and states are restored to default values.



Note: in case of 743 family, this function resets also the SAMLONG DLLs.

With respect to 730, 731, 751 and 761 digitizer families, starting from CAENDigitizer release 2.6.1, the Reset function has been modified so that it no longer includes the channel calibration routine implemented in the code. This calibration must be performed on command by the dedicated Calibrate function (go to page 27).

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_Reset (int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler to be reset

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

GetInfo

Description

This function reads from the board some information such as serial number, model, number of channels, firmware release and other parameters of the device.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetInfo(int handle,
                  CAEN_DGTZ_BoardInfo_t *BoardInfo
                  );

//-----
//Types Definition
//-----
typedef struct {
    char                ModelName[12];
    uint32_t            Model;
    uint32_t            Channels;
    uint32_t            FormFactor;
    uint32_t            FamilyCode;
    char                ROC_FirmwareRel[20];
    char                AMC_FirmwareRel[40];
    uint32_t            SerialNumber;
    char                MezzanineSerNum[4][8]; //used only for x743 boards
    uint32_t            PCB_Revision;
    uint32_t            ADC_NBits;
    uint32_t            SAMCorrectionDataLoaded; //used only for x743 boards
    int                 CommHandle;
    int                 VMEHandle;
    char                License[MAX_LICENSE_LENGTH];
} CAEN_DGTZ_BoardInfo_t;

typedef enum
{
    CAEN_DGTZ_V1724    =0L,
    CAEN_DGTZ_V1721    =1L,
    CAEN_DGTZ_V1731    =2L,
    CAEN_DGTZ_V1720    =3L,
    CAEN_DGTZ_V1740    =4L,
    CAEN_DGTZ_V1751    =5L,
    CAEN_DGTZ_DT5724    =6L,
    CAEN_DGTZ_DT5721    =7L,
    CAEN_DGTZ_DT5731    =8L,
    CAEN_DGTZ_DT5720    =9L,
    CAEN_DGTZ_DT5740    =10L,
    CAEN_DGTZ_DT5751    =11L,
    CAEN_DGTZ_N6724     =12L,
```

```

CAEN_DGTZ_N6721      =13L,
CAEN_DGTZ_N6731      =14L,
CAEN_DGTZ_N6720      =15L,
CAEN_DGTZ_N6740      =16L,
CAEN_DGTZ_N6751      =17L,
CAEN_DGTZ_DT5742     =18L,
CAEN_DGTZ_N6742      =19L,
CAEN_DGTZ_V1742      =20L,
CAEN_DGTZ_DT5780     =21L,
CAEN_DGTZ_N6780      =22L,
CAEN_DGTZ_V1780      =23L,
CAEN_DGTZ_DT5761     =24L,
CAEN_DGTZ_N6761      =25L,
CAEN_DGTZ_V1761      =26L,
CAEN_DGTZ_DT5743     =27L,
CAEN_DGTZ_N6743      =28L,
CAEN_DGTZ_V1743      =29L,
CAEN_DGTZ_DT5730     =30L,
CAEN_DGTZ_N6730      =31L,
CAEN_DGTZ_V1730      =32L,
CAEN_DGTZ_DT5790     =33L,
CAEN_DGTZ_N6790      =34L,
CAEN_DGTZ_V1790      =35L,
CAEN_DGTZ_DT5781     =36L,
CAEN_DGTZ_N6781      =37L,
CAEN_DGTZ_V1781      =38L,
CAEN_DGTZ_DT5725     =39L,
CAEN_DGTZ_N6725      =40L,
CAEN_DGTZ_V1725      =41L,
} CAEN_DGTZ_BoardModel_t;

typedef enum {
    CAEN_DGTZ_XX724_FAMILY_CODE = 0L,
    CAEN_DGTZ_XX721_FAMILY_CODE = 1L,
    CAEN_DGTZ_XX731_FAMILY_CODE = 2L,
    CAEN_DGTZ_XX720_FAMILY_CODE = 3L,
    CAEN_DGTZ_XX740_FAMILY_CODE = 4L,
    CAEN_DGTZ_XX751_FAMILY_CODE = 5L,
    CAEN_DGTZ_XX742_FAMILY_CODE = 6L,
    CAEN_DGTZ_XX780_FAMILY_CODE = 7L,
    CAEN_DGTZ_XX761_FAMILY_CODE = 8L,
    CAEN_DGTZ_XX743_FAMILY_CODE = 9L,
    CAEN_DGTZ_XX730_FAMILY_CODE = 11L,
    CAEN_DGTZ_XX790_FAMILY_CODE = 12L,
    CAEN_DGTZ_XX781_FAMILY_CODE = 13L,
    CAEN_DGTZ_XX725_FAMILY_CODE = 14L,
} CAEN_DGTZ_BoardFamilyCode_t;

```

Arguments

Name	I/O	Description
handle	Input	Device handler
*BoardInfo	Output	Pointer to the structure containing the Board Info filled by the <i>CAEN_DGTZ_GetInfo</i>

BoardInfo Fields

Name	Description
ModelName	Model name: for example “V1724”
Model	See type enum <i>CAEN_DGTZ_BoardModel_t</i>
Channels	Number of channels
FormFactor	Format Factor (VME, NIM, Desktop); see type <i>CAEN_DGTZ_BoardFormFactor_t</i>
FamilyCode	Family (ADC type); see type <i>CAEN_DGTZ_FamilyCode_t</i>
ROC_FirmwareRel	Firmware Revision of the FPGA on the mother board (ROC); for example “01.02”
AMC_FirmwareRel	Firmware Revision of the FPGA on the daughterboard (AMC)
SerialNumber	Serial number of the board
PCB_Revision	PCB Revision number
ADC_NBits	Number of bits of the ADC
CommHandle	Device handler for the underlying library <i>CAENComm</i>
VMEHandle	Device handler for the underlying library <i>CAENVME</i>
License	License number of the board

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Interrupt Configuration

All digitizers can generate interrupt requests (IRQ) to the PC to the occurrence of a condition: if the memory contains at least N_e events ready for reading, where N_e is a programmable parameter.

This allows to create programs that build the process of readout (read access to the memory buffer) on interrupts: they perform passive wait cycles, until they are awakened by the driver at the arrival of an interrupt from the digitizer; at such point, the process can read data, aware to find at least N_e events in memory, without having to check in advance the presence of data, as in the case of the readout based on polling.

The readout based on the interrupts is therefore more efficient, in terms of employment of the PC resources, compared to the one based on polling.

The interrupt requests are transferred from the digitizer to the PC via the optical link, in one of the following ways:

- Direct connection to the optical link (all models): the digitizer sends the interrupt request on the optical link to the A2818 PCI or A3818 PCIe connected to the PC, and these, in their turn, assert the interrupt request on the PCI bus or PCIe respectively. In this case, the interrupt request coming to the PC is uniquely associated with the digitizer which sent it.
- Connection via VME bus: in this case, the digitizer asserts the interrupt request on the VME bus on one of the 7 IRQ lines, and this request is detected by the VME master (V2718), which sends it via optical link to the PC, in the same manner described above. In this case, since the lines IRQ [7 .. 1] of the VME are shared with all modules on VME bus, it is necessary to identify the module that sent the request, as explained farther.



Note: interrupts cannot be used in case of communication via USB (either directly or through V1718 and VME)

Set / GetInterruptConfig

Description

This function enables / disables the digitizer to generate an interrupt request when the memory contains at least N_e events ready for reading, where N_e is the parameter *event_number*.

- In the case of VME models, the IRQ level to be activated on VME bus can be set from 1 to 7;
- in the case of the optical link, level should be 1.

The *status_id*, according to the specifications of the VME bus, is the value returned by the card during the interrupt acknowledge cycle and allows the operator to see which digitizer has asserted the interrupt request on the VME bus; in the programming stage, the user must set different *status_id* values for each digitizer. In the case of the optical link, the *status_id* is meaningless.

The mode parameter sets the interrupt release policy of the digitizer: in particular, **Roak** (Release On Acknowledge) mode in case of VME boards foresees that the request is issued immediately after the interrupt acknowledge cycle (IACK), while in the case of **Rora** (Release on Register Access) mode, the interrupt request is not released by the digitizer until the user accesses a particular registry to disable it; in the case of the digitizer, the release occurs by setting to zero the level in the VME Control register, by calling the "Set" function of **Set / GetInterruptConfig** with *status* = disabled.

The methods Rora and Roak, arising from the VME specifications, are implemented also in the CONET protocol of the optical link, with the exception that the Roak does not require the interrupt acknowledge cycle (IACK).

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetInterruptConfig (int handle,
                             CAEN_DGTZ_EnaDis_t state,
                             uint8_t level,
                             uint32_t status_id,
                             uint16_t event_number,
                             CAEN_DGTZ_IRQMode_t mode
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetInterruptConfig (int handle,
                             CAEN_DGTZ_EnaDis_t *state,
                             uint8_t *level,
                             uint32_t *status_id,
                             uint16_t *event_number,
                             CAEN_DGTZ_IRQMode_t *mode
                             );

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_ENABLE = 1L,
    CAEN_DGTZ_DISABLE = 0L,
} CAEN_DGTZ_EnaDis_t;

typedef enum {
    CAEN_DGTZ_IRQ_MODE_RORA = 0,
    CAEN_DGTZ_IRQ_MODE_ROAK = 1,
} CAEN_DGTZ_IRQMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
state	Input (Set) / Output (Get)	Enable/Disable
level	Input (Set) / Output (Get)	VME IRQ Level (from 1 to 7). Must be 1 for direct connection through CONET
status_id	Input (Set) / Output (Get)	32-bit number assigned to the device and returned by the device during the Interrupt Acknowledge
event_number	Input (Set) / Output (Get)	If the number of events ready for the readout is equal to or greater than <i>event_number</i> , then the digitizer asserts the interrupt request
mode	Input (Set) / Output (Get)	Interrupt release mode: <i>CAEN_DGTZ_IRQ_MODE_RORA</i> (release on register access) or <i>CAEN_DGTZ_IRQ_MODE_ROAK</i> (release on acknowledge)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

IRQWait

Description

Once set up the digitizer to generate an interrupt request (see **Set / GetInterruptConfig**), the reading process can enter a state of passive waiting to be woken up as the interrupt request from the digitizer which is communicating with (the one identified uniquely from the handle passed as a parameter), is sent. This function is valid only for direct connection to the digitizer with the optical link. In the case of communication via the VME bus, use **VMEIRQWait**. The timeout parameter indicates the maximum waiting time before being forced to wake up even without interrupt. In this case, the value returned by the function is 18.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_IRQWait(int handle,
                  uint32_t timeout
                  );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
timeout	Input	Timeout (max wait time) expressed in ms

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

VMEIRQWait

Description

This function, as the one described above, implements the passive waiting from which the waking occurs in response to an interrupt request from the digitizer. The main difference is that in this case, the digitizer asserts a IRQ (1 to 7) on the VME bus and this is transferred to the PC by the master VME V2718. Since other digitizers could be on the VME bus (and therefore different handles that identify them within the program), and each one can generate interrupts, even on the same IRQ line, the management of interrupts cannot take place through the handle of the digitizer (which cannot be uniquely associated with the request arrived at the PC) but must be performed through the handle of the master VME V2718 which is the unique collector of interrupt requests to the PC. Once awakened from the waiting status, the process of reading can understand what digitizer has actually sent the request via the interrupt acknowledge cycle.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_VMEIRQWait(CAEN_DGTZ_ConnectionType LinkType,
    int LinkNum,
    int ConetNode,
    uint8_t IRQMask,
    uint32_t timeout,
    int *VMEHandle
);

//-----
//Types Definition
//-----
typedef enum CAEN_DGTZ_ConnectionType {
    CAEN_DGTZ_USB = 0,
    CAEN_DGTZ_OpticalLink = 1,
    CAEN_DGTZ_PCI_OpticalLink = 1, //Deprecated use 'CAEN_DGTZ_OpticalLink'
    CAEN_DGTZ_PCIE_OpticalLink = 1, //Deprecated use 'CAEN_DGTZ_OpticalLink'
    CAEN_DGTZ_PCIE_EmbeddedDigitizer = 1, //Deprecated use 'CAEN_DGTZ_OpticalLink'
} CAEN_DGTZ_ConnectionType;
```

Arguments

Name	I/O	Description
LinkType	Input	Indicates the physical communication channel. It can be: -CAEN_DGTZ_USB (either direct connection or VME through V1718), -CAEN_DGTZ_OpticalLink (either direct connection or VME through V2718). Note: functions <i>CAEN_DGTZ_PCI_OpticalLink</i> , <i>CAEN_DGTZ_PCIE_OpticalLink</i> , and <i>CAEN_DGTZ_PCIE_EmbeddedDigitizer</i> are now deprecated.
LinkNum	Input	Link number: in case of USB, the link numbers are assigned by the PC when you connect the cable to the device; it is 0 for the first device, 1 for the second and so on. There is not a fixed correspondence between the USB port and the link number. For the CONET, the link number indicates which A2818 or A3818 is used; also in this case, it is not known a priori which PCI/PCIE card is assigned to which number.
ConetNode	Input	The CONET node identifies which device in the Daisy chain is being addressed. The node is 0 for the first device in the chain, 1 for the second and so on. In case of USB, ConetNode must be 0.
IRQMask	Input	A bit-mask indicating the IRQ lines
timeout	Input	Timeout (max wait time) expressed in ms
*VMEHandle	Output	Pointer to the device handler of the CAEN VME Bridge that received the interrupt request

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

VMEIRQCheck

Description

This function allows to read the status of interrupt requests on the VME bus (IRQ1-7) and, for this reason, the handle to be passed is the VME master one, not the digitizer one. This function can only be used for digitizer that communicate via the VME bus. The purpose of this function is almost exclusively for debugging.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_VMEIRQCheck(int VMEHandle,
                      uint8_t *Mask
                      );
```

Arguments

Name	I/O	Description
VMEHandle	Input	Device handler of the VME bridge that raised the interrupt request
*Mask	Output	Pointer to the bitmask representing the active VME interrupt requests (1=IRQ active, 0=IRQ not active)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

VMEIACKCycle

Description

This function performs an interrupt acknowledge cycle to know the board_id of the board that raised an interrupt. As described previously, in the case of interrupt requests on the VME bus, it is not possible to know in advance which digitizer asserted a certain IRQ line. Indeed, it could also happen that a line is asserted by any other slave on the VME bus with which no communication is established. For this reason, when the reading process on hold in a specific IRQ is awakened, it must perform an interrupt acknowledge cycle to see which one generated the interrupt. The identification is as follows: during acknowledge cycle (which is very similar to a read cycle), the slave that caused the interruption puts on his bus status_id, actually the value previously programmed by the user through the "Set" function of **Set / GetInterruptConfig** function. In the case of multiple cards having different values of the programmed status_id, the user will be able to figure out who sent the request, and then which one is to be read. It should be noted that in the case of multiple cards on the bus (even inhomogeneous), the interrupt management must be centralized, as the acknowledge cycle should be performed only once. It is therefore not recommended (although possible) to have more process waiting on the same IRQ line.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_VMEIACKCycle(int VMEHandle,
                      uint8_t level,
                      int32_t *board_id
                      );
```

Arguments

Name	I/O	Description
VMEHandle	Input	Device handler of the CAEN VME bridge that raised the interrupt
level	Input	IRQ level (from 1 to 7) on which to perform the interrupt acknowledge cycle
*board_id	Output	Pointer to the ID of the interrupter VME Board

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

RearmInterrupt

Description

Rearm the Interrupt.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_RearmInterrupt (int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes)

Data Readout

The data reading from the memories of the digitizer is done through BlockRead cycles (although it is possible also to run cycles to read each buffer). In the case of direct communication via USB or optical link, the protocol that manages the blocks transfer is CAEN proprietary and therefore there are no ambiguities or special options to be decided. Conversely, if reading takes place through the VME bus, since the standard provides different types of access and not all VME masters support all modes (or do it differently), the reading mode may need to be adapted according to the master features. The library foresees the use of master CAEN V1718 and V2718 and the readout mode is optimized for these modules.

ClearData

Description

This function clears the data stored in the buffers of the Digitizer.



Note: generally, it is not necessary to call this function, because the digitizer runs automatically a clear cycle when an acquisition starts. The function can be used during an acquisition when aware that the data stored in memory are not interesting and not going to be read

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API  
CAEN_DGTZ_ClearData(int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

DisableEventAlignedReadout

Description

By default, in the data transfer from the memory of the digitizer to the PC, regardless of the type of link used, events are aligned: the digitizer stop the transfer after transferring an integer number N_e of events, where N_e is user programmable through the "Set" function of **Set / GetMaxNumEventsBLT**, even if the user has requested the transfer of more data. In the case of communication via USB and optical links, the premature termination of the transfer is foreseen by the protocol; instead, for the VME Block Transfer, the transfer is interrupted by the digitizer asserting the bus error (if enabled, see above).

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API  
CAEN_DGTZ_DisableEventAlignedReadout(int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetMaxNumEventsBLT

Description

Concerning the digitizers running the waveform recording firmware for waves digitizing, this function sets/gets the maximum number of events for each transfer. Regardless of the type of link, during a block transfer cycle, the digitizer stops the transfer after a predetermined number of events (or when the memory is empty). The greater the number of events transferred (and thus the size of the block read), the greater the efficiency of the readout, since the protocol overhead is smaller. In contrast, higher values for **MaxNumEventsBLT** imply the need to allocate a memory buffer for very large the readout.



Note: if using DPP-PHA, DPP-PSD, DPP-QDC or DPP-CI firmware, you have to refer to the **SetDPPEventAggregation** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetMaxNumEventsBLT(int handle,
                             uint32_t numEvents
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetMaxNumEventsBLT(int handle,
                             uint32_t *numEvents
                             );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
numEvents	Input (Set)/ Output (Get)	Maximum number (or pointer to the maximum number, in case of Get) of events to transfer in a BlockRead

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

MallocReadoutBuffer

Description

This function allocates the memory buffer for the data block transfer from the digitizer to the PC. The size of the buffer allocated is calculated by the function according to the size of the event, the number of enabled channels and the maximum number of events transferred by each block transfer (see previous function). For this reason, the function must be called after having programmed the digitizer. If the parameters that determine the size of the buffer change, it is necessary to free it by calling the **FreeReadoutBuffer** function and then reallocated.



Note: the buffer pointer must be initialized to NULL.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_MallocReadoutBuffer(int handle,
                              char **buffer
                              uint32_t *size);
```

Arguments

Name	I/O	Description
handle	Input	Device handler
**buffer	Output	Address of the buffer pointer to the allocated memory buffer (WARNING: **buffer MUST be initialized to NULL)
*size	Output	Pointer to the value of the size (in byte) of the buffer allocated

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

FreeReadoutBuffer

Description

Frees memory allocated by the **MallocReadoutBuffer** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_FreeReadoutBuffer(char **buffer);
```

Arguments

Name	I/O	Description
**buffer	Input	Address of the buffer pointer to the allocated memory buffer to free, returned by the MallocReadoutBuffer function.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

ReadData

Description

This function performs a block transfer of data from the digitizer to the computer. The size of the block to be transferred is determined by the function according to parameters set and the mode of readout. The block can contain one or more events. The data is transferred into the buffer memory previously allocated by **MallocReadoutBuffer** function. The function returns in *bufferSize* the size of the data block read from the card, expressed in bytes.



Note:

CAEN_DGTZ_SLAVE_TERMINATED_READOUT_MBLT for VME accesses:

In this case the digitizer is programmed to assert the VME Bus Error during a Block Transfer cycle to prematurely end the cycle when it no longer has data to transfer or has completed the transfer of the maximum number of events planned (see *BLT_EVENT_NUM* register, or **Set / GetMaxNumEventsBLT** function). This use of the Bus Error, though not specifically provided by the VME standard for this purpose, it is actually very common. However, some VME masters have a Bus Error management not suitable for this purpose.

CAEN_DGTZ_POLLING_MBLT for VME accesses:

The VME Bus Error generation is disabled, the transfer always continues until the completion of the number of bytes required and, if there are no data to be transferred, the digitizer will insert filler words (0xFFFFFFFF)

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_ReadData(int handle,
                   CAEN_DGTZ_ReadMode_t mode,
                   char *buffer,
                   uint32_t *bufferSize
                   );

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_SLAVE_TERMINATED_READOUT_MBLT = 0,
    CAEN_DGTZ_SLAVE_TERMINATED_READOUT_2eVME = 1,
    CAEN_DGTZ_SLAVE_TERMINATED_READOUT_2eSST = 2,
    CAEN_DGTZ_POLLING_MBLT = 3,
    CAEN_DGTZ_POLLING_2eVME = 4,
    CAEN_DGTZ_POLLING_2eSST = 5,
} CAEN_DGTZ_ReadMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode	Input	CAEN_DGTZ_ReadMode_t type, identifying the reading mode (see the definition of the type above)
*buffer	Output	Address the buffer that will store data
*bufferSize	Output	Pointer to the size of the data stored in the buffer (expressed in bytes)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

GetNumEvents

Description

This function scans the readout buffer and gets the number of events contained in the data block previously read by the **ReadData** function. The number of events is returned in the parameter *numEvents*.



Note: if using DPP-PHA, DPP-PSD, DPP-QDC or DPP-CI firmware, you must refer to the **GetDPPEvents** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetNumEvents(int handle,
                       char *buffer,
                       uint32_t buffsize,
                       uint32_t *numEvents
                       );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*buffer	Input	Pointer to the readout buffer
buffsize	Input	Size of the data stored in the acquisition buffer. This value is given by the ReadData function.
*numEvents	Output	Pointer to the number of events contained in the readout buffer

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

GetEventInfo

Description

This function retrieves the information (trigger time stamp, event number, channel mask, etc.) associated to one event contained in the readout buffer. This function reads the header of the *numEvent* event in the buffer, fills the eventInfo structure and set the data pointer *EventPtr* to the first word of the event data in the readout buffer. This pointer will be passed to the **DecodeEvent** function described below.



Note: if using DPP-PHA, DPP-PSD, DPP-QDC or DPP-CI firmware, you must refer to the **GetDPPEvents** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetEventInfo(int handle,
                       char *buffer,
                       uint32_t buffsize,
                       int32_t numEvent,
                       CAEN_DGTZ_EventInfo_t *eventInfo,
                       char **EventPtr
                       );

//-----
//Types Definition
//-----
typedef struct
{
    uint32_t EventSize;
    uint32_t BoardId;
    uint32_t Pattern;
    uint32_t ChannelMask;
    uint32_t EventCounter;
    uint32_t TriggerTimeTag;
} CAEN_DGTZ_EventInfo_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*buffer	Input	Pointer to the readout buffer
buffsize	Input	Size of the data stored in the readout buffer
numEvent	Input	Number of events stored in the acquisition buffer
*eventInfo	Output	Pointer to the CAEN_DGTZ_EventInfo_t structure that contains the information about the requested event. Find the detailed description of the event structure in the Digitizer User Manual. IMPORTANT note: the <i>Pattern</i> field of the structure is a 32-bit word where

		bit[15:0] represent the Pattern described in the event format of the digitizer User Manual, while bit[31:16] are not used for all digitizers except for the x751 ones. For the x751 digitizers only, the value of bit[18:16] indicates which sample the Trigger Time Tag refers to. For all digitizers but x751 ones, TTT is referred to sample[0] in the configured record length. In case of x751 digitizers, if bit[18:16] = 5, as an example, then TTT refers to sample[5]. This information is important when synchronizing multiple x751 digitizers.
**EventPtr	Output	Pointer to the requested event data in the readout buffer

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

DecodeEvent

Description

Each type of digitizer has a different event data format. This function decodes (unpacks) the data of a specified event and fills the event structure containing the data of each channel (i.e. the waveform and/or other parameters in case of DPP) separately. There are two ways to allocate the memory for the unpacked event data:

- If the pointer ****Evt** to the event structure passed to the function is initialized to NULL, then the event is automatically allocated by the **DecodeEvent** function that knows the exact size of the decoded event data, hence there is no waste in the memory usage. In this case, the user must free the event memory buffer once it has been used.
- The memory buffer for the decoded event can be allocated once at the beginning of the acquisition; this is done by the **AllocateEvent** function. This solution is more efficient in terms of readout rate (no waste of time to allocate and free the memory) but requires more memory because the buffer must be able to contain the maximum event size. In this mode, the memory free must be done at the end of the acquisition.



Note: if using DPP-PHA, DPP-PSD, DPP-QDC or DPP-CI firmware, you must refer to the **GetDPPEvents** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_DecodeEvent(int handle,
                      char *evtPtr,
                      void **Evt
                      );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*evtPtr	Input	Pointer to the event data in the readout buffer (this is the pointer returned by the GetEventInfo function).
**Evt	Output	Pointer to the decoded event structure. This pointer must be initialized to NULL if you want the function to allocate the memory buffer automatically. Conversely, if the memory buffer has been already allocated, this is the pointer to that memory buffer. The latter case is more efficient in terms of readout rate.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

AllocateEvent

Description

This function allocates the memory buffer for the decoded event data. The size of the buffer is calculated to keep the maximum event size.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_AllocateEvent(int handle,
                        void **Evt
                        );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
**Evt	Input	Pointer to memory buffer for the event structure.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

FreeEvent

Description

This function releases the event memory buffer allocated by either the **DecodeEvent** or **AllocateEvent** function.



Note: if using DPP-PHA, DPP-PSD or DPP-CI firmware, you must refer to the **FreeDPPEvents** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_FreeEvent(int handle,
                    void **Evt
                    );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
**Evt	Input	Pointer to memory buffer for the event structure.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Calibrate

Description

This function must be used with 725, 730, 751 and 761 digitizer families, as well as with V1731 digitizers, to perform the channel calibration after the ADCs have stabilized their operating temperature (see also **ReadTemperature** function). The calibration will not need to be repeated at each acquisition run unless the operating temperature changes significantly or clock settings are modified (e.g. switching from internal to external clock).

In case of V1731 digitizers, this function performs a software calibration to align the channels samples. This calibration is required any time the digitizer switches from DES mode to normal mode and viceversa.



Note: please refer to the User Manual of the relevant board for the calibration description.

Synopsis

```
CAEN_DGTZ_ErrorCode
CAENDGTZ_API CAEN_DGTZ_Calibrate(int handle
                                );
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

ReadTemperature

Description

This function is to be used with 725, 730 and 751 digitizer families and returns the channel temperature value. Monitoring for the internal temperature takes part in the channel calibration required by such modules (see also **Calibrate** function).



Note: please refer to the User Manual of the relevant board for the calibration description.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_ReadTemperature(int handle,
                          int32_t ch,
                          uint32_t *temp
                          );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
ch	Input	The channel number referred to the temperature value to read
*temp	Output	Pointer to the channel temperature value expressed in °C

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

LoadDRS4CorrectionData

Description

Regarding the x742 series, a data correction is required to compensate for unavoidable construction differences in the DRS4 chips (for details, please refer to the User Manual of the board). This function loads the correction parameters stored on board, while a **DecodeEvent** function is then needed to apply them. The correction parameters to load depend on the operating sampling frequency.



Note: to be used only with x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_LoadDRS4CorrectionData (int handle,
                                  CAEN_DGTZ_DRS4Frequency_t frequency
                                  );

//-----
//Types Definition
//-----

typedef enum
{
    CAEN_DGTZ_DRS4_5GHz    = 0L,
    CAEN_DGTZ_DRS4_2_5GHz = 1L,
    CAEN_DGTZ_DRS4_1GHz   = 2L,
} CAEN_DGTZ_DRS4Frequency_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
frequency	Input	The DRS4 sampling frequency, defined as a CAEN_DGTZ_DRS4Frequency_t type

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Enable/Disable DRS4Correction

Description

Enables/disables the data correction in the x742 series.



Note: to be used only with x742 series.



Note: if enabled, the data correction through the **DecodeEvent** function only applies if a **LoadDRS4CorrectionData** has been previously called, otherwise the **DecodeEvent** runs the same, but data will be provided out not compensated.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_EnableDRS4Correction (int handle);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_DisableDRS4Correction (int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

GetCorrectionTables

Description

This function reads the correction tables from the x742 digitizer FLASH, related to the selected sampling frequency, and fills in a structure with the read values. This way, the stored correction table become available to be used, for instance, with a software relying on the CAENDigitizer library.



Note: to be used only with 742 digitizer series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetCorrectionTables(int handle,
                              int frequency,
                              void *CTable
                              );

//-----
//Types Definition
//-----
typedef struct {
    int16_t    cell[MAX_X742_CHANNEL_SIZE][1024];
    int8_t     nsample[MAX_X742_CHANNEL_SIZE][1024];
    float      time[1024];
} CAEN_DGTZ_DRS4Correction_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
frequency	Input	Sampling frequency of the DRS4 chips which sample the input analog signal and the fast trigger signal
*CTable	Output	The pointer to a <i>CAEN_DGTZ_DRS4Correction_t</i> structure to be filled in with the values read from the x742 FLASH.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

3 Trigger Configuration

The acquisition in the digitizer is ruled by the trigger, which is a signal that decides when to start the acquisition window and save samples of the ADC or the values of interest calculated on line (DPP) in the digitizer memory.

The digitizer can have the following trigger sources: External Trigger (digital signal from the panel), Software Trigger (write access to the specific register), Self Trigger Channel (internal signal generated by a digitizer channel under certain conditions, for example when the input signal exceeds a programmable threshold).

All trigger sources can be enabled or not to generate the acquisition trigger for the channels. Similarly, it is possible to decide which triggers should participate in the generation of the Trigger Output (NIM or TTL digital output of the digitizer panel). Trigger Output cannot necessarily coincide with the acquisition trigger: for example, in order to trigger multiple cards at once, as one of their channel has “auto triggered”; for this purpose, the auto triggering channel is used only to generate the Trigger Outputs (but not for the acquisition trigger); all Trigger Outputs are ORed externally to the cards and the resulting signal is sent in parallel to all cards Trigger Inputs, which are programmed to enable only the Trigger Input to generate the acquisition Trigger.



Note: in digitizer series X740, the auto trigger channel is divided into two levels: each 8-channel group generates a “group local trigger”, given by the OR of channel triggers enabled to generate them. The group triggers, in their turn, may participate or not to generate the acquisition trigger and / or trigger output.

SendSWtrigger

Description

This function sends a Software trigger to the Digitizer. The SW trigger can be used to save an acquisition window on all channels at the same time and/or to generate a pulse on the Trigger Output of the board, according to the SW trigger mode set by the “Set” function of the **Set / GetSWTriggerMode**.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API  
CAEN_DGTZ_SendSWtrigger (int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetSWTriggerMode

Description

This function decides whether the trigger software should only be used to generate the acquisition trigger, only to generate the trigger output, or both.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetSWTriggerMode(int handle,
                           CAEN_DGTZ_TriggerMode_t mode
                           );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetSWTriggerMode(int handle,
                           CAEN_DGTZ_TriggerMode_t *mode);

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_TRGMODE_DISABLED      = 0,
    CAEN_DGTZ_TRGMODE_EXTOUT_ONLY   = 2,
    CAEN_DGTZ_TRGMODE_ACQ_ONLY      = 1,
    CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT = 3,
}CAEN_DGTZ_TriggerMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode	Input (Set)/Output (Get)	Trigger mode value (or pointer in case of Get)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetExtTriggerInputMode

Description

This function decides whether the external trigger should only be used to generate the acquisition trigger, only to generate the trigger output, or both.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetExtTriggerInputMode(int handle,
                                  CAEN_DGTZ_TriggerMode_t mode
                                  );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetExtTriggerInputMode(int handle,
                                  CAEN_DGTZ_TriggerMode_t *mode);

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_TRGMODE_DISABLED      = 0,
    CAEN_DGTZ_TRGMODE_EXTOUT_ONLY   = 2,
    CAEN_DGTZ_TRGMODE_ACQ_ONLY      = 1,
    CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT = 3,
}CAEN_DGTZ_TriggerMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode	Input (Set)/Output (Get)	External trigger mode value (or pointer in case of Get)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetChannelSelfTrigger

Description

This function mainly applies to the digitizers running the waveform recording firmware for waves digitizing, since it manages the global trigger generation and its propagation through the TRG-OUT connector.

For the x740 series, use the **Set / GetGroupSelfTrigger** function.



Note: since 730, 725 and 743 families have even and odd channels paired, the user shouldn't call this function separately for the channels of the same pair, otherwise the second call will overwrite the setting of the first one. The user should instead call at maximum once for every pair with the relevant bits of the *channelmask* already set to the correct value.

In case of Digital Pulse Processing (DPP) firmware, this function can also be used when it is required to manage the global trigger and/or propagate it out on TRG-OUT connector.



Note: with DPP firmware, you should enable each channel you want to acquire to self-trigger on its own input. If you want to disable this option, you have to set `DPPParams.selftr = 0` (for DPP-PSD and DPP-CI).

In case of DPP-PHA this option is not available in the library, while it is available via register write: set bit[24] = 0 of register 0x1n80 to enable the self-trigger of channel n, set it to 1 to disable the self-trigger.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelSelfTrigger(int handle,
                                CAEN_DGTZ_TriggerMode_t mode,
                                uint32_t channelmask
                                );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelSelfTrigger(int handle,
                                CAEN_DGTZ_TriggerMode_t *mode,
                                uint32_t channel
                                );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_TRGMODE_DISABLED      = 0,
    CAEN_DGTZ_TRGMODE_EXTOUT_ONLY  = 2,
    CAEN_DGTZ_TRGMODE_ACQ_ONLY     = 1,
    CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT = 3,
}CAEN_DGTZ_TriggerMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode	Input (Set)/Output (Get)	Channel Self Trigger mode CAEN_DGTZ_TRGMODE_DISABLED = 0, The channel self-trigger neither participates in the global trigger generation nor it is propagated on TRG-OUT. CAEN_DGTZ_TRGMODE_EXTOUT_ONLY = 2, The channel self-trigger doesn't participate in the global trigger generation, but it is propagated out on TRG-OUT. CAEN_DGTZ_TRGMODE_ACQ_ONLY = 1, The channel self-trigger participates in the global trigger generation but it is not propagated out on TRG-OUT. CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT = 3, The channel self-trigger participates in the global trigger generation and it is propagated out on TRG-OUT.
channelmask	Input	Channel mask to select the channels affected by the SetChannelSelfTrigger function. The function applies only to those channels that have the relevant bit in the mask equal to 1.
channel	Input	INT value corresponding to the channel index (only for Get)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetGroupSelfTrigger

Description

This function is valid only for the x740 series. In fact, in this type of digitizer, the channels are grouped 8 by 8. The trigger properties are referred to the groups and cannot be set individually channel by channel. Each group of 8 channels generates one single self trigger which is the OR of the 8 self triggers in the group (with a programmable trigger enable mask, see next function). The group self trigger can generate the acquisition trigger for the board and/or a pulse on the Trigger Output.



Note: to be used only with x740 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetGroupSelfTrigger(int handle,
                               CAEN_DGTZ_TriggerMode_t mode,
                               uint32_t groupmask
                              );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetGroupSelfTrigger(int handle,
                               CAEN_DGTZ_TriggerMode_t *mode
                               uint32_t group
                              );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_TRGMODE_DISABLED      = 0,
    CAEN_DGTZ_TRGMODE_EXTOUT_ONLY   = 2,
    CAEN_DGTZ_TRGMODE_ACQ_ONLY      = 1,
    CAEN_DGTZ_TRGMODE_ACQ_AND_EXTOUT = 3,
}CAEN_DGTZ_TriggerMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode	Input (Set)/Output (Get)	Group Self Trigger mode value (or pointer in case of Get)
groupmask	Input	Group mask to select the groups affected by the SetGroupSelfTrigger function. The function applies only to those groups that have the relevant bit in the mask equal to 1.
group	Input	INT value corresponding to the group index (only for Get)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetChannelGroupMask

Description

This function decides which channels in a group of 8 participate to the generation of the self-trigger of that group. The self-trigger is the OR of the channels enabled by this function that are above the threshold.

WARNING: the channels that are not connected must be disabled here, otherwise it may happen that one channel has a DC offset higher than the threshold and it keeps the OR always active.



Note: to be used only with x740 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelGroupMask(int handle,
                               uint32_t group,
                               uint32_t channelmask
                              );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelGroupMask(int handle,
                               uint32_t group,
                               uint32_t *channelmask
                              );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
group	Input	INT value corresponding to the group index
channelmask	Input (Set)/Output (Get)	Channels Trigger mask for the group (8 bits)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetChannelTriggerThreshold

Description

This function sets the Trigger Threshold for a specific channel. The threshold is applied to the digital signal after the ADC and it is expressed in ADC counts. The user should take care of the DC offset adjust when converting the digital threshold in the corresponding voltage level on the analog input.


For the x740 series, use the **Set / GetGroupTriggerThreshold** function. For DPP-QDC firmware on X740D family, use this function. For the others DPP firmware, use the **SetDPPParameters** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelTriggerThreshold(int handle,
                                     uint32_t channel,
                                     uint32_t Tvalue
                                    );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelTriggerThreshold(int handle,
                                     uint32_t channel,
                                     uint32_t *Tvalue
                                    );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channel	Input	Channel to set
Tvalue	Input (Set)/Output (Get)	Threshold value (in ADC counts). Note: in case of x743 digitizer, the threshold value is described in the scheme below. <div style="text-align: center;">  </div>

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetGroupTriggerThreshold

Description

This function sets/gets the Trigger Threshold for a specified group of channels. The threshold is common to the 8 channels in the group. See the **Set / GetChannelTriggerThreshold** function for further details.



Note: to be used only with x740 series. Not used for DPP-QDC firmware for x740D family.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetGroupTriggerThreshold(int handle,
                                   uint32_t group,
                                   uint32_t Tvalue
                                   );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetGroupTriggerThreshold(int handle,
                                   uint32_t group,
                                   uint32_t *Tvalue
                                   );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
group	Input	Group to set
Tvalue	Input (Set)/Output (Get)	Threshold value (in ADC counts)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetRunSynchronizationMode

Description

Sets/gets the run synchronization mode of the digitizer, used to synchronize an acquisition on multiple boards.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetRunSynchronizationMode (int handle,
                                     CAEN_DGTZ_RunSyncMode_t mode
                                    );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetRunSynchronizationMode (int handle,
                                     CAEN_DGTZ_RunSyncMode_t *mode
                                    );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_RUN_SYNC_Disabled,
    CAEN_DGTZ_RUN_SYNC_TriggerOutTriggerInDaisyChain,
    CAEN_DGTZ_RUN_SYNC_TriggerOutSinDaisyChain,
    CAEN_DGTZ_RUN_SYNC_SinFanout,
    CAEN_DGTZ_RUN_SYNC_GpioDaisyChain
} CAEN_DGTZ_RunSyncMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	<p>The run synchronization mode to set/get, defined as a CAEN_DGTZ_RunSyncMode_t type:</p> <p>CAEN_DGTZ_RUN_SYNC_Disabled = run synchronization mode is disabled</p> <p>CAEN_DGTZ_RUN_SYNC_TriggerOutTriggerInDaisyChain = to be used with the run synchronization based on TRG-OUT / TRG-IN Daisy chain; each slave board is configured in "First Trigger Controlled" start/stop mode (see the digitizer User Manual); the start acquisition must be by software (SWStartAcquisition function) or by external signal on TRG-IN; stop acquisition is by software too (SWStopAcquisition function);</p> <p>CAEN_DGTZ_RUN_SYNC_TriggerOutSinDaisyChain = to be used with the run synchronization mode based on TRG-OUT / S-IN Daisy chain, where each slave board is configured in "S-IN Controlled" start/stop mode (see the digitizer User Manual); the start acquisition can be by software (SWStartAcquisition function) or by external signal on S-IN;</p> <p>CAEN_DGTZ_RUN_SYNC_SinFanout = to be used when sending the start/stop run in parallel to all the boards from an external source on S-IN connectors, where each board is configured in "S-IN Controlled" start/stop mode (see the digitizer User Manual);</p> <p>CAEN_DGTZ_RUN_SYNC_GpioDaisyChain = not used.</p>

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetIOLevel

Description

Sets/gets the I/O level.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetIOLevel (int handle,
                      CAEN_DGTZ_IOLevel_t level
                      );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetIOLevel (int handle,
                      CAEN_DGTZ_IOLevel_t *level
                      );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_IOLevel_NIM = 0L,
    CAEN_DGTZ_IOLevel_TTL = 1L,
} CAEN_DGTZ_IOLevel_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
level/*level	Input (Set)/Output (Get)	The I/O level of the digitizer to set/get, defined as a CAEN_DGTZ_IOLevel_t type

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetTriggerPolarity

Description

Sets/gets the trigger polarity of a specified channel.



Note: not to be used with DPP firmware.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetTriggerPolarity (int handle,
                             uint32_t channel,
                             CAEN_DGTZ_TriggerPolarity_t Polarity
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetTriggerPolarity (int handle,
                              uint32_t channel,
                              CAEN_DGTZ_TriggerPolarity_t *Polarity
                              );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_TriggerOnRisingEdge = 0L,
    CAEN_DGTZ_TriggerOnFallingEdge = 1L,
} CAEN_DGTZ_TriggerPolarity_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channel	Input	The channel to set/get the trigger polarity for
Polarity/*Polarity	Input (Set)/Output (Get)	The polarity of the trigger to set/get, defined as a CAEN_DGTZ_TriggerPolarity_t type



Note: *channel* parameter is unused (i.e. the setting is common to all channels) for those digitizers that do not support the individual trigger polarity setting. Please refer to the Registers Description document of the relevant board for check

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetGroupFastTriggerThreshold

Description

Sets/gets the threshold value on TRn input (used as external trigger) for the local trigger generation in x742 series. As the threshold is a hardware threshold (input of a programmable 16-bit DAC, whose voltage output goes to a comparator), it is not easy to set, and the user can refer to the board User Manual for setting examples.



Note: to be used only with x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetGroupFastTriggerThreshold (int handle,
                                         uint32_t group,
                                         uint32_t Tvalue
                                         );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetGroupFastTriggerThreshold (int handle,
                                         uint32_t group,
                                         uint32_t *Tvalue
                                         );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
group	Input	The channels group the threshold is applied to
Tvalue/*Tvalue	Input (Set)/Output (Get)	The value of the TRn threshold to set/get

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetGroupFastTriggerDCOffset

Description

Regarding the x742 series, these functions set/get the TRn signal DC offset when it is sampled in the DRS4 chips to make positive, negative or bipolar input signals to be compliant with the DRS4 input dynamics. The DC offset also affects the TRn when used as trigger, in this case it relates to the threshold setting above described (please refer to the board User Manual for setting examples).



Note: to be used only with x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetGroupFastTriggerDCOffset (int handle,
                                         uint32_t group,
                                         uint32_t DCvalue
                                         );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetGroupFastTriggerDCOffset (int handle,
                                         uint32_t group,
                                         uint32_t *DCvalue
                                         );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
group	Input	The channels group the DC offset is applied to
DCvalue/*DCvalue	Input (Set)/Output (Get)	The value of the TRn DC offset to set/get

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetFastTriggerDigitizing

Description

Regarding the x742 series, enables/disables (set) the presence of the TRn signal in the data readout as well as allows for checking the status of the setting (get).



Note: to be used only with x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetFastTriggerDigitizing (int handle,
                                     CAEN_DGTZ_EnaDis_t enable
                                    );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetFastTriggerDigitizing (int handle,
                                     CAEN_DGTZ_EnaDis_t *enable
                                    );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_ENABLE = 1L,
    CAEN_DGTZ_DISABLE = 0L,
} CAEN_DGTZ_EnaDis_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
enable/*enable	Input (Set)/Output (Get)	The enable flag to set/get, defined as a CAEN_DGTZ_EnaDis_t type

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetFastTriggerMode

Description

Enables/disables (set) the TRn input as local trigger in x742 series, as well as allows for checking the status of the setting (get).



Note: to be used only with x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetFastTriggerMode (int handle,
                              CAEN_DGTZ_TriggerMode_t mode
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetFastTriggerMode (int handle,
                              CAEN_DGTZ_TriggerMode_t *mode
                             );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_TRGMODE_DISABLED = 0L,
    CAEN_DGTZ_TRGMODE_ACQ_ONLY = 1L,
} CAEN_DGTZ_TriggerMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	The fast trigger mode value to set/get, defined as a CAEN_DGTZ_TriggerMode_t type

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetDRS4SamplingFrequency

Description

Regarding the x742 series, sets/gets the sampling frequency of the DRS4 chips which sample the input analog signal and the fast trigger signal.



Note: to be used only with x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDRS4SamplingFrequency (int handle,
                                     CAEN_DGTZ_DRS4Frequency_t frequency
                                    );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDRS4SamplingFrequency (int handle,
                                     CAEN_DGTZ_DRS4Frequency_t *frequency
                                    );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_DRS4_5GHz           = 0L,
    CAEN_DGTZ_DRS4_2_5GHz        = 1L,
    CAEN_DGTZ_DRS4_1GHz          = 2L,
    CAEN_DGTZ_DRS4_750MHz        = 3L,
    _CAEN_DGTZ_DRS4_COUNT_       = 4L,
} CAEN_DGTZ_DRS4Frequency_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
frequency/*frequency	Input (Set)/Output (Get)	The sampling frequency value to set/get, defined as a CAEN_DGTZ_DRS4Frequency_t type

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetOutputSignalMode

Description

Sets/gets the signal to be provided out over the TRG-OUT output channel in the x742 series.

Note: to be used only with x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetOutputSignalMode (int handle,
                                CAEN_DGTZ_OutputSignalMode_t mode
                               );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetOutputSignalMode (int handle,
                                CAEN_DGTZ_OutputSignalMode_t *mode
                               );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_TRIGGER             = 0L,
    CAEN_DGTZ_FASTTRG_ALL         = 1L,
    CAEN_DGTZ_FASTTRG_ACCEPTED    = 2L,
    CAEN_DGTZ_BUSY                = 3L,
} CAEN_DGTZ_OutputSignalMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	The output signal mode to set/get, defined as a CAEN_DGTZ_OutputSignalMode_t type.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

4 Acquisition

Set / GetChannelEnableMask

Description

This function enables/disables the channels for the acquisition. Disabled channels don't give any trigger and don't participate to the event data.

For the x740, x742 and x743 series, use the **Set / GetGroupEnableMask** function.



Note for DPP Firmware: *SetChannelEnableMask()* should be called before the **SetDPPEventAggregation** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelEnableMask(int handle,
                               uint32_t mask
                              );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelEnableMask(int handle,
                               uint32_t *mask
                              );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mask/*mask	Input (Set)/Output (Get)	Channel Enable Mask. Bit n corresponds to channel n. Please, refer to the User Manual of the specific board for the allowed number of channels

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetGroupEnableMask

Description

This function enables/disables the groups for the acquisition. This function is valid only for the x740, x742 and x743 series. Disabled groups don't give any trigger and don't participate to the event data. The 8 channels (for x740 and x742) or 2 channels (for x743) in a group are all enabled/disabled according to the relevant bit in the enable mask.



Note: to be used only with x740 and x742 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetGroupEnableMask(int handle,
                              uint32_t mask
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetGroupEnableMask(int handle,
                              uint32_t *mask
                             );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mask/*mask	Input (Set)/Output (Get)	Group Enable Mask. Bit n corresponds to group n. Please, refer to the User Manual of the specific board for the allowed number of groups.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

SWStartAcquisition

Description

This function starts the acquisition in a board using a software command. When the acquisition starts, the relevant RUN LED on the front panel lights up. It is worth noticing that in case of multiple board systems, the software start doesn't allow the digitizer to start synchronously. For this purpose, it is necessary to use to start the acquisition using a physical signal, such as the S-IN or GPI as well as the TRG-IN-TRG-OUT Daisy chain. Please refer to Digitizer manual for more details on this issue.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API  
CAEN_DGTZ_SWStartAcquisition(int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

SWStopAcquisition

Description

This function stops the acquisition in a board using a software command.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API  
CAEN_DGTZ_SWStopAcquisition(int handle);
```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetRecordLength

Description

This function sets the size of the acquisition window, that is the number of samples that belong to it. Due to the way the samples are written into the memory (more samples are put in parallel), there is a specific granularity of the record length depending on the board model. For example, in the x720 series, the samples are written 4 by 4, hence the record length must be a multiple of 4. Please, refer to the User Manual of the specific board for the granularity value. The function accepts any value for the parameter size and then takes the closest value multiple of the granularity. The function **GetRecordLength** returns the exact value.



Note: each time the record length is changed, the post-trigger must be updated (through the *SetPostTriggerSize*).



Note for DPP Firmware: *SetRecordLength()* should be called before the **SetDPPEventAggregation** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetRecordLength (int handle,
                          uint32_t size,
                          ...
                          );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetRecordLength (int handle,
                          uint32_t *size,
                          ...
                          );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
size/*size	Input (Set)/Output (Get)	The size of the record (in samples) to set/get. Note: in case of x743, the allowed sizes are those for which: “size mod 16 = 0”; the minimum accepted size is: size > 4*16
channel (optional)	Input	INT value corresponding to the channel index. To be used only with digitizers running DPP-PSD and DPP-PHA firmware. This parameter is not supported in DPP-CI firmware. Note: for the DPP-PSD and DPP-PHA firmware of 730 family, the Set function is managed inside couples of channels, i.e. inside channel 0 and channel 1, channel 2 and channel 3, etc. It is only possible to set the record length for the even channel of the couple, then the same value is automatically applied for the odd channel of the couple.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetPostTriggerSize

Description

This function sets the post trigger size, that is the position of the trigger within the acquisition window. The size is expressed in percentage of the record length. 0% means that the trigger is at the end of the window, while 100% means that it is at the beginning.



Note: the post-trigger must be updated each time the record length is changed (through the *SetRecordLength*).

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetPostTriggerSize(int handle,
                             uint32_t percent
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetPostTriggerSize(int handle,
                             uint32_t *percent
                             );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
percent/*percent	Input (Set)/Output (Get)	Post trigger to set/get (expressed in percent of the record length)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetAcquisitionMode

Description

Gets/Sets digitizer acquisition mode.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetAcquisitionMode(int handle,
                             CAEN_DGTZ_AcqMode_t mode
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetAcquisitionMode(int handle,
                             CAEN_DGTZ_AcqMode_t *mode
                             );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_SW_CONTROLLED          = 0,
    CAEN_DGTZ_S_IN_CONTROLLED        = 1,
    CAEN_DGTZ_FIRST_TRG_CONTROLLED   = 2
}CAEN_DGTZ_AcqMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	<p>The acquisition mode, defined as a CAEN_DGTZ_AcqMode_t type.</p> <p>CAEN_DGTZ_SW_CONTROLLED = 0. Start and stop acquisition is issued by software command.</p> <p>CAEN_DGTZ_S_IN_CONTROLLED = 1. Acquisition starts when the external signal on S-IN connector sets high, while is stopped when it sets low. Instead of S-IN. GPI connector must be referred in case of Desktop/NIM boards.</p> <p>CAEN_DGTZ_FIRST_TRG_CONTROLLED = 2. Start is issued on the first trigger pulse (rising edge) on the TRG-IN connector. This pulse is not used as a trigger; actual triggers start from the second pulse on TRG-IN. The Stop acquisition must be SW controlled.</p> <p>Please refer to the digitizer documentation for details.</p>

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetChannelDCOffset

Description

This function sets the 16-bit DAC that adds a DC offset to the input signal to adapt it to the dynamic range of the ADC. By default, the DAC is set to middle scale (0x7FFF) which corresponds to a DC offset of $-V_{pp}/2$, where V_{pp} is the voltage range (peak to peak) of the ADC. This means that the input signal can range from $-V_{pp}/2$ to $+V_{pp}/2$. If the DAC is set to 0x0000, then no DC offset is added, and the range of the input signal goes from $-V_{pp}$ to 0. Conversely, when the DAC is set to 0xFFFF, the DC offset is $-V_{pp}$ and the range goes from 0 to $+V_{pp}$. The DC offset can be set on channel basis except for the x740 in which it is set on group basis; in this case, you must use the **Set / GetGroupDCOffset** functions.



Note: from AMC FPGA firmware release 0.10 on, it is possible to apply an 8-bit positive digital offset individually to each channel inside a group of the x740 digitizer to finely correct the baseline mismatch. This function is not supported by the CAENDigitizer library, but the user can refer the registers documentation.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelDCOffset(int handle,
                             uint32_t channel,
                             uint32_t Tvalue
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelDCOffset(int handle,
                             uint32_t channel,
                             uint32_t *Tvalue
                             );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channel	Input	INT value corresponding to the channel index
Tvalue/*Tvalue	Input (Set)/Output (Get)	DAC value (from 0x0000 to 0xFFFF)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetGroupDCOffset

Description

The same as Set/Get ChannelDCOffset, but in this case, it is applied to the groups of the x740 series.



Note: to be used only with x740 series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetGroupDCOffset(int handle,
                             uint32_t group,
                             uint32_t Tvalue
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetGroupDCOffset(int handle,
                             uint32_t group,
                             uint32_t *Tvalue
                             );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
group	Input	INT value corresponding to the group index
Tvalue/*Tvalue	Input (Set)/Output (Get)	DAC value (from 0x0000 to 0xFFFF)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetDESMODE

Description

This function enables or disables the Dual Edge Sampling mode, that is the channel interleaving option to double the sampling frequency. This option is available in the x731 and x751 series only.



WARNING: when the DES mode is enabled, only the odd channels (for the x751) or the even channels (for the x731) will work; the other channels must be left unconnected

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDESMODE(int handle,
                     CAEN_DGTZ_EnaDis_t mode);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDESMODE(int handle,
                     CAEN_DGTZ_EnaDis_t *mode);

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_ENABLE    = 1,
    CAEN_DGTZ_DISABLE   = 0,
} CAEN_DGTZ_EnaDis_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	CAEN_DGTZ_EnaDis_t type defining the DES mode enable flag

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetDecimationFactor

Description

This function configures or reads the decimation factor for the Decimation function supported only by x740 and x724 digitizer series running the waveform recording firmware.



Note: for the Decimation of the DPP firmware, please refer to Chap. 6.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDecimationFactor(int handle,
                              uint16_t factor,
                              );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDecimationFactor(int handle,
                              uint16_t *factor,
                              );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
factor/*factor	Input (Set)/Output (Get)	Decimation factor "n". Allowed values: 1, 2, 4, 8, 16, 32, 64, 128. The relevant sampling frequency is then: Nominal Frequency / 2 ⁿ

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetZeroSuppressionMode

Description

This function sets/gets the Zero Suppression mode.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetZeroSuppressionMode(int handle,
                                   CAEN_DGTZ_ZS_Mode_t mode
                                   );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetZeroSuppressionMode (int handle,
                                   CAEN_DGTZ_ZS_Mode_t *mode
                                   );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_ZS_NO           = 0,
    CAEN_DGTZ_ZS_INT          = 1,
    CAEN_DGTZ_ZS_ZLE          = 2,
    CAEN_DGTZ_ZS_AMP          = 3,
} CAEN_DGTZ_ZS_Mode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	Zero Suppression Mode, defined as a CAEN_DGTZ_ZS_Mode_t type: CAEN_DGTZ_ZS_NO = 0 (no Zero suppression), CAEN_DGTZ_ZS_INT = 1 (Full Suppression based on the integral of the signal), CAEN_DGTZ_ZS_ZLE = 2 (Zero Length Encoding), CAEN_DGTZ_ZS_AMP = 3 (Full Suppression based on the signal amplitude),

Supported digitizers and permitted zero suppression modes

Digitizer	0	1	2	3
x720	X		X	X
v1721/v1731	X		X	X
x724	X	X	X	X

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetChannelZSParams

Description

Sets/Gets Zero Suppression parameters for a specific channel in the supported digitizers (see the table in the **Set / GetZeroSuppressionMode** functions).



Note: the **Set / GetChannelZSParams** functions are to be used in combination with **Set / GetTriggerPolarity** and **Set / GetZeroSuppressionMode** functions which relate to the trigger polarity logic and the zero suppression algorithm.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelZSParams(int handle,
                             uint32_t channel,
                             CAEN_DGTZ_ThresholdWeight_t weight,
                             int32_t threshold,
                             int32_t nsamp
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelZSParams(int handle,
                             uint32_t channel,
                             CAEN_DGTZ_ThresholdWeight_t *weight,
                             int32_t *threshold,
                             int32_t *nsamp
                             );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_ZS_FINE      = 0,
    CAEN_DGTZ_ZS_COARSE    = 1,
}CAEN_DGTZ_ThresholdWeight_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channel	Input	INT value corresponding to the channel index. Use -1 for all channels
weight/*weight	Input (Set)/Output (Get)	Zero Suppression weight*. Used in “Full Suppression based on the integral of the signal” supported only by x724 series. CAEN_DGTZ_ZS_FINE = 0 (Fine threshold step; the threshold is the <i>threshold</i> parameter), CAEN_DGTZ_ZS_COARSE = 1 (Coarse threshold step; the threshold is <i>threshold</i> × 64) For “Full Suppression based on the signal amplitude” and “Zero Length Encoding” algorithms, the value of <i>weight</i> doesn’t affect the function working.
threshold/*threshold	Input (Set)/Output (Get)	Zero Suppression Threshold to set/get depending on the ZS algorithm*.
nsamp/*nsamp	Input (Set)/Output (Get)	Number of samples of the ZS algorithm to set/get*.

*Refer to the digitizer User Manual for definition and representation.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetAnalogMonOutput

Description

Sets/Gets the signal to output on the Analog Monitor Front Panel output in VME digitizers running the waveform recording firmware for waves digitizing.



Note: this function is not supported by V1742, V1743, and any digitizer when running a DPP firmware.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetAnalogMonOutput(int handle,
                             CAEN_DGTZ_AnalogMonitorOutputMode_t mode
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetAnalogMonOutput(int handle,
                             CAEN_DGTZ_AnalogMonitorOutputMode_t *mode
                             );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_AM_TRIGGER_MAJORITY    = 0,
    CAEN_DGTZ_AM_TEST                = 1,
    CAEN_DGTZ_AM_ANALOG_INSPECTION   = 2,
    CAEN_DGTZ_AM_BUFFER_OCCUPANCY    = 3,
    CAEN_DGTZ_AM_VOLTAGE_LEVEL       = 4,
}CAEN_DGTZ_AnalogMonitorOutputMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	Analog Monitor Mode defined as CAEN_DGTZ_AnalogMonitorOutputMode_t type

Supported digitizers and permitted AM modes

Digitizer	0	1	2	3	4
V1720-V1721-V1731-V1740-V1743-V1751	X	X		X	X
V1724	X	X	X	X	X

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetAnalogInspectionMonParams

Description

Sets/Gets the Analog Inspection Monitor parameters for a V1724 digitizer running the waveform recording firmware for waves digitizing.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetAnalogInspectionMonParams(int handle,
                                       uint32_t channelmask,
                                       uint32_t offset,
                                       CAEN_DGTZ_AnalogMonitorMagnify_t mf,
                                       CAEN_DGTZ_AnalogMonitorInspectorInverter_t ami
                                       );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetAnalogInspectionMonParams(int handle,
                                       uint32_t channelmask,
                                       uint32_t *offset,
                                       CAEN_DGTZ_AnalogMonitorMagnify_t *mf,
                                       CAEN_DGTZ_AnalogMonitorInspectorInverter_t *ami
                                       );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_AM_MAGNIFY_1X          = 0,
    CAEN_DGTZ_AM_MAGNIFY_2X          = 1,
    CAEN_DGTZ_AM_MAGNIFY_4X          = 2,
    CAEN_DGTZ_AM_MAGNIFY_8X          = 3,
}CAEN_DGTZ_AnalogMonitorMagnify_t;

typedef enum
{
    CAEN_DGTZ_AM_INSPECTORINVERTER_P_1X = 0,
    CAEN_DGTZ_AM_INSPECTORINVERTER_N_1X = 1,
}CAEN_DGTZ_AnalogMonitorInspectorInverter_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channelmask	Input	Channel enable mask
offset/*offset	Input (Set)/Output (Get)	DC Offset for the analog output signal
mf/*mf	Input (Set)/Output (Get)	Multiply factor (see definition of CAEN_DGTZ_AnalogMonitorMagnify_t)
ami/*ami	Input (Set)/Output (Get)	Invert Output (see definition of CAEN_DGTZ_AnalogMonitorInspectorInverter_t)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetEventPackaging

Description

This function allows to enable or disable the Pack 2.5 mode of V1720/DT5720 Digitizers

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetEventPackaging(int handle,
                             CAEN_DGTZ_EnaDis_t mode
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetEventPackaging(int handle,
                             CAEN_DGTZ_EnaDis_t *mode
                             );

//-----
//Types Definition
//-----
typedef enum {
                                CAEN_DGTZ_ENABLE = 1L,
                                CAEN_DGTZ_DISABLE = 0L,
} CAEN_DGTZ_EnaDis_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/Output (Get)	Pack 2.5 mode enable flag

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Waveform Recording Firmware Example Code

CAEN provides an example code, compliant to digitizers running the waveform recording firmware, which is included in the library packet together with source C files and Visual Studio project (**compliant with Visual Studio Professional 2010**). This example can be used as **base for users who want to develop their own software to control digitizers running the waveform recording firmware**.

In Windows systems, the sample code is in the “Samples” subfolder of “CAEN” main directory (see **Fig. 4.1**).

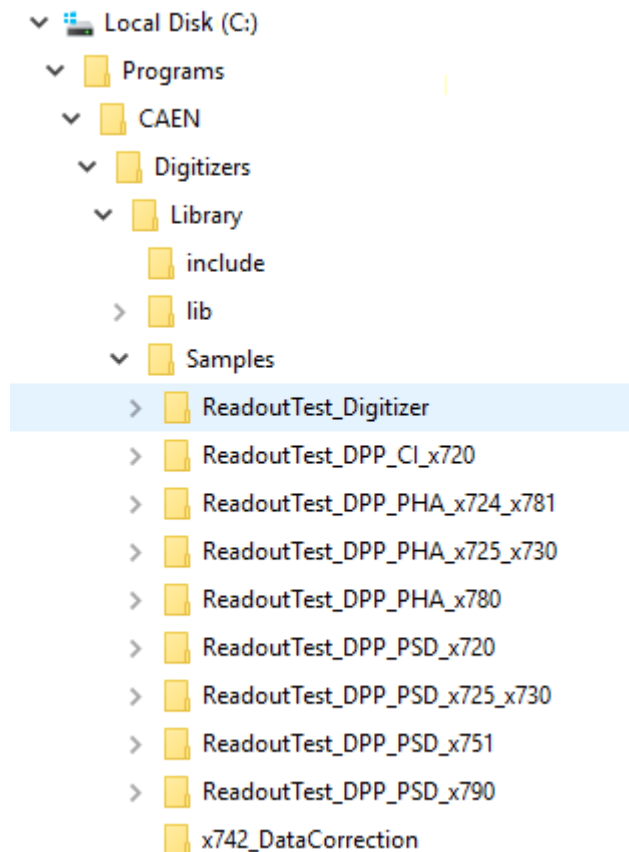


Fig. 4.1: Acquisition example location in Windows OS

In Linux systems, the sample code is in the “samples” subfolder of the library main directory. The examples can be compiled under Linux OS typing ‘make’ in the chosen example directory. The executable file is then accessible in the “bin” subfolder.

To give a brief example for Windows OS, we can perform a readout test using a CAEN **DT5724 Digitizer** running a **Waveform Recording firmware**, connected through a direct USB connection to the computer. We used the CAEN DT5800 Digital Detector Emulator to send exponential signals to ch 0.

Compile the Visual Studio project in the *ReadoutTest_Digitizer/build* subfolder and run the compiled code. The example demo opens the following window:

```
Connected to CAEN Digitizer Model DT5724, recognized as board 0
ROC FPGA Release is 04.14 - Build 1407
AMC FPGA Release is 00.14 - Build 1125

Press 's' to start the acquisition
Press 'k' to stop the acquisition
Press 'q' to quit the application
_
```

Type ‘s’ to start the acquisition. A series of dots typed continuously in the command shell will appear. Typing ‘k’ will stop the acquisition and the **total number of retrieved events** will appear.

After stopping the acquisition, the only available option is to quit the application pressing 'Enter'.

```
Connected to CAEN Digitizer Model DT5724, recognized as board 0
ROC FPGA Release is 04.14 - Build 1407
AMC FPGA Release is 00.14 - Build 1125
```

```
Press 's' to start the acquisition
Press 'k' to stop the acquisition
Press 'q' to quit the application
```

```
.....
.....
.....
```

```
Board 0: Retrieved 387 Events
Press 'Enter' key to exit
```

x742 Offline Data Correction Functions

In the installation package of CAENDigitizer library, additional functions are provided inside the “Sample” folder to let the user perform offline the correction of raw data acquired with x742 digitizers.



Note: the functions are not included in the CAENDigitizer run time library and are intended also for offline use.

ApplyDataCorrection

Description

Applies the desired correction data (through a correction mask) to the raw data acquired by the user.

Synopsis

```
void ApplyDataCorrection(DataCorrection_t *CTable,
                        CAEN_DGTZ_DRS4Frequency_t frequency,
                        int CorrectionLevelMask,
                        CAEN_DGTZ_X742_GROUP_t *data
                        );

//-----
//Types Definition
//-----
See LoadCorrectionTable and LoadDRS4CorrectionData
```

Arguments

Name	I/O	Description
*CTable	Input	Pointer to the table containing the data corrections
frequency	Input	DSR4 sampling frequency
CorrectionLevelMask	Input	Mask for the correction type to be applied (3-bit): Bit0 = Cell Offset correction Bit1 = Index Sampling correction Bit2= Time correction
*data	Input	Pointer to the raw acquired data to be corrected

Return Values

0: Success

SaveCorrectionTables

Description

Writes the correction table of a x742 board into an output file

Synopsis

```
int SaveCorrectionTables(char *outputFileName,
                        uint32_t groupMask,
                        CAEN_DGTZ_DRS4Correction_t *tables
                        );

//-----
//Types Definition
//-----
//See GetCorrectionTables
```

Arguments

Name	I/O	Description
*outputFileName	Input	Pointer to the output filename
groupMask	Input	Group mask of tables to be saved
*tables	Output	Pointer to the data correction group tables

Return Values

0: Success.

LoadCorrectionTable

Description

Reads the correction table of a x742 board from .txt files

Synopsis

```
int LoadCorrectionTable(char *baseInputFileName,
                       CAEN_DGTZ_DRS4Correction_t *tb
                       );
//-----
//Types Definition
//-----
//See GetCorrectionTables
```

Arguments

Name	I/O	Description
*baseInputFileName	Input	Base Filename of the input file. Actual filenames loaded will be: a) baseInputFileName + "_cell.txt" b) baseInputFileName + "_nsample.txt" c) baseInputFileName + "_time.txt"
*tb	Input	Pointer to the data correction table

Return Values

0: Success.

GetNumEvents

Description

Gets the current number of events stored in the acquisition buffer.

Synopsis

```
int32_t GetNumEvents(char *buffer,
                    uint32_t buffsize,
                    uint32_t *numEvents
                    );
```

Arguments

Name	I/O	Description
*buffer	Input	Address of the acquisition buffer
buffsize	Input	Size of the data stored in the acquisition buffer
*numEvents	Output	Number of events stored in the acquisition buffer

Return Values

0: Success.

GetEventPtr

Description

Retrieves the event pointer of a specified event in the acquisition buffer.

Synopsis

```
int32_t GetEventPtr(char *buffer,
                   uint32_t buffsize,
                   int32_t numEvent,
                   char **EventPtr
                   );
```

Arguments

Name	I/O	Description
*buffer	Input	Address of the acquisition buffer
buffersize	Input	Acquisition buffer size
numEvent	Input	Index of the requested event
**EventPtr	Output	Pointer to the requested event in the acquisition buffer

Return Values

0: Success.

X742_DecodeEvent

Description

Decodes a specified event stored in the acquisition buffer writing data in Evt memory.



Note: once used, the Evt memory MUST be deallocated by the caller.

Synopsis

```
int32_t X742_DecodeEvent(char *evtPtr,
                        void **Evt
                        );
```

Arguments

Name	I/O	Description
*evtPtr	Input	Pointer to the requested event in the acquisition buffer (MUST BE NULL)
**Evt	Output	Pointer to the event structure with the decoded event data

Return Values

0: Success.

5 x743 Specific Functions

This paragraph describes the CAENDigitizer functions that specifically apply to the digitizers in the 743 series and are not to be used with other digitizer series. A set of the main functionalities of the board can be managed, like the acquisition, the calibrations and the test pulse generation.

Note that the SAM acronym used in some of the described functions corresponds to the SAMLONG Sampling Analog Memory the x743 boards are based on, housing two channels (i.e. one SAMLONG chip houses 2 channels).

Each SAMLONG chip in an N-channel board is indexed between 0 and $N/2 - 1$. This index is called *SamIndex* and is used in some of the functions below to define parameters that are common to the group of two channels housed in the same chip. (The corresponding channel numbers are $2*SamIndex$ and $2*SamIndex + 1$).

For example, in a 16-channel digitizer board, the SAMLONG chip with *SamIndex* = 3 is housing channels 6 and 7.

Set / GetSAMCorrectionLevel

Description

These functions allow to set and get the configuration of the different types of data correction required by the x743, to compensate for unavoidable construction differences among the SAMLONG chips (refer to the board User Manual).

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetSAMCorrectionLevel(
    int handle,
    CAEN_DGTZ_SAM_CORRECTION_LEVEL_t level
);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetSAMCorrectionLevel(
    int handle,
    CAEN_DGTZ_SAM_CORRECTION_LEVEL_t *level
);

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_SAM_CORRECTION_DISABLED      = 0,
    CAEN_DGTZ_SAM_CORRECTION_PEDESTAL_ONLY = 1,
    CAEN_DGTZ_SAM_CORRECTION_INL          = 2,
    CAEN_DGTZ_SAM_CORRECTION_ALL          = 3,
} CAEN_DGTZ_SAM_CORRECTION_LEVEL_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
level / *level	Input (Set)/Output (Get)	The value of (or the pointer to, in case of Get) the <i>CAEN_DGTZ_SAM_CORRECTION_LEVEL_t</i> structure indicating the correction setting to program. See the type definition above.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetSAMPostTriggerSize

Description

These functions allow to set and get the post-trigger delay value (with respect to the actual trigger signal) which provokes the freezing of the currently stored signal in the sampling capacitance cells of the SAMLONG chip.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetSAMPostTriggerSize(
    int handle,
    int SamIndex,
    uint8_t value
);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetSAMPostTriggerSize(
    int handle,
    int SamIndex,
    int32_t *value
);
```

Arguments

Name	I/O	Description
handle	Input	Device handler
SamIndex	Input	Index of the SAMLONG chip that is housing the channels : $2*SamIndex$ and $2*SamIndex + 1$. See description above.
value / *value	Input (Set)/Output (Get)	Value (range between 1 and 255) of the post-trigger delay (pointer to, in case of Get) . Unit is the sampling period multiplied by 16.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetSAMSamplingFrequency

Description

These functions allow to set and get the sampling frequency of the SAMLONG chip.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAENDGTZ_API CAEN_DGTZ_SetSAMSamplingFrequency(
    int handle,
    CAEN_DGTZ_SAMFrequency_t frequency
);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetSAMSamplingFrequency(
    int handle,
    CAEN_DGTZ_SAMFrequency_t *frequency
);

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_SAM_3_2GHz      = 0L,
    CAEN_DGTZ_SAM_1_6GHz      = 1L,
    CAEN_DGTZ_SAM_800MHz      = 2L,
    CAEN_DGTZ_SAM_400MHz      = 3L,
} CAEN_DGTZ_SAMFrequency_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
frequency / *frequency	Input (Set)/Output (Get)	The value of (or the pointer to, in case of Get) the <i>CAEN_DGTZ_SAMFrequency_t</i> structure indicating the SAMLONG sampling frequency. See the type definition above.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Read_EEPROM

Description

This function allows to read data from the on-board EEPROM where various information about the daughterboard is stored (See the **Data Correction** paragraph in the User Manual of the x743 board).

Synopsis

```

ErrorCode CAENDGTZ_API
_CAEN_DGTZ_Read_EEPROM(
    int handle,
    int EEPROMIndex,
    unsigned short add,
    int nbOfBytes,
    unsigned char* buf
);

```

Arguments

Name	I/O	Description
handle	Input	Device handler
EEPROMIndex	Input	Corresponds to the Index of the daughterboard that houses 2 SAMLONG chips, which corresponds to 4 channels.
add	Input	Address in the EEPROM to access (range value between 0 and 65535)
nbOfBytes	Input	Number of Bytes to read
*buf	Output	Returned buffer of the read values (in bytes)

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

LoadSAMCorrectionData

Description

This function loads all the calibrations values present in the on-board EEPROMs as Individual Pedestal correction values, Time INL Corrections values, Trigger Thresholds DAC Offset calibrations values and Line Offset calibrations.

Synopsis

```

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_LoadSAMCorrectionData(
    int handle
);

```

Arguments

Name	I/O	Description
handle	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Enable / DisableSAMPulseGen

Description

These functions allow to enable and disable the generation of the test pulses from the individual pulser each input channel is equipped with (refer to the board User Manual).



Note: `pulseSource` is common to each pair of channels sharing the same SAMLONG chip.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_EnableSAMPulseGen(
    int handle,
    int channel,
    unsigned short pulsePattern,
    CAEN_DGTZ_SAMPulseSourceType_t pulseSource
);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_DisableSAMPulseGen(
    int handle,
    int channel
);

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_SAMPulseSoftware = 0, //a pulse will be sent to the enabled inputs using
                                     //SendSAMPulse() function
    CAEN_DGTZ_SAMPulseCont     = 1, //pulse are sent continuously from the FPGA to the
                                     //enabled inputs using the internal oscillator
} CAEN_DGTZ_SAMPulseSourceType_t;
```

Arguments

Name	I/O	Description
<code>handle</code>	Input	Device handler
<code>channel</code>	Input	INT value corresponding to the channel index
<code>pulsePattern</code>	Input	Pulse pattern value. It is a 16-bit word. Example 1: if <code>pulsePattern = 1</code> , the generated pattern is a single pulse of 1-clock duration (e.g. 5 ns @ 3.2 GS/s). Example 2: if <code>pulsePattern = AAAA</code> , the generated pattern is the sequence of 1010101010101010, where each 1 is a pulse of 1-clock duration.
<code>pulseSource</code>	Input	The value of the <code>CAEN_DGTZ_SAMPulseSourceType_t</code> structure. See the type definition above.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

SendSAMPulse

Description

This function allows to send a single pulse from the FPGA to the enabled channels (see **Enable / DisableSAMPulseGen**) *but only* if the `pulseSource` for the selected channel is set to `CAEN_DGTZ_SAMPulseSoftware`.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SendSAMPulse(
    int handle
);
```

Arguments

Name	I/O	Description
<code>handle</code>	Input	Device handler

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetSAMAcquisitionMode

Description

These functions allow to set and get the acquisition mode of the x743 digitizer, selecting between the digital oscilloscope and the embedded charge integration mode.



Note: the x743 firmware features a charge integration acquisition mode here referred to as DPP-CI, but it must not be confused with the special DPP-CI firmware supported by the 720 digitizer series.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetSAMAcquisitionMode(
    int handle,
    CAEN_DGTZ_AcquisitionMode_t mode
);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetSAMAcquisitionMode(
    int handle,
    CAEN_DGTZ_AcquisitionMode_t *mode
);

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_AcquisitionMode_STANDARD = 0, //Digital oscilloscope mode
    CAEN_DGTZ_AcquisitionMode_DPP_CI   = 1, //Charge integration mode - charge data are
                                           //expressed in pC
} CAEN_DGTZ_AcquisitionMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode / *mode	Input (Set)/Output (Get)	The value of (or the pointer to, in case of Get) the <i>CAEN_DGTZ_AcquisitionMode_t</i> structure indicating the acquisition mode. See type definition above.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetChannelPairTriggerLogic

Description

These functions allow to set and get the trigger logic for the generation of the trigger requests from the self-trigger signals coming from each pair of channels of the x743 digitizers.

For a generic couple ChannelA-ChannelB (i.e. CH0-CH1, or CH2-CH3 and so on), the AND/OR trigger requests are allowed.

The trigger requests from each channel pair can be further processed by another trigger logic level (see **Set / GetTriggerLogic** function) to generate the board common trigger (i.e. the event storing).



Note: please, consult the x743 User Manual for details about the trigger management.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelPairTriggerLogic(int handle,
                                     uint32_t channelA,
                                     uint32_t channelB,
                                     CAEN_DGTZ_TriggerLogic_t logic,
                                     uint16_t coincidenceWindow
                                    );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelPairTriggerLogic(int handle,
                                     uint32_t channelA,
                                     uint32_t channelB,
                                     CAEN_DGTZ_TriggerLogic_t *logic,
                                     uint16_t *coincidenceWindow
                                    );

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_LOGIC_OR          = 0, //The trigger is the OR of the self-trigger signals
                                //from the pair
    CAEN_DGTZ_LOGIC_AND        = 1, //The trigger is the AND of the self-trigger signals
                                //from the pair
} CAEN_DGTZ_TriggerLogic_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channelA / channelB	Input	The number of the channels in a pair. A and B must belong to the same couple (CH0-CH1; CH2-CH3; ...)
logic / *logic	Input (Set)/Output (Get)	The value (or the pointer to, in case of Get) of the <i>CAEN_DGTZ_TriggerLogic_t</i> structure, defining the trigger logic mode (AND / OR). See type definition above
coincidenceWindow / *coincidenceWindow	Input (Set)/Output (Get)	The coincidence gate (in ns). It corresponds to the <i>Primitives Gate Length</i> parameter of the WaveCatcher software (see the software User Manual). Note: it must be ≥ 15 ns. (it should be a multiple of 5 ns also; otherwise, the library will put the closer multiple of 5 as gate length). Maximum value is $5 \cdot 255 = 1275$ ns.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetTriggerLogic

Description

Sets/gets the trigger logic configuration to process, at the motherboard level, the trigger requests generated by the enabled channel pairs according to the **Set / GetChannelPairTriggerLogic** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAENDGTZ_API CAEN_DGTZ_SetTriggerLogic(int handle,
                                         CAEN_DGTZ_TrigerLogic_t logic,
                                         uint32_t majorityLevel
                                         );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAENDGTZ_API CAEN_DGTZ_GetTriggerLogic(int handle,
                                         CAEN_DGTZ_TrigerLogic_t *logic,
                                         uint32_t *majorityLevel
                                         );

//-----
//Types Definition
//-----
typedef enum {
    CAEN_DGTZ_LOGIC_OR          = 0, //Common trigger is the OR of
                                   //the channel pairs
    CAEN_DGTZ_LOGIC_AND        = 1, //common trigger is the AND
                                   //of the channel pairs
    CAEN_DGTZ_LOGIC_MAJORITY   = 2, //Common trigger is given by
                                   //the majority policy upon the
                                   //value of the majority Level
                                   //parameter
} CAEN_DGTZ_TrigerLogic_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
logic/*logic	Input (Set)/Output (Get)	The trigger logic to set/get according to the <i>CAEN_DGTZ_TrigerLogic_t</i> structure. See type definition above
majorityLevel/ *majorityLevel	Input (Set)/Output (Get)	Value of the majority level. Allowed values range between 0 and (max num. channel – 1). “0” means more than 0, i.e. ≥ 1.

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

Set / GetSAMTriggerCountVetoParam

Description

This function enables/disables the trigger counter veto, and programs the time window for the veto.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetSAMTriggerCountVetoParam(
    int handle,
    int channel,
    CAEN_DGTZ_EnaDis_t enable,
    uint32_t vetoWindow
);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetSAMTriggerCountVetoParam(
    int handle,
    int channel,
    CAEN_DGTZ_EnaDis_t *enable,
    uint32_t *vetoWindow
);

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_ENABLE           = 1, //Veto is enabled
    CAEN_DGTZ_DISABLE         = 0, //Veto is disabled
} CAEN_DGTZ_EnaDis_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channel	Input	INT value corresponding to the channel index
enable/*enable	Input (Set)/Output (Get)	Veto enable/disable flag according to the <i>CAEN_DGTZ_EnaDis_t</i> structure. See definition above.
vetoWindow/ *vetoWindow	Input (Set)/Output (Get)	Time window for veto duration expressed in ns

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

6 DPP Specific Functions

To handle acquisitions with the DPP firmware (PHA, PSD, QDC, ZLEplus, DAW, CI), the C functions described in this chapter can be used.

DPP codes

Description

Define DPP firmware code

Synopsis

```
#define STANDARD_FW_CODE      (0x00) // In case of waveform recording firmware
#define V1724_DPP_PHA_CODE    (0x80) // In case of the DPP-PHA for x724 boards
#define V1720_DPP_CI_CODE     (0x82) // In case of the DPP-CI for x720 boards
#define V1720_DPP_PSD_CODE    (0x83) // In case of the DPP-PSD for x720 boards
#define V1751_DPP_PSD_CODE    (0x84) // In case of the DPP-PSD for x751 boards
#define V1751_DPP_ZLE_CODE    (0x85) // In case of the DPP-ZLE for x751 boards
#define V1743_DPP_CI_CODE     (0x86) // In case of the DPP-PSD for x743 boards
#define V1740_DPP_QDC_CODE    (0x87) // In case of the DPP-QDC for x740D boards
#define V1730_DPP_PSD_CODE    (0x88) // In case of the DPP-PSD for x730 boards
#define V1730_DPP_PHA_CODE    (0x8B) // In case of the DPP-PHA for x730 boards
#define V1730_DPP_ZLE_CODE    (0x8C) // In case of the DPP-ZLE for x725/x730 boards
#define V1730_DPP_DAW_CODE    (0x8D) // In case of the DPP-DAW for x725/x730 boards
```

Set / GetDPPPreTriggerSize

Description

Sets/gets the pre-trigger size, which is the portion of acquisition window visible before a trigger.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPPPreTriggerSize (int handle,
                                int ch,
                                uint32_t samples
                                );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPPPreTriggerSize (int handle,
                                int ch,
                                uint32_t *samples
                                );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
ch	Input	The channel whose pre-trigger has to be set/get. ch= -1 writes the same value for all channels. DPP-CI only supports ch= -1 (different channels must have the same pre-trigger)
samples/*samples	Input (Set)/Output (Get)	The size (or the pointer to, in case of Get) of the pre-trigger (in samples)

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetChannelPulsePolarity

Description

Sets/gets the value of the pulse polarity for the specified channel.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetChannelPulsePolarity (int handle,
                                   uint32_t channel,
                                   CAEN_DGTZ_PulsePolarity_t pol
                                   );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetChannelPulsePolarity (int handle,
                                   uint32_t channel,
                                   CAEN_DGTZ_PulsePolarity_t *pol
                                   );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_PulsePolarityPositive = 0,
    CAEN_DGTZ_PulsePolarityNegative = 1,
} CAEN_DGTZ_PulsePolarity_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channel	Input	The channel to set/get information for
pol/*pol	Input (Set)/Output (Get)	Value (or the pointer to, in case of Get) of the pulse polarity

Return Values

0: Success; Negative numbers are error codes (see Return Codes).

GetDPPEvents

Description

Decodes and returns all the DPP events stored in the acquisition buffers.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPPEvents (int handle,
                        char *buffer,
                        uint32_t buffsize,
                        void **events,
                        uint32_t *numEventsArray
                        );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*buffer	Input	The address of the acquisition buffer
buffsize	Input	The acquisition buffer size (in samples)
**events	Output	The pointer to the event list (allocated via MallocDPPEvents)
*numEventsArray	Output	The pointer to an array of <i>int</i> which contains the number of events found per channel

Return Values

0: Success; negative numbers are error codes (see Return Codes).

MallocDPPEvents

Description

Allocates the event buffer matrix which is handled by the **GetDPPEvents** function. The matrix has one event array per channel and must be declared as a MAX_CH-sized array of pointers.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_MallocDPPEvents (int handle,
                          void **events,
                          uint32_t *allocatedSize
                          );

//-----
//Types Definition
//-----
typedef struct
{
    uint32_t Format;
    uint64_t TimeTag;
    uint16_t Energy;
    int16_t Extras;
    uint32_t Extras2;
    uint32_t *Waveforms; //Pointer to coded data inside the readout buffer. only meant to
                        //be supplied to DecodeDPPWaveforms function
} CAEN_DGTZ_DPP_PHA_Event_t;

typedef struct
{
    uint32_t Format;
    uint32_t Format2; //Enables the event format including the CFD information
                    //(751 family only)
    uint32_t TimeTag;
    int16_t ChargeShort;
    int16_t ChargeLong;
    int16_t Baseline;
    int16_t Pur;
    uint32_t *Waveforms; //Pointer to coded data inside the readout buffer. only meant to
                        //be supplied to DecodeDPPWaveforms function
    uint32_t Extras;
} CAEN_DGTZ_DPP_PSD_Event_t;

typedef struct
{
    uint32_t Format;
    uint32_t TimeTag;
    int16_t Charge;
    int16_t Baseline;
    uint32_t *Waveforms; //Pointer to coded data inside the readout buffer. only meant to
                        //be supplied to DecodeDPPWaveforms function
    uint32_t Extras
} CAEN_DGTZ_DPP_CI_Event_t;

typedef struct
{
    uint8_t isExtendedTimeStamp;
    uint32_t Format;
    uint64_t TimeTag;
    uint16_t Charge;
    int16_t Baseline;
    uint16_t Pur;
    uint16_t Overrange;
    uint16_t SubChannel;
    uint32_t *Waveforms; //Pointer to coded data inside the readout buffer. only meant to
                        //be supplied to DecodeDPPWaveforms function
    uint32_t Extras;
} CAEN_DGTZ_DPP_QDC_Event_t;

typedef struct
{
    uint32_t size;
    uint16_t chmask;
    uint32_t tcounter;
    uint64_t timeStamp;
```

```

    CAEN_DGTZ_730_DAW_Channel_t *Channel[MAX_V1730_CHANNEL_SIZE];
} CAEN_DGTZ_730_DAW_Event_t;

typedef struct
{
    uint32_t truncate;
    uint32_t EvType;
    uint32_t size;
    uint64_t timeStamp;
    uint16_t baseline;
    uint16_t *DataPtr;
    CAEN_DGTZ_730_DAW_Waveforms_t *Waveforms; //pointer to raw data for DPP-DAW, used for
                                              //uniformity with other firmware
} CAEN_DGTZ_730_DAW_Channel_t;

```

Arguments

Name	I/O	Description
handle	Input	Device handler
**events	Output	The pointer to the event matrix, which shall be of type: CAEN_DGTZ_DPP_PHA_Event_t, for DPP-PHA, CAEN_DGTZ_DPP_PSD_Event_t, for DPP-PSD CAEN_DGTZ_DPP_CI_Event_t, for DPP-CI CAEN_DGTZ_DPP_QDC_Event_t, for DPP-QDC CAEN_DGTZ_730_DAW_Event_t, for x725/x730 DPP-DAW Note: please refer to the DPP User Manual for the event format description
*allocatedSize	Output	The size in bytes of the event list

Return Values

0: Success; negative numbers are error codes (see Return Codes).

FreeDPPEvents

Description

Deallocates the event buffer matrix.

Synopsis

```

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_FreeDPPEvents (int handle,
                        void **events
                        );

```

Arguments

Name	I/O	Description
handle	Input	Device handler
**events	Input	The pointer to the event buffer

Return Values

0: Success; negative numbers are error codes (see Return Codes).

FreeDPPWaveforms

Description

Deallocates the waveform buffer.

Synopsis

```

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_FreeDPPWaveforms (int handle,
                          void *waveforms
                          );

```

Arguments

Name	I/O	Description
handle	Input	Device handler
*waveforms	Input	The pointer to the waveform buffer

Return Values

0: Success; negative numbers are error codes (see Return Codes).

MallocDPPWaveforms

Description

Allocates the waveform buffer, which is used by `CAEN_DGTZ_DecodeDPPWaveforms`.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_MallocDPPWaveforms (int handle,
                              void **waveforms,
                              uint32_t *allocatedSize
                              );

//-----
//Types Definition
//-----
typedef struct
{
    uint32_t Ns;
    uint8_t  DualTrace;
    uint8_t  VProbe1;
    uint8_t  VProbe2;
    uint8_t  VDProbe;
    int16_t  *Trace1;
    int16_t  *Trace2;
    uint8_t  *DTrace1;
    uint8_t  *DTrace2;
} CAEN_DGTZ_DPP_PHA_Waveforms_t;

typedef struct
{
    uint32_t Ns;
    uint8_t  dualTrace;
    uint8_t  anlgProbe;
    uint8_t  dgtProbe1;
    uint8_t  dgtProbe2;
    uint16_t *Trace1;
    uint16_t *Trace2;
    uint8_t  *DTrace1;
    uint8_t  *DTrace2;
    uint8_t  *DTrace3;
    uint8_t  *DTrace4;
} CAEN_DGTZ_DPP_PSD_Waveforms_t;

#define CAEN_DGTZ_DPP_CI_Waveforms_t CAEN_DGTZ_DPP_PSD_Waveforms_t
//Waveform types for DPP-CI and DPP-PSD are the same, hence this define

typedef struct
{
    uint32_t Ns;
    uint8_t  dualTrace;
    uint8_t  anlgProbe;
    uint8_t  dgtProbe1;
    uint8_t  dgtProbe2;
    uint16_t *Trace1;
    uint16_t *Trace2;
    uint8_t  *DTrace1;
    uint8_t  *DTrace2;
    uint8_t  *DTrace3;
    uint8_t  *DTrace4;
} CAEN_DGTZ_DPP_QDC_Waveforms_t;

typedef struct
{
    uint16_t *Trace;
} CAEN_DGTZ_730_DAW_Waveforms_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
**waveforms	Output	The pointer to the waveform buffer, which shall be of type: CAEN_DGTZ_DPP_PHA_Waveforms_t, for DPP-PHA CAEN_DGTZ_DPP_PSD_Waveforms_t, for DPP-PSD CAEN_DGTZ_DPP_CI_Waveforms_t, for DPP-CI

		CAEN_DGTZ_DPP_QDC Waveforms_t, for DPP-QDC CAEN_DGTZ_DPP_730 Waveforms_t, for x725/x730 DPP-DAW
*allocatedSize	Output	The size in bytes of the waveform buffer

Return Values

0: Success; negative numbers are error codes (see Return Codes).

DecodeDPPWaveforms

Description

Decodes the waveforms contained inside an event.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_DecomposeDPPWaveforms (int handle,
                                   void *event,
                                   void *waveforms
                                   );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*event	Input	The pointer to the event
*waveforms	Output	The pointer to the (preallocated) waveform list

Return Values

0: Success; negative numbers are error codes (see Return Codes).

SetDPPEventAggregation

Description

Sets the parameters related to the event aggregation.

Note: this function should be called only after any other function affecting the internal event aggregation, specifically:

- *SetRecordLength()*
- *SetChannelEnableMask()*
- *SetNumEventsPerAggregate()*
- *SetDPPAcquisitionMode()*



Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPPEventAggregation (int handle,
                                   int threshold,
                                   int maxsize
                                   );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
threshold	Input	Specifies how many events are accumulated in the board memory before they are available for readout. A low number maximizes responsiveness, since data is read as soon as it is stored in memory, while a higher number maximizes the efficiency, since fewer transfers are made. Supplying 0 lets the library choose the most reasonable value depending on the acquisition mode and other parameters.
maxsize	Input	Specifies the maximum size in bytes of the event buffer on the PC side. This parameter might be useful in case of computers with very low RAM to reduce the amount of memory used. In normal cases supply maxsize = 0 to let the library choose an appropriate value automatically.

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetNumEventsPerAggregate

Description

Sets/Gets the number of events that each aggregate will contain.



Note: *SetNumEventsPerAggregate()* should be called before the **SetDPPEventAggregation** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetNumEventsPerAggregate (int handle,
                                     uint32_t numEvents,
                                     ...
                                     );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetNumEventsPerAggregate (int handle,
                                     uint32_t *numEvents,
                                     ...
                                     );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
numEvents/ *numEvents	Input (Set)/ Output (Get)	Number of events per aggregate.
channel (optional)	Input	INT value corresponding to the channel index (required for DPP-PSD and DPP-CI, ignored by DPP-PHA). Note: for the DPP-PSD firmware of 730 family, the Set function is managed inside couples of channels, i.e. inside channel 0 and channel 1, channel 2 and channel 3, etc. It is only possible to set the record length for the even channel of the couple, then the same value is automatically applied for the odd channel of the couple.

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetMaxNumAggregatesBLT

Description

Sets/Gets the maximum number of aggregates for each transfer.



Note: with DPP-PHA, DPP-PSD, DPP-QDC or DPP-CI, also the *maxsize* parameter of **SetDPPEventAggregation** can be used.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetMaxNumAggregatesBLT (int handle,
                                    uint32_t numAggr
                                    );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetMaxNumAggregatesBLT (int handle,
                                    uint32_t *numAggr
                                    );
```

Arguments

Name	I/O	Description
handle	Input	Device handler
numAggr/*numAggr	Input (Set)/ Output (Get)	Max number of aggregates per block transfer (BLT)

Return Values

0: Success; negative numbers are error codes (see Return Codes).

SetDPPParameters

Description

Sets DPP configuration parameters for DPP-PHA, DPP-PSD, DPP-QDC and DPP-CI.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPPParameters (int handle,
                             uint32_t channelMask,
                             void *params
                             );

//-----
//Types Definition
//-----

/* DPP parameter structure to be initialized and passed to CAEN_DGTZ_SetDPPParameters.
To be used only for DPP-PHA */

typedef struct{
    int M      [MAX_DPP_PHA_CHANNEL_SIZE]; //Signal Decay Time Constant (ns)
    int m      [MAX_DPP_PHA_CHANNEL_SIZE]; //Trapezoid Flat Top (ns)
    int k      [MAX_DPP_PHA_CHANNEL_SIZE]; //Trapezoid Rise Time(ns)
    int ftd    [MAX_DPP_PHA_CHANNEL_SIZE]; //Flat Top Delay (ns)
    int a      [MAX_DPP_PHA_CHANNEL_SIZE]; //Trigger Filter smoothing factor, that is the
                                           //number of samples over which to perform the
                                           //smoothing;
                                           //options: 1, 2, 3, 4, 16, 32 samples

    int b      [MAX_DPP_PHA_CHANNEL_SIZE]; //Input Signal Rise time (ns)
    int thr    [MAX_DPP_PHA_CHANNEL_SIZE]; //Trigger Threshold (LSB)
    int nsbl   [MAX_DPP_PHA_CHANNEL_SIZE]; //Number of Samples for Baseline Mean
                                           //options:
                                           //    0 = 0
                                           //    1 = 16    samples
                                           //    2 = 64    samples
                                           //    3 = 256   samples
                                           //    4 = 1024  samples
                                           //    5 = 4096  samples
                                           //    6 = 16384 samples

    int nspk   [MAX_DPP_PHA_CHANNEL_SIZE]; //Number of Samples for Peak Mean
                                           //options:
                                           //    0 = 1 sample
                                           //    1 = 4 samples
                                           //    2 = 16 samples
                                           //    3 = 64 samples

    int pkho   [MAX_DPP_PHA_CHANNEL_SIZE]; //Peak Hold Off (ns)
    int blho   [MAX_DPP_PHA_CHANNEL_SIZE]; //Base Line Hold Off (ns)
    int otrej; //This parameter is deprecated
    int trgho  [MAX_DPP_PHA_CHANNEL_SIZE]; //Trigger Hold Off (ns)
    int twwdt  [MAX_DPP_PHA_CHANNEL_SIZE]; //Zero crossing acceptance window for the Rise
                                           //Time Discriminator (RTD) in the pile-up
                                           //rejection, starting from the RC-CR2
                                           //overthreshold which arms the acquisition
                                           //(when 0, the RTD is disabled)

    int trgwin [MAX_DPP_PHA_CHANNEL_SIZE]; //Trigger acceptance window in coincidence
    int dgain  [MAX_DPP_PHA_CHANNEL_SIZE]; //Digital Gain to apply to the input
                                           //digitized waveform
                                           //options:
                                           //    0 = Digital Gain is 1
                                           //    1 = Digital Gain is 2
                                           //    2 = Digital Gain is 4
                                           //    3 = Digital Gain is 8

    float enf  [MAX_DPP_PHA_CHANNEL_SIZE]; //Energy Normalization Factor
    int decimation [MAX_DPP_PHA_CHANNEL_SIZE]; //Number of samples over which to apply the
                                           //decimation of the input digitized waveform
                                           //options:
                                           //    0: Decimation disabled
                                           //    1: 2 samples (50 MSps)
                                           //    2: 4 samples (25 MSps)
                                           //    3: 8 samples (12.5 MSps)
} CAEN_DGTZ_DPP_PHA_Params_t;

/* DPP parameter structure to be initialized and passed to CAEN_DGTZ_SetDPPParameters.
To be used only for DPP-PSD */

typedef struct {
```

```

int blthr    [MAX_DPP_PSD_CHANNEL_SIZE]; //This parameter is deprecated
int bltmo    [MAX_DPP_PSD_CHANNEL_SIZE]; //This parameter is deprecated
int trgho    [MAX_DPP_PSD_CHANNEL_SIZE]; //Trigger Hold Off (samples)
int thr      [MAX_DPP_PSD_CHANNEL_SIZE]; //Trigger Threshold (LSB)
int selft    [MAX_DPP_PSD_CHANNEL_SIZE]; //Channel self-trigger enable
//options:
//      0 = Disabled,
//      1 = Enabled
int csens     [MAX_DPP_PSD_CHANNEL_SIZE]; //Charge Sensitivity
//options for x720:
//      0 = 40 fC/LSB,
//      1 = 160,
//      2 = 640,
//      3 = 2560 fC/LSB;
//options for x751:
//      0 = 20 fC/LSB
//      1 = 40 fC/LSB
//      2 = 80 fC/LSB
//      3 = 160 fC/LSB
//      4 = 320 fC/LSB
//      5 = 640 fC/LSB
int sgate     [MAX_DPP_PSD_CHANNEL_SIZE]; //Short (i.e. prompt) Charge Integration Gate
//width (samples)
int lgate     [MAX_DPP_PSD_CHANNEL_SIZE]; //Long (i.e. total) Charge Integration Gate
//width (samples)
int pgate     [MAX_DPP_PSD_CHANNEL_SIZE]; //Gate Offset (samples)
int tvaw      [MAX_DPP_PSD_CHANNEL_SIZE]; //Trigger Validation Acceptance Window in
//coincidence mode (samples)
int nsbl      [MAX_DPP_PSD_CHANNEL_SIZE]; //Number of samples for Baseline Mean
//options for x720:
//      0 = FIXED
//      1 = 8
//      2 = 32
//      3 = 128
//options for x751:
//      0 = FIXED,
//      1 = 8
//      2 = 16
//      3 = 32
//      4 = 64
//      5 = 128
//      6 = 256
//      7 = 512
//options for x730/x725:
//      0 = FIXED,
//      1 = 16
//      2 = 64
//      3 = 256
//      4 = 1024
CAEN_DGTZ_DPP_TriggerConfig_t trgc //This parameter is deprecated (must be set to 1)
CAEN_DGTZ_DPP_PUR_t purh           //Pile-Up option selection:
//      0 = CAEN_DGTZ_DPP_PSD_PUR_DetectOnly,
//      1 = CAEN_DGTZ_DPP_PSD_PUR_Enabled)
int purgap    //Pile-Up Rejection GAP value (LSB)
//Ignored for x751
} CAEN_DGTZ_DPP_PSD_Params_t;

/* DPP parameter structure to be initialized and passed to CAEN_DGTZ_SetDPPParameters.
To be used only for DPP-CI */
typedef struct {
int blthr    [MAX_DPP_CI_CHANNEL_SIZE]; //this parameter is deprecated
int bltmo    [MAX_DPP_CI_CHANNEL_SIZE]; //this parameter is deprecated
int trgho    [MAX_DPP_CI_CHANNEL_SIZE]; //Trigger Hold Off (samples)
int thr      [MAX_DPP_CI_CHANNEL_SIZE]; //Trigger Threshold (LSB)
int selft    [MAX_DPP_CI_CHANNEL_SIZE]; //Channel self-trigger enable
//options:
//      0 = Disabled
//      1 = Enabled
int csens     [MAX_DPP_CI_CHANNEL_SIZE]; //Charge Sensitivity Charge Sensitive,
//options:
//      0 = 40 fC/LSB
//      1 = 160 fC/LSB
//      2 = 640 fC/LSB
//      3 = 2560 fC/LSB
int gate      [MAX_DPP_CI_CHANNEL_SIZE]; //Charge Integration Gate width (samples)
int pgate     [MAX_DPP_CI_CHANNEL_SIZE]; //Gate Offset (samples)
int tvaw      [MAX_DPP_CI_CHANNEL_SIZE]; //Trigger Validation Acceptance Window in

```

```

int nsbl      [MAX_DPP_CI_CHANNEL_SIZE]; //coincidence mode (samples)
//Number of Samples for Baseline Mean,
//options for x720:
//          0 = FIXED
//          1 = 8
//          2 = 32
//          3 = 128

CAEN_DGTZ_DPP_TriggerConfig_t trgc //This parameter is deprecated (must be set to 1)
} CAEN_DGTZ_DPP_CI_Params_t;

/* DPP parameter structure to be initialized and passed to CAEN_DGTZ_SetDPPParameters.
   To be used only for DPP-QDC */

typedef struct {
    int trgho      [MAX_X740_GROUP_SIZE]; //Trigger Hold Off (steps of 16 ns)
    int GateWidth  [MAX_X740_GROUP_SIZE]; //Charge Integration Gate width
    int PreGate    [MAX_X740_GROUP_SIZE]; //Charge Integration PreGate width
    int FixedBaseline [MAX_X740_GROUP_SIZE]; //Fixed baseline value (samples)
    int DisTrigHist [MAX_X740_GROUP_SIZE]; //Disable Trigger Hysteresis
    int DisSelfTrigger [MAX_X740_GROUP_SIZE]; //Disable Self Trigger
    int BaselineMode [MAX_X740_GROUP_SIZE]; //Set Baseline Mode
    //options:
    //          0 = FIXED
    //          1 = 4 samples
    //          2 = 16 samples
    //          3 = 64 samples
    //          4 = 256 samples

    int TrgMode     [MAX_X740_GROUP_SIZE]; //Set Trigger Mode
    //options:
    //          0 = Normal
    //          1 = Paired
    //          2 = External Trigger

    int ChargeSensitivity [MAX_X740_GROUP_SIZE]; //Set Charge Sensitivity
    //options:
    //          0 = 0.16 pC
    //          1 = 0.32 pC
    //          2 = 0.64 pC
    //          3 = 1.28 pC
    //          4 = 2.56 pC
    //          5=5.12pC, 6=10.24pC, 7=20.48pC)

    int PulsePol    [MAX_X740_GROUP_SIZE]; //Set Pulse Polarity
    //options:
    //          1 = Negative
    //          0 = Positive

    int EnChargePed  [MAX_X740_GROUP_SIZE]; //Enable Charge Pedestal
    int TestPulsesRate [MAX_X740_GROUP_SIZE]; //Set Test Pulses Rate
    //options:
    //          0 = 1   kHz
    //          1 = 10  kHz
    //          2 = 100 kHz
    //          3 = 1   MHz)

    int EnTestPulses [MAX_X740_GROUP_SIZE]; //Enable Test Pulses
    int InputSmoothing [MAX_X740_GROUP_SIZE]; //Set Smoothing
    //options:
    //          0 = No smoothing
    //          otherwise mean over 2^n samples,
    //          with n <= 6

    int EnableExtendedTimeStamp; //Enable extended timestamp
} CAEN_DGTZ_DPP_QDC_Params_t;

//-----
//Constants definition
//-----
#define MAX_X740_GROUP_SIZE (8)
#define MAX_DPP_CI_CHANNEL_SIZE (MAX_V1720DPP_CHANNEL_SIZE) //The max number of
//channels for DPP-CI
#define MAX_DPP_PSD_CHANNEL_SIZE (MAX_V1730DPP_CHANNEL_SIZE) // The max number of
//channels for DPP-PSD
#define MAX_DPP_PHA_CHANNEL_SIZE (MAX_V1730DPP_CHANNEL_SIZE) //The max number of
//channels for DPP-PHA

```

Arguments

Name	I/O	Description
handle	Input	Device handler
channelMask	Input	A bit mask indicating to which channels the DPP parameters are applied
*params	Input	The pointer to a preallocated struct of type: CAEN_DGTZ_DPP_PHA_Params_t, in case of DPP-PHA

		CAEN_DGTZ_DPP_PSD_Params_t, in case of DPP-PSD CAEN_DGTZ_DPP_CI_Params_t, in case of DPP-CI CAEN_DGTZ_DPP_QDC_Params_t, in case of DPP-QDC Note: refer to the User Manual of the relevant DPP for parameters extended description
--	--	---

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetDPPAcquisitionMode

Description

Sets/gets the DPP acquisition mode.



Note: *SetDPPAcquisitionMode()* should be called before the **SetDPPEventAggregation** function.

Synopsis

```

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPPAcquisitionMode (int handle,
                                CAEN_DGTZ_DPP_AcqMode_t mode,
                                CAEN_DGTZ_DPP_SaveParam_t param
                                );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPPAcquisitionMode (int handle,
                                CAEN_DGTZ_DPP_AcqMode_t * mode,
                                CAEN_DGTZ_DPP_SaveParam_t *param
                                );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_DPP_ACQ_MODE_Oscilloscope = 0L,
    CAEN_DGTZ_DPP_ACQ_MODE_List         = 1L,
    CAEN_DGTZ_DPP_ACQ_MODE_Mixed        = 2L,
} CAEN_DGTZ_DPP_AcqMode_t;

typedef enum
{
    CAEN_DGTZ_DPP_SAVE_PARAM_EnergyOnly   = 0L,
    CAEN_DGTZ_DPP_SAVE_PARAM_TimeOnly     = 1L,
    CAEN_DGTZ_DPP_SAVE_PARAM_EnergyAndTime = 2L,
    CAEN_DGTZ_DPP_SAVE_PARAM_ChargeAndTime = 4L,
    CAEN_DGTZ_DPP_SAVE_PARAM_None         = 3L,
} CAEN_DGTZ_DPP_SaveParam_t;

```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/ Output (Get)	The DPP acquisition mode to set/get. CAEN_DGTZ_DPP_ACQ_MODE_Oscilloscope = 0L: enables the acquisition of the samples of the digitized waveforms. Note: Oscilloscope mode is not supported by DPP-PSD firmware of the 730 digitizer family and by DPP-QDC of x740D family. CAEN_DGTZ_DPP_ACQ_MODE_List = 1L: enables the acquisition of time stamps and energy values for each DPP firmware. CAEN_DGTZ_DPP_ACQ_MODE_Mixed = 2L: enables the acquisition of both waveforms, energies or charges, and time stamps.
param/*param	Input (Set)/ Output (Get)	The acquisition data to retrieve in the acquisition Note: DPP-QDC firmware for X740D Digitizer family only supports CAEN_DGTZ_DPP_SAVE_PARAM_ChargeAndTime

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetDPPTriegerMode

Description

Sets/gets the DPP Trigger mode.



Note: to be used only with DPP-PSD and DPP-CI firmware.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPPTriegerMode (int handle,
                             CAEN_DGTZ_DPP_TriegerMode_t mode
                             );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPPTriegerMode (int handle,
                             CAEN_DGTZ_DPP_TriegerMode_t *mode
                             );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_DPP_TriegerMode_Normal,
    CAEN_DGTZ_DPP_TriegerMode_Coincidence,
} CAEN_DGTZ_DPP_TriegerMode_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/ Output (Get)	For SetDPPTriegerMode, it is the desired trigger mode to be set For GetDPPTriegerMode, it is the current trigger mode.

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetDPP_VirtualProbe

Description

Set/gets the information about virtual probes (both analog and digital probes) of any of the DPP firmware PHA/PSD/QDC/CI.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPP_VirtualProbe (int handle,
                               int trace,
                               int probe);

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPP_VirtualProbe (int handle,
                               int trace,
                               int *probe);

//-----
//Constants Definition
//-----
#define ANALOG_TRACE_1 (0)
#define ANALOG_TRACE_2 (1)
#define DIGITAL_TRACE_1 (2)
#define DIGITAL_TRACE_2 (3)
#define DIGITAL_TRACE_3 (4)
#define DIGITAL_TRACE_4 (5)

#define CAEN_DGTZ_DPP_VIRTUALPROBE_Input (0)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_Delta (1)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_Delta2 (2)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_Trapezoid (3)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_TrapezoidReduced (4)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_Baseline (5)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_Threshold (6)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_CFD (7)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_SmoothedInput (8)
#define CAEN_DGTZ_DPP_VIRTUALPROBE_None (9)
#define CAEN_DGTZ_DPP_DIGITALPROBE_TRGWin (10)
#define CAEN_DGTZ_DPP_DIGITALPROBE_Armed (11)
#define CAEN_DGTZ_DPP_DIGITALPROBE_PkRun (12)
#define CAEN_DGTZ_DPP_DIGITALPROBE_Peaking (13)
#define CAEN_DGTZ_DPP_DIGITALPROBE_CoincWin (14)
#define CAEN_DGTZ_DPP_DIGITALPROBE_BLHoldoff (15)
#define CAEN_DGTZ_DPP_DIGITALPROBE_TRGHoldoff (16)
#define CAEN_DGTZ_DPP_DIGITALPROBE_TRGVal (17)
#define CAEN_DGTZ_DPP_DIGITALPROBE_ACQVeto (18)
#define CAEN_DGTZ_DPP_DIGITALPROBE_BFMVeto (19)
#define CAEN_DGTZ_DPP_DIGITALPROBE_ExtTRG (20)
#define CAEN_DGTZ_DPP_DIGITALPROBE_OverThr (21)
#define CAEN_DGTZ_DPP_DIGITALPROBE_TRGOut (22)
#define CAEN_DGTZ_DPP_DIGITALPROBE_Coincidence (23)
#define CAEN_DGTZ_DPP_DIGITALPROBE_PileUp (24)
#define CAEN_DGTZ_DPP_DIGITALPROBE_Gate (25)
#define CAEN_DGTZ_DPP_DIGITALPROBE_GateShort (26)
#define CAEN_DGTZ_DPP_DIGITALPROBE_Trigger (27)
#define CAEN_DGTZ_DPP_DIGITALPROBE_None (28)
#define CAEN_DGTZ_DPP_DIGITALPROBE_BLFreeze (29)
#define CAEN_DGTZ_DPP_DIGITALPROBE_Busy (30)
#define CAEN_DGTZ_DPP_DIGITALPROBE_PrgVeto (31)
```

Arguments

Name	I/O	Description
handle	Input	Device handler
trace	Input	The Trace to set/get
probe	Input (Set)/ Output (Get)	The Virtual Probe to set/get on the given trace

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Examples

1. How to set the “Delta2” probe on the “analog trace 1” of the DPP-PHA firmware:

```
ret |= CAEN_DGTZ_SetDPP_VirtualProbe(handle, ANALOG_TRACE_1, CAEN_DGTZ_DPP_VIRTUALPROBE_Delta2);
```

2. How to disable the dual trace, i.e. set “None” on the “analog trace 2” of the DPP-PHA firmware:

```
ret |= CAEN_DGTZ_SetDPP_VirtualProbe(handle, ANALOG_TRACE_2, CAEN_DGTZ_DPP_VIRTUALPROBE_None);
```

3. How to set the “Peaking” probe on the “digital trace 1” of the DPP-PHA firmware:

```
ret |= CAEN_DGTZ_SetDPP_VirtualProbe(handle, DIGITAL_TRACE_1, CAEN_DGTZ_DPP_DIGITALPROBE_Peaking).
```

GetDPP_SupportedVirtualProbes

Description

Get the list of virtual probes supported on board's given trace any of the DPP firmware (PHA/PSD/CI).

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPP_SupportedVirtualProbes(int handle,
                                         int trace,
                                         int probes[],
                                         int *numProbes
                                         );

//-----
//Constants Definition
//-----
#define MAX_SUPPORTED_PROBES (16) //The maximum number of probes supported by a trace
```

Arguments

Name	I/O	Description
handle	Input	Device handler
trace	Input	The Trace to be get the probes list of
probes[]	Output	The list of Virtual Probes supported by the trace. Note: it must be an array of length MAX_SUPPORTED_PROBES
numProbes	Output	The number of probes supported by the trace

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetDPP_PHA_VirtualProbe

Description

Set/gets the information about the output signal of the DPP-PHA acquisition mode.



Note: this function is currently **deprecated**. Please use the **Set / GetDPP_VirtualProbe** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPP_PHA_VirtualProbe (int handle,
                                   CAEN_DGTZ_DPP_VirtualProbe_t mode,
                                   CAEN_DGTZ_DPP_PHA_VirtualProbe1_t vp1,
                                   CAEN_DGTZ_DPP_PHA_VirtualProbe2_t vp2,
                                   CAEN_DGTZ_DPP_PHA_DigitalProbe_t dp
                                   );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPP_PHA_VirtualProbe (int handle,
                                   CAEN_DGTZ_DPP_VirtualProbe_t *mode,
                                   CAEN_DGTZ_DPP_PHA_VirtualProbe1_t *vp1,
                                   CAEN_DGTZ_DPP_PHA_VirtualProbe2_t *vp2,
                                   CAEN_DGTZ_DPP_PHA_DigitalProbe_t *dp
                                   );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_DPP_VIRTUALPROBE_SINGLE = 0,
    CAEN_DGTZ_DPP_VIRTUALPROBE_DUAL   = 1,
} CAEN_DGTZ_DPP_VirtualProbe_t;

typedef enum
{
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE1_Input      = 0,
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE1_Delta      = 1,
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE1_Delta2     = 2,
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE1_trapezoid   = 3,
} CAEN_DGTZ_DPP_PHA_VirtualProbe1_t;

typedef enum
{
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE2_Input      = 0,
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE2_S3         = 1,
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE2_DigitalCombo = 2,
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE2_trapBaseline = 3,
    CAEN_DGTZ_DPP_PHA_VIRTUALPROBE2_None       = 4,
} CAEN_DGTZ_DPP_PHA_VirtualProbe2_t;

typedef enum
{
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_trgWin      = 0,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_Armed       = 1,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_PkRun       = 2,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_PURFlag     = 3,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_Peaking     = 4,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_TVAW       = 5,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_BLHoldoff   = 6,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_TRGHoldOff  = 7,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_TRGVal      = 8,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_ACQVeto     = 9,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_BFMVeto     = 10,
    CAEN_DGTZ_DPP_PHA_DIGITAL_PROBE_ExtTRG      = 11,
} CAEN_DGTZ_DPP_PHA_DigitalProbe_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/ Output (Get)	The Virtual Probe mode to set/get.
vp1/*vp1	Input (Set)/ Output (Get)	The Virtual Probe1 mode to set/get
vp2/*vp2	Input (Set)/ Output (Get)	The Virtual Probe2 mode to set/get
dp/*dp	Input (Set)/ Output (Get)	The Digital Probe mode to set/get

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetDPP_PSD_VirtualProbe

Description

Sets/gets the information about the output signal of the DPP-PSD acquisition mode.



Note: this function is currently **deprecated**. Please use the **Set / GetDPP_VirtualProbe** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPP_PSD_VirtualProbe (int handle,
                                    CAEN_DGTZ_DPP_VirtualProbe_t mode,
                                    CAEN_DGTZ_DPP_PSD_VirtualProbe_t vp,
                                    CAEN_DGTZ_DPP_PSD_DigitalProbe1_t dp1,
                                    CAEN_DGTZ_DPP_PSD_DigitalProbe2_t dp2
                                    );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPP_PSD_VirtualProbe (int handle,
                                    CAEN_DGTZ_DPP_VirtualProbe_t *mode,
                                    CAEN_DGTZ_DPP_PSD_VirtualProbe_t *vp,
                                    CAEN_DGTZ_DPP_PSD_DigitalProbe1_t *dp1,
                                    CAEN_DGTZ_DPP_PSD_DigitalProbe2_t *dp2
                                    );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_DPP_VIRTUALPROBE_SINGLE = 0,
    CAEN_DGTZ_DPP_VIRTUALPROBE_DUAL   = 1,
} CAEN_DGTZ_DPP_VirtualProbe_t;

typedef enum
{
    CAEN_DGTZ_DPP_PSD_VIRTUALPROBE_Baseline = 0,
    CAEN_DGTZ_DPP_PSD_VIRTUALPROBE_Threshold = 1,
} CAEN_DGTZ_DPP_PSD_VirtualProbe_t;

typedef enum
{
    /******
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    * The following values are valid for the following DPP-PSD *
    * Firmwares: *
    * x720 Boards: AMC_REL <= 131.5 *
    * x751 Boards: AMC_REL <= 132.5 *
    * For newer firmwares, use the values marked with 'R6' in *
    * the name. *
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    * *****/

    /* x720 Digital Probes Types */
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_Armed           = 0,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_Trigger         = 1,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_ChargeReady     = 2,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_PileUp          = 3,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_B1OutSafeBand   = 4,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_B1Timeout       = 5,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_CoincidenceMet  = 6,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_Tvaw           = 7,
    /* x751 Digital Probes Types */
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_OverThr        = 8,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_GateShort       = 9,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_None           = 10,

    /******
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    * The following values are valid for the following DPP-PSD *
    * Firmwares: *
    * x720 Boards: AMC_REL >= 131.6 *
    * x751 Boards: AMC_REL >= 132.6 *
    * For older firmwares, use the values above. *
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    * *****/

```

```

CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_R6_ExtTrg      = 11, /* x720 only */
CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_R6_OverThr     = 12,
CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_R6_TrigOut     = 13,
CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_R6_CoincWin    = 14,
CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_R6_PileUp      = 15,
CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_R6_Coincidence = 16,
CAEN_DGTZ_DPP_PSD_DIGITALPROBE1_R6_GateLong    = 17, /* x751 only */
} CAEN_DGTZ_DPP_PSD_DigitalProbe1_t;

typedef enum
{
    /******
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    * The following values are valid for the following DPP-PSD *
    * Firmwares: *
    * x720 Boards: AMC_REL <= 131.5 *
    * x751 Boards: AMC_REL <= 132.5 *
    * For newer firmwares, use the values marked with 'R6' in *
    * the name. *
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    *****/

    /* x720 Digital Probes Types */
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_Armed      = 0,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_Trigger    = 1,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_ChargeReady = 2,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_PileUp      = 3,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_BlOutSafeBand = 4,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_BlTimeout  = 5,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_CoincidenceMet = 6,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_Tvaw       = 7,

    /* x751 Digital Probes Types */
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_GateShort  = 8,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_GateLong   = 9,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_None       = 10,

    /******
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    * The following values are valid for the following DPP-PSD *
    * Firmwares: *
    * x720 Boards: AMC_REL >= 131.6 *
    * x751 Boards: AMC_REL >= 132.6 *
    * For older firmwares, use the values above. *
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
    *****/

    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_R6_GateShort = 11,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_R6_OverThr  = 12,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_R6_TrigVal  = 13,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_R6_TrigHO   = 14,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_R6_PileUp    = 15,
    CAEN_DGTZ_DPP_PSD_DIGITALPROBE2_R6_Coincidence = 16,
} CAEN_DGTZ_DPP_PSD_DigitalProbe2_t;

```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/ Output (Get)	The Virtual Probe mode to set/get.
vp/*vp	Input (Set)/ Output (Get)	The Virtual Probe to set/get. Note: ignored for x751; VirtualProbes are always Input and Baseline
dp1/*dp1	Input (Set)/ Output (Get)	The Digital Probe1 to set/get
dp2/*dp2	Input (Set)/ Output (Get)	The Digital Probe2 to set/get

Return Values

0: Success; negative numbers are error codes (see Return Codes).

Set / GetDPP_CI_VirtualProbe

Description

Sets/gets the information about the output signal of the DPP-CI acquisition mode.



Note: this function is supported only by DPP-CI firmware from release 3.4_130.16 on.



Note: this function is currently **deprecated**. Please use the **Set / GetDPP_VirtualProbe** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetDPP_CI_VirtualProbe (int handle,
                                   CAEN_DGTZ_DPP_VirtualProbe_t mode,
                                   CAEN_DGTZ_DPP_CI_VirtualProbe_t vp,
                                   CAEN_DGTZ_DPP_CI_DigitalProbe1_t dp1,
                                   CAEN_DGTZ_DPP_CI_DigitalProbe2_t dp2
                                   );

CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetDPP_CI_VirtualProbe (int handle,
                                   CAEN_DGTZ_DPP_VirtualProbe_t *mode,
                                   CAEN_DGTZ_DPP_CI_VirtualProbe_t *vp,
                                   CAEN_DGTZ_DPP_CI_DigitalProbe1_t *dp1,
                                   CAEN_DGTZ_DPP_CI_DigitalProbe2_t *dp2
                                   );

//-----
//Types Definition
//-----
typedef enum
{
    CAEN_DGTZ_DPP_VIRTUALPROBE_SINGLE = 0,
    CAEN_DGTZ_DPP_VIRTUALPROBE_DUAL   = 1,
} CAEN_DGTZ_DPP_VirtualProbe_t;

typedef enum
{
    CAEN_DGTZ_DPP_CI_VIRTUALPROBE_Baseline = 0,
} CAEN_DGTZ_DPP_CI_VirtualProbe_t;

typedef enum
{
    /*****
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING
    * The following values are valid for the following DPP-CI
    * Firmwares:
    *   x720 Boards: AMC_REL <= 130.20
    * For newer firmwares, use the values marked with 'R22' in
    * the name.
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING
    *****/
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_B1OutSafeBand = 0,
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_B1Timeout    = 1,
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_CoincidenceMet = 2,
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_Tvaw         = 3,

    /*****
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING
    * The following values are valid for the following DPP-CI
    * Firmwares:
    *   x720 Boards: AMC_REL >= 130.22
    * For older firmwares, use the values above.
    * WARNING WARNING WARNING WARNING WARNING WARNING WARNING
    *****/
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_R22_ExtTrg      = 4,
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_R22_OverThr     = 5,
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_R22_TrigOut     = 6,
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_R22_CoincWin    = 7,
    CAEN_DGTZ_DPP_CI_DIGITALPROBE1_R22_Coincidence = 9,
} CAEN_DGTZ_DPP_CI_DigitalProbe1_t;

typedef enum
{
    /*****
```

```

* WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
* The following values are valid for the following DPP-CI *
* Firmwares: *
*   x720 Boards: AMC REL <= 130.20 *
* For newer firmwares, use the values marked with 'R22' in *
* the name. *
* WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
*****/
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_B1OutSafeBand    = 0,
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_B1Timeout       = 1,
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_CoincidenceMet  = 2,
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_Tvaw           = 3,

/*****
* WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
* The following values are valid for the following DPP-CI *
* Firmwares: *
*   x720 Boards: AMC_REL >= 130.22 *
* For older firmwares, use the values above. *
* WARNING WARNING WARNING WARNING WARNING WARNING WARNING *
*****/
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_R22_OverThr     = 5,
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_R22_TrgVal     = 6,
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_R22_TrgHO      = 7,
CAEN_DGTZ_DPP_CI_DIGITALPROBE2_R22_Coincidence = 9,
} CAEN_DGTZ_DPP_CI_DigitalProbe2_t;

```

Arguments

Name	I/O	Description
handle	Input	Device handler
mode/*mode	Input (Set)/ Output (Get)	The Virtual Probe mode to set/get.
vp/*vp	Input (Set)/ Output (Get)	The Virtual Probe to set/get
dp1/*dp1	Input (Set)/ Output (Get)	The Digital Probe1 to set/get
dp2/*dp2	Input (Set)/ Output (Get)	The Digital Probe2 to set/get

Return Values

0: Success; negative numbers are error codes (see Return Codes).

DPP Example Codes

CAEN provides a set of **DPP example codes** intended to let the developer deal with the library C functions to build up a readout cycle when using the provided types of DPP firmware. The CAENDigitizer installation package includes the source C files and Visual Studio projects (compliant to Visual Studio Professional 2010).



Note: the DPP example codes may not include all the library functions and the user must consider tailoring the code and parameters according to his specific purposes.



Note: the DPP-CI example code works only with the DPP-CI firmware from release 3.4_130.16 on.



Note: the DPP-QDC has no examples in CAENDigitizers. Refer to **CAEN DPP-QDC Demo** for a readout example.

In Windows systems, the DPP example codes are in the “Samples” subfolder of “CAEN” main directory (see **Fig. 6.1**).

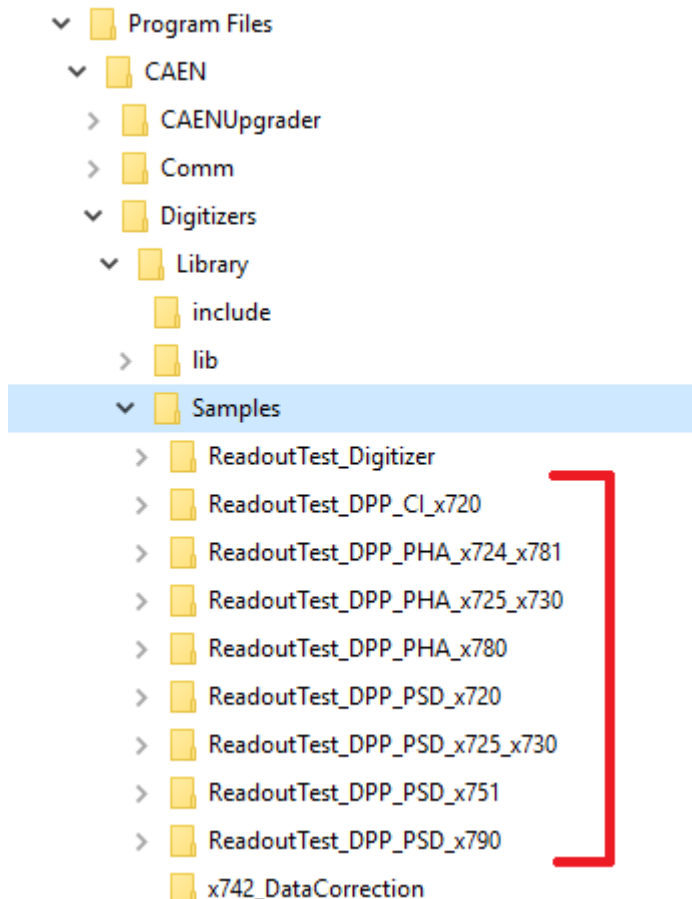


Fig. 6.1: DPP examples location in Windows OS

In Linux systems, the DPP example codes are in the “samples” subfolder of the library main directory. The examples can be compiled under Linux OS typing ‘make’ in the chosen example directory. The executable file is then accessible in the “bin” subfolder.

To give a brief example for Windows OS, we acquire signals using a CAEN **DT5724 Digitizer** running a **DPP-PHA firmware**, connected via a direct USB connection to the computer. We used the CAEN DT5800 Digital Detector Emulator to send exponential signals to ch 0.

Compile the Visual Studio project in the *ReadoutTest_DPP_PHA_x724/build* subfolder and run the compiled code. The example demo opens the following window:

```

Connected to CAEN Digitizer Model DT5724, recognized as board 0
ROC FPGA Release is 04.15 - Build 1511
AMC FPGA Release is 128.36 - Build 1303

s ) Start acquisition
S ) Stop acquisition
r ) Restart acquisition
q ) Quit
t ) Send a software trigger
h ) Save Histograms to file
w ) Save waveforms to file

Type a command: _
    
```

The demo has recognized the board model and the firmware release. We now start the acquisition typing ‘s’, in order to start the acquisition of an exponential signal sent to channel 0 with a Digital Detector Emulator.

We see that channel 0 is triggering, as shown in the following picture. The trigger rate and the pileup rate are also shown.

```
Readout Rate=31.36 MB
Board 0:
  Ch 0:   TrgRate=0.33 KHz      FileUpRate=2.68%
  Ch 1:   No Data
  Ch 2:   No Data
  Ch 3:   No Data
  Ch 4:   No Data
  Ch 5:   No Data
  Ch 6:   No Data
  Ch 7:   No Data
```

We can type 'w' or 'h' to save the waveform or the energy histogram to file. Files are saved in the *Samples/ReadoutTest_DPP_PHA_x724/build* subfolder as "Waveform__<board>_<channel>_<trace>.txt" or "Histo_<board>_<channel>.txt"

7 ZLEplus Specific Functions

This chapter describes the CAENDigitizer library functions relying on the DPP-ZLEplus special firmware supported by the digitizers of the 725, 730 and 751 series. Be aware that functions and types defined as CAEN_DGTZ_730_ZLE[...] are valid for **both x730 and x725** boards.

MallocZLEEvents

Description

It allocates the event buffer matrix which is handled by the **GetZLEEvents** function. The matrix has one event array per channel and must be declared as a MAX_CH-sized array of pointers.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_MallocZLEEvents(
    int handle,
    void **events,
    uint32_t *allocatedSize
);

//-----
//Types Definition
//-----
typedef struct
{
    uint32_t size;
    uint16_t chmask;
    uint32_t tcounter;
    uint64_t timeStamp;
    CAEN_DGTZ_730_ZLE_Channel_t *Channel[MAX_V1730_CHANNEL_SIZE];
}CAEN_DGTZ_730_ZLE_Event_t; //For x730 and x725 boards

typedef struct
{
    uint32_t fifo_full;
    uint32_t size_wrd;
    uint32_t baseline;
    uint32_t *DataPtr;
    CAEN_DGTZ_730_ZLE_Waveforms_t *Waveforms;
} CAEN_DGTZ_730_ZLE_Channel_t; //For x730 and x725 boards

typedef struct
{
    uint32_t TraceNumber;
    uint16_t *Trace;
    uint32_t *TraceIndex;
} CAEN_DGTZ_730_ZLE_Waveforms_t; //For x730 and x725 boards

typedef struct
{
    uint32_t timeTag;
    uint32_t baseline;
    uint32_t *Waveforms;
} CAEN_DGTZ_751_ZLE_Event_t;

//-----
//Constants Definition
//-----
#define MAX_V1730_CHANNEL_SIZE (16)
```

Arguments

Name	I/O	Description
handle	Input	Device handler
**events	Input	The pointer to the event matrix, which shall be of type: CAEN_DGTZ_730_ZLE_Event_t for x730 and x725 boards. CAEN_DGTZ_751_ZLE_Event_t for x751
*allocatedSize	Input	The size in bytes of the event list

Return Values

0: Success; negative numbers are error codes (see Return Codes).

FreeZLEEvents

Description

Deallocates the event buffer matrix.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_FreeZLEEvents(
    int handle,
    void **events
);
```

Arguments

Name	I/O	Description
handle	Input	Device handler
**events	Input	Pointer to the event buffer matrix

Return Values

0: Success; negative numbers are error codes (see Return Codes).

GetZLEEvents

Description

This function returns an array of events (and the number of events present in the array) that are the events stored in the buffer read from the **ReadData** function). Each event has a baseline, a Time Tag and an associate waveform (to be decoded by the **DecodeZLEWaveforms** function)

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_GetZLEEvents (
    int handle,
    char *buffer,
    uint32_t buffsize,
    void **events,
    uint32_t *numEventsArray
);
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*buffer	Input	Pointer to the address of the acquisition buffer
buffsize	Input	The acquisition buffer size (in samples)
**events	Output	Pointer to the event list (allocated via the MallocDPPEvents function)
*numEventsArray	Output	Pointer to an array of <i>int</i> which will contain the number of events found per channel

Return Values

0: Success; negative numbers are error codes (see Return Codes).

MallocZLEWaveforms

Description

Allocates the waveform buffer which is handled by the **DecodeZLEWaveforms** function.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_MallocZLEWaveforms(
    int handle,
    void **waveforms,
    uint32_t *allocatedSize
);

//-----
//Types Definition
//-----
typedef struct
{
    uint32_t TraceNumber;
    uint16_t *Trace;
    uint32_t *TraceIndex;
} CAEN_DGTZ_730_ZLE_Waveforms_t;

typedef struct
{
    uint32_t Ns;
    uint16_t *Trace1;
    uint8_t *Discarded;
} CAEN_DGTZ_751_ZLE_Waveforms_t;
```

Arguments

Name	I/O	Description
handle	Input	Device handler
**waveforms	Output	The pointer to the waveform buffer, which shall be of type: CAEN_DGTZ_730_ZLE_Waveforms_t for x730 and x725 boards . CAEN_DGTZ_751_ZLE_Waveforms_t for x751 boards.
*allocatedSize	Output	The size in bytes of the waveform buffer

Return Values

0: Success; negative numbers are error codes (see Return Codes).

FreeZLEWaveforms

Description

Deallocates the waveform buffer.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_FreeZLEWaveforms(
    int handle,
    void *waveforms
);
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*waveforms	Input	The pointer to the waveform buffer

Return Values

0: Success; negative numbers are error codes (see Return Codes).

DecodeZLEWaveforms

Description

This function decodes the waveforms contained inside an event: takes one event and returns the waveform associated to that event in a waveform buffer.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_DecodeZLEWaveforms(
    int handle,
    void *event,
    void *waveforms
);
```

Arguments

Name	I/O	Description
handle	Input	Device handler
*event	Input	The pointer to the event
*waveforms	Output	The pointer to the (preallocated) waveform list

Return Values

0: Success; negative numbers are error codes (see Return Codes).

SetZLEParameters

Description

It allows to set the ZLE parameters.

Synopsis

```
CAEN_DGTZ_ErrorCode CAENDGTZ_API
CAEN_DGTZ_SetZLEParameters(
    int handle,
    uint32_t channelMask,
    void *params
);

//-----
//Types Definition
//-----
typedef struct {
    int NSampBck      [MAX_ZLE_CHANNEL_SIZE];
    int NSampAhe      [MAX_ZLE_CHANNEL_SIZE];
    int ZleUpThr      [MAX_ZLE_CHANNEL_SIZE];
    int ZleUndThr      [MAX_ZLE_CHANNEL_SIZE];
    int selNumSampBsl [MAX_ZLE_CHANNEL_SIZE];
    int bslThrshld     [MAX_ZLE_CHANNEL_SIZE];
    int bslTimeOut     [MAX_ZLE_CHANNEL_SIZE];
    int preTrgg;
} CAEN_DGTZ_751_ZLE_Params_t;

//not yet implemented for x730 and x725 boards

//-----
//Constants Definition
//-----
#define MAX_V1751DPP_CHANNEL_SIZE (8)
#define MAX_ZLE_CHANNEL_SIZE      (MAX_V1751DPP_CHANNEL_SIZE)
```

Arguments

Name	I/O	Description
handle	Input	Device handler
channelMask	Input	A bit mask indicating to which channels the ZLE parameters are applied.
*params	Input	Pointer to the ZLE paramters array, which shall be of type: CAEN_DGTZ_751_ZLE_Params_t (see the <i>DPP-ZLEplus User Manual</i> for parameters description) for x751 boards.

Return Values

0: Success; negative numbers are error codes (see Return Codes).

8 Applications Notes

Triggering with the x743 Modules

Fig. 8.1 reports the list of CAENDigitizer functions required to manage the trigger modes supported by the 743 digitizer family. The relation with CAEN WaveCatcher software GUI is illustrated for developer's convenience. WaveCatcher is a graphical software application to control the x743 digitizers (see the program web page at: www.caen.it).

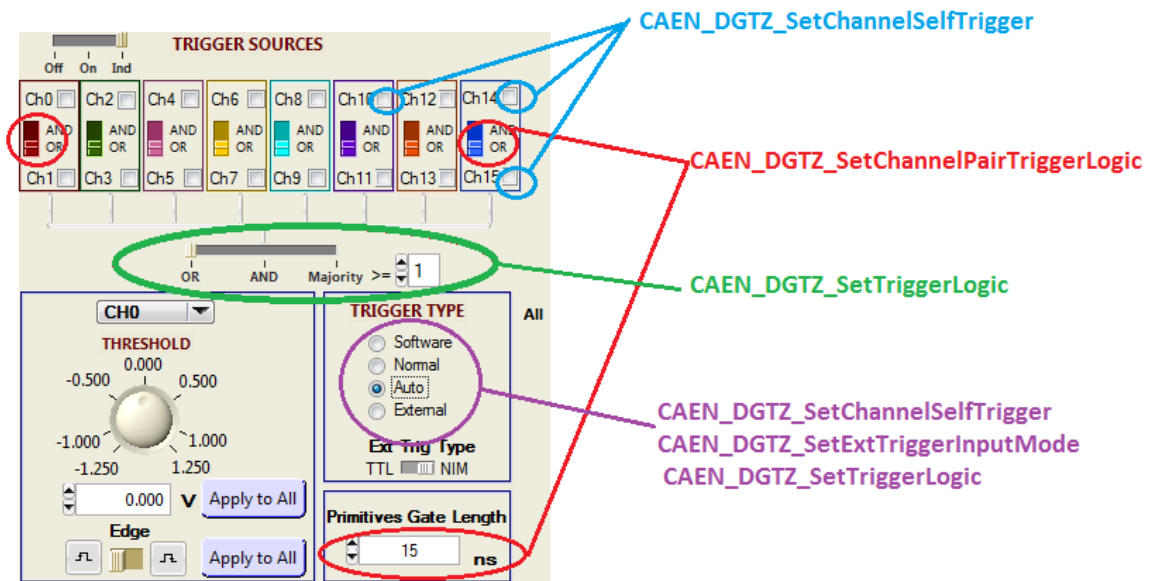


Fig. 8.1: The Trigger tab of CAEN WaveCatcher software

Trigger Modes

A. Software Mode

In Software mode, the x743 is forced to trigger by a software command.

1. Disable all channels sources:

```
CAEN_DGTZ_SetChannelSelfTrigger(int handle, CAEN_DGTZ_TRGMODE_DISABLED, uint32_t channelmask);
```

2. Disable External Trigger Input:

```
CAEN_DGTZ_SetExtTriggerInputMode(int handle, CAEN_DGTZ_TRGMODE_DISABLED);
```

3. Set Global Trigger Option to OR:

```
CAEN_DGTZ_SetTriggerLogic(int handle, CAEN_DGTZ_LOGIC_OR, uint32_t majoritylevel);
// majoritylevel does not matter in this case;
```

4. Use *CAEN_DGTZ_SendSWtrigger (Handle)* to send a trigger.

B. Normal Mode

In Normal mode, the x743 triggers on the channel self-triggers (the trigger threshold and the DC offset parameter need to be properly set with respect to the input pulse).

1. Enable only the desired channels using:

```
CAEN_DGTZ_SetChannelSelfTrigger(int handle, CAEN_DGTZ_TriggerMode_t mode, uint32_t channelmask)
```

2. Disable External Trigger Input:

```
CAEN_DGTZ_SetExtTriggerInputMode(int handle, CAEN_DGTZ_TRGMODE_DISABLED);
```

C. Auto (Normal OR Software) Mode

In Auto mode, the x743 triggers either by software command or on the channel self-triggers.

1. Enable only the desired channels using:

```
CAEN_DGTZ_SetChannelSelfTrigger(int handle, CAEN_DGTZ_TriggerMode_t mode, uint32_t channelmask)
```

2. Disable External Trigger Input:

```
CAEN_DGTZ_SetExtTriggerInputMode(int handle, CAEN_DGTZ_TRGMODE_DISABLED);
```

3. Set Global Trigger Option to OR:

```
CAEN_DGTZ_SetTriggerLogic(int handle, CAEN_DGTZ_LOGIC_OR, uint32_t majoritylevel);  
# majoritylevel does not matter in this case;
```

4. If channels do not trigger by themselves, use *CAEN_DGTZ_SendSWtrigger()* to send a software trigger.

D. External Mode

In External mode, the x743 triggers on the leading edge of an external TLL or NIM signal provided on the TRG-IN front panel connector (NIM by default).

1. Disable all channels sources:

```
CAEN_DGTZ_SetChannelSelfTrigger(int handle, CAEN_DGTZ_TRGMODE_DISABLED, uint32_t channelmask);
```

2. Enable External Trigger Input:

```
CAEN_DGTZ_SetExtTriggerInputMode(int handle, CAEN_DGTZ_TRGMODE_ACQ_ONLY);
```

Enabling Channels as Trigger Sources

The trigger sources are enabled through the function:

```
CAEN_DGTZ_SetChannelSelfTrigger(
    int handle,
    CAEN_DGTZ_TriggerMode_t mode,
    uint32_t channelmask
);
```

See Sec. [Set / GetChannelSelfTrigger](#) .

Example code:

```
channelsMask = 0;
for(i = 0; i<SystemParams.NbOfSAMBlocks; i++) //in the desktop, there are 4 blocks of two
                                                channels.
channelsMask += (SAMBBlockParams[i].EnTriggerCh[0] + ((SAMBBlockParams[i].EnTriggerCh[1])
<<1)<<(2*i));

channelsMask = (~channelsMask) & channelMaskInit;

if(DeviceHandle >= 0) CAEN_DGTZ_SetChannelSelfTrigger(DeviceHandle,
CAEN_DGTZ_TRGMODE_ACQ_ONLY, channelMaskInit);

if(DeviceHandle >= 0) CAEN_DGTZ_SetChannelSelfTrigger(DeviceHandle,
CAEN_DGTZ_TRGMODE_DISABLED, channelsMask);

if(DeviceHandle >= 0) CAEN_DGTZ_SetExtTriggerInputMode(DeviceHandle,
CAEN_DGTZ_TRGMODE_DISABLED);
```

Logic between Channels

Once the trigger sources are defined, the logic between channels is configured by:

```
CAEN_DGTZ_SetChannelPairTriggerLogic(
    int handle,
    uint32_t channelA,
    uint32_t channelB,
    CAEN_DGTZ_TrigerLogic_t logic,
    uint16_t coincidenceWindow
);
```

This function configures the trigger mode for pairs of channels in the same group (0-1, 2-3, 4-5. etc.); it defines the "primitives gate length" (*coincidenceWindows* parameter) that the daughterboards send to the motherboard which then will make an AND or OR or Majority, etc., and it defines also if you want to make an AND or an OR between the two channels of the same group (see Sec. [Set / GetChannelPairTriggerLogic](#)).

Example code:

```
errCode = CAEN_DGTZ_SetChannelPairTriggerLogic(
    handle,
    sa2mIndex*,
    samIndex*2+1,
    (CAEN_DGTZ_TrigerLogic_t)SAMBBlockParams[samIndex].EnCoincidence,
    primitivesGateLength /* in ns, must be a multiple of 5, minium value is 15 ns */
);
```

Global Trigger Logic

In order to define the global trigger logic at the motherboard level (OR, AND or Majority), the reference function is:

```
CAEN_DGTZ_SetTriggerLogic (
    int handle,
    CAEN_DGTZ_TrigerLogic_t logic,
    uint32_t majorityLevel
)
```

See Sec. [Set / GetTriggerLogic](#).

Example of Coincidence between Channel 0 and Channel 5

In order to make coincidence between channel 0 and channel 5:

1. Enable channel 0 and channel 5 using *CAEN_DGTZ_SetChannelSelfTrigger*
2. Define channel pair trigger logic and primitive gatelength which set the coincidence time window:

```
SetChannelPairTriggerLogic(handle, 0 /* channel0*/, 1 /* channel 1*/, 0 ( no coincidence = OR), gatelength);
```

```
SetChannelPairTriggerLogic(handle, 2 /* channel2*/, 3 /* channel 3*/, 0 ( no coincidence = OR), gatelength);  
/* for the other channels it does not matter because they are not enabled */
```

3. Define an AND as an overall trigger option using the *CAEN_DGTZ_SetTriggerLogic* function.

Setting Charge Mode for the x743 Modules

The firmware of the x743 digitizer supports an embedded Charge Mode, where it is possible to calculate the charge inside the portion of the acquired waveform in the event. This section shows how to use the library functions to manage the Charge Mode.

Variable Definitions

```
CAEN_DGTZ_DPP_X743_Event_t *ChargeEvents [MAX_NB_OF_CHANNELS_IN_SYSTEM];
CAEN_DGTZ_DPP_X743_Params_t ChargeParams;
int NbOfEventsPerChannel [MAX_NB_OF_CHANNELS_IN_SYSTEM];
uint32_t ReadoutBufferSize;
uint32_t *ReadoutBuffer;
uint32_t ReadoutBufferMaxSize;
uint32_t NumberOfEvents ;

typedef struct {
CAEN_DGTZ_EnaDis_t disableSuppressBaseline;
unsigned int startCell[MAX_X743_CHANNELS_X_GROUP * MAX_V1743_GROUP_SIZE];
unsigned short chargeLength[MAX_X743_CHANNELS_X_GROUP * MAX_V1743_GROUP_SIZE];
CAEN_DGTZ_EnaDis_t enableChargeThreshold[MAX_X743_CHANNELS_X_GROUP *
MAX_V1743_GROUP_SIZE];
float chargeThreshold[MAX_X743_CHANNELS_X_GROUP * MAX_V1743_GROUP_SIZE]; //
in pC
} CAEN_DGTZ_DPP_X743_Params_t;
```

Enable/Disable Charge Mode

```
/* Enable Charge Mode
CAEN_DGTZ_SetSAMAcquisitionMode(DeviceHandle,CAEN_DGTZ_AcquisitionMode_DPP_CI);

/* Allocate the DPP events
CAEN_DGTZ_MallocDPPEvents(DeviceHandle, (void **)ChargeEvents, &allocatedSize);
```

```
/* Disable Charge Mode
CAEN_DGTZ_SetSAMAcquisitionMode(DeviceHandle,CAEN_DGTZ_AcquisitionMode_STANDARD);

/* Free DPP events
CAEN_DGTZ_FreeDPPEvents(DeviceHandle, (void **)ChargeEvents);
```

Charge Parameters Setting

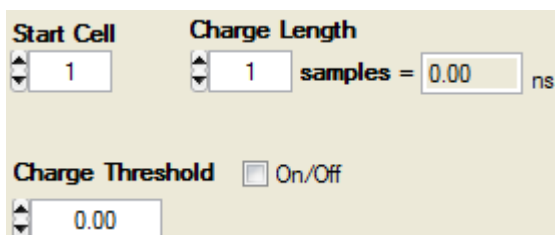


Fig. 8.2: Charge Mode parameters in WaveCatcher software

After building the *ChargeParams* structure, use the following function:

```
CAEN_DGTZ_SetDPPParameters(DeviceHandle, dummy, &ChargeParams);
```

Allocation on the Readout Buffer

Allocation must be done once at the start of the acquisition or software startup. The way is the same as for the Standard Mode (Charge Mode disabled):

```
CAEN_DGTZ_MallocReadoutBuffer(DeviceHandle, (char **)& ReadoutBuffer,
&ReadoutBufferMaxSize);
```


Reading Event Buffer

This is the same as for the Standard Mode.

```
CAEN_DGTZ_ReadData(DeviceHandle, CAEN_DGTZ_SLAVE_TERMINATED_READOUT_MBLT,  
(char*)ReadoutBuffer, &ReadoutBufferSize);
```

Decoding Charge Event

If Charge Mode is enabled, instead of using *CAEN_DGTZ_DecodeEvent* for the Standard Mode, use the function:

```
CAEN_DGTZ_GetDPPEvents(DeviceHandle, (char *)ReadOutBuffer, ReadoutBufferSize,  
(void**)ChargeEvents, NbOfEventsPerChannel);
```

Free of the Readout Buffer

This must be done once at the end of the acquisition or software exit. The way is the same as for the Standard Mode:

```
CAEN_DGTZ_FreeReadoutBuffer((char **) &ReadoutBuffer);
```

9 Examples of Communication Settings

Example No.1

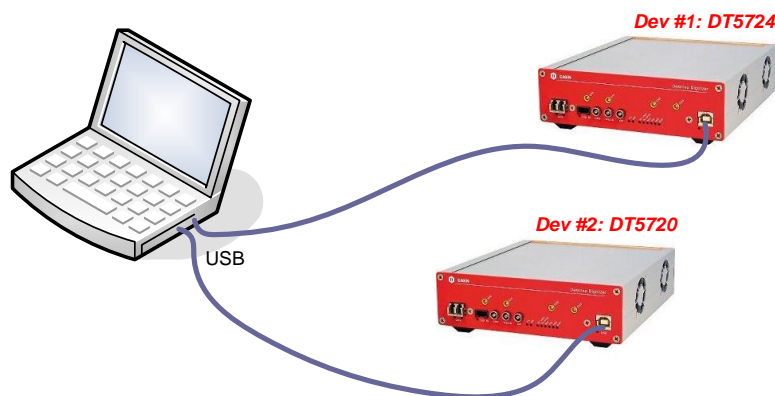


Fig. 9.1: Connection example no.1

The host PC is connected via 2 USB ports to two desktop digitizers:

- Dev#1: DT5724 - 4 Channel 14 bit 100 MS/s Digitizer
- Dev#2: DT5720 - 4 Channel 12 bit 250 MS/s Digitizer

The computer is first connected to DT5724 then to the DT5720.

Open Dev#1: DT5724 connected via USB cable

```
CAEN_DGTZ_OpenDigitizer (  
    CAEN_DGTZ_USB,    LinkType: physical communication channel = USB  
    0,                LinkNum: Link number = 0 first device  
    0,                ConetNode: if USB = 0  
    0,                VMEBaseAddress: must be = 0  
    &handleDT5724_1  Pointer to the handler returned by function  
);
```

Open Dev#2: DT5720 connected via USB cable

```
CAEN_DGTZ_OpenDigitizer (  
    CAEN_DGTZ_USB,    LinkType: physical communication channel = USB  
    1,                LinkNum: Link number = 1 second device  
    0,                ConetNode: if USB = 0  
    0,                VMEBaseAddress: must be = 0  
    &handleDT5720_2  Pointer to the handler returned by function  
);
```

Arguments description

Name	Description
LinkType	= CAEN_DGTZ_USB. Indicates USB as the physical communication channel.
LinkNum	Link number: in case of USB, the link numbers are assigned by the PC when you connect the cable to the device; it is 0 for the first device (DT5724), 1 for the second (DT5720). There is not a fixed correspondence between the USB port and the link number.
ConetNode	In case of USB, ConetNode must be 0.
VMEBaseAddress	Not used = 0 (used only for model accessed via VME).

Example No.2

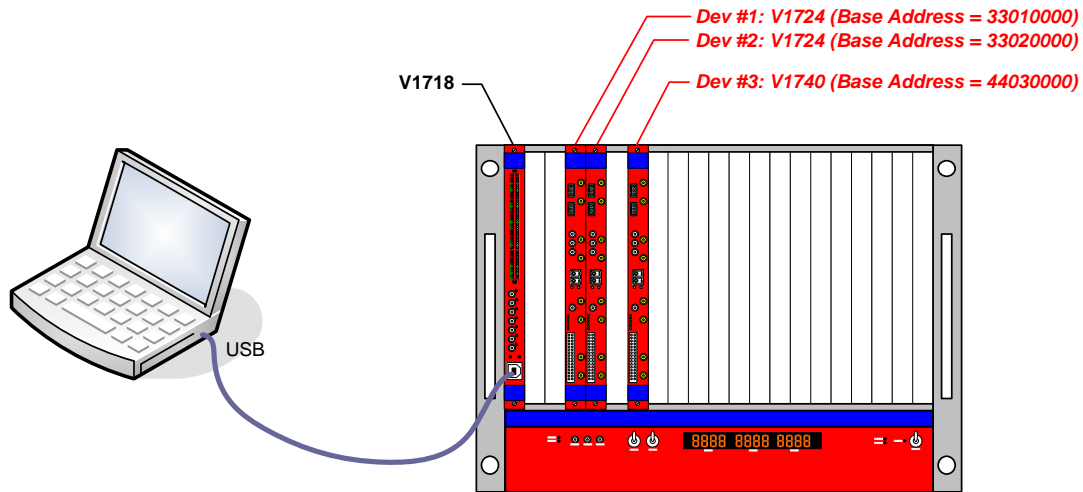


Fig. 9.2: Connection example no.2

The host PC is connected via USB ports to one V1718 VME-USB2.0 Bridge housed in a VME crate. The crate contains also the following boards

- Dev#1: V1724 - 8 Channel 14 bit 100 MS/s Digitizer (Base address = 0x33010000)
- Dev#2: V1724 - 8 Channel 14 bit 100 MS/s Digitizer (Base address = 0x33020000)
- Dev#3: V1740 - 64 Channel 12 bit 62.5 MS/s Digitizer (Base address = 0x44030000)

Open Dev#1: V1724 (VME base address 0x33010000) accessed via VMEbus through the V1718:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_USB,      LinkType: physical communication channel = USB
    0,                  LinkNum: Link number = 0 first device
    0,                  ConetNode: if USB = 0
    0x33010000,          VMEBaseAddress
    &handleV1724_1       Pointer to the handler returned by function
);
```

Open Dev#12: V1724 (VME base address 0x33020000) accessed via VMEbus through the V1718:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_USB,      LinkType: physical communication channel = USB
    0,                  LinkNum: Link number = 0 first device
    0,                  ConetNode: if USB = 0
    0x33020000,          VMEBaseAddress
    &handleV1724_2       Pointer to the handler returned by function
);
```

Open Dev#3: V1740 (VME base address 0x44030000) accessed via VMEbus through the V1718:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_USB,      LinkType: physical communication channel = USB
    0,                  LinkNum: Link number = 0 first device
    0,                  ConetNode: if USB = 0
    0x44030000,          VMEBaseAddress
    &handleV1740_3       Pointer to the handler returned by function
);
```

Arguments description

Name	Description
LinkType	= CAEN_DGTZ_USB. Indicates USB as the physical communication channel.
LinkNum	Link number: in case of USB, the link numbers are assigned by the PC when you connect the cable to the device; it is 0 for the first device, 1 for the second. There is not a fixed correspondence between the USB port and the link number.
ConetNode	In case of USB, ConetNode must be 0.
VMEBaseAddress	VME Base Address of the board (rotary switches setting) expressed as a 32-bit number. This argument is used only for the VME models accessed through the VME bus and MUST BE 0 in all other cases.

Example No.3

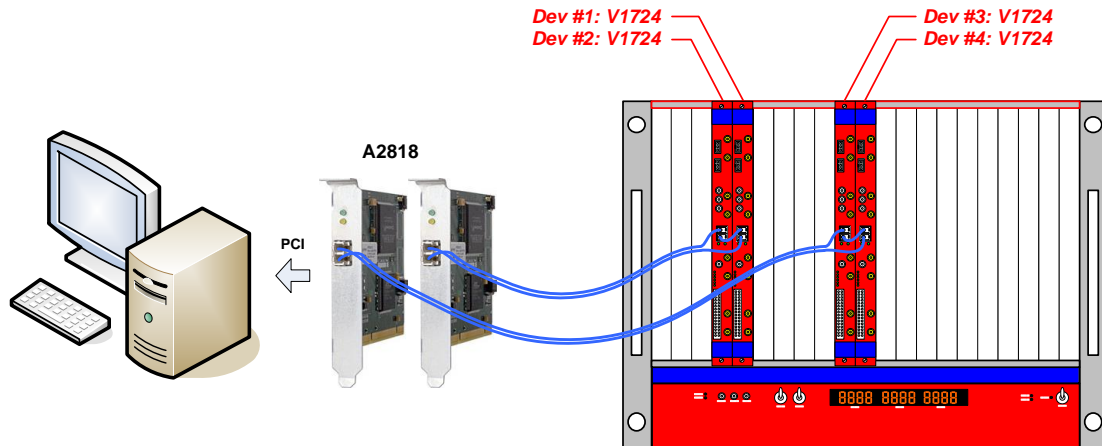


Fig. 9.3: Connection example no.3

The host PC houses two CAEN A2818 PCI CONET Controllers; the VME crate houses the following boards:

- Two V1724 Digitizer connected in a Daisy chain between them end to the A2818 #0: Dev#1 (first in Daisy chain) and Dev#2 (second in Daisy chain)
- Two V1724 Digitizer connected in a Daisy chain between them end to the A2818 #1: Dev#3 (first in Daisy chain) and Dev#4 (second in Daisy chain)



Note: the A2818 number refers to the PCI slot and depends on the motherboard of the PC used. **It is not known a priori which PCI card is assigned to which number.** In this example, we assume that the A2818 connected to Dev#1 and Dev#2, is inserted into the first PCI slot and get Link Number = 0.

Open Dev#1: V1724 first device in Daisy chain of A2818#0:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,    LinkType: physical communication channel = Optical Link via A2818 (PCI Controller)
    0,                        LinkNum: Link number = 0 first device
    0,                        ConetNode: first device in the chain = 0
    0,                        VMEBaseAddress: must be = 0
    &handleV1724_1            Pointer to the handler returned by function
);
```

Open Dev#2: V1724 second device in Daisy chain of A2818#0:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,    LinkType: physical communication channel = Optical Link via A2818 (PCI Controller)
    0,                        LinkNum: Link number = 0 first device
    1,                        ConetNode: second device in the chain = 1
    0,                        VMEBaseAddress: must be = 0
    &handleV1724_2            Pointer to the handler returned by function
);
```

Open Dev#3: V1724 first device in Daisy chain of A2818#1:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,    LinkType: physical communication channel = Optical Link via A2818 (PCI Controller)
    1,                        LinkNum: Link number = 1 second device
    0,                        ConetNode: first device in the chain = 0
    0,                        VMEBaseAddress: must be = 0
    &handleV1724_3            Pointer to the handler returned by function
);
```

Open Dev#4: V1724 second device in Daisy chain of A2818#1:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,    LinkType: physical communication channel = Optical Link via A2818 (PCI Controller)
    1,                        LinkNum: Link number = 1 second device
    1,                        ConetNode: second device in the chain = 1
    0,                        VMEBaseAddress: must be = 0
    &handleV1724_4            Pointer to the handler returned by function
);
```

Arguments description

Name	Description
LinkType	= CAEN_DGTZ_OpticalLink. Indicates A2818 -> Optical Link, either direct connection or VME through V2718 as the physical communication channel. Note: the function CAEN_DGTZ_PCI_OpticalLink is now deprecated, though it is still possible to use it.
LinkNum	Link number: For the CONET, the link number indicates which link of A2818 or A3818 is used. For A2818 refers to the PCI slot and depends on the motherboard of the PC used. Link index start from 0 (1 st link in the 1 st slot used). It is not known a priori which is the first slot used.
ConetNode	The CONET node identifies which device in the Daisy chain is being addressed. The node is 0 for the first device in the chain, 1 for the second and so on. See Fig. 9.4 .
VMEBaseAddress	Not used = 0 (used only for model accessed via VME).

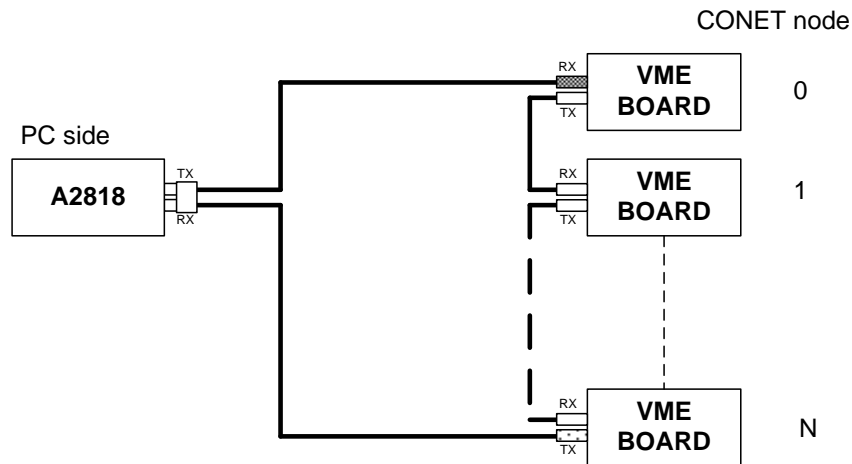


Fig. 9.4: A2818 network scheme

Example No.4

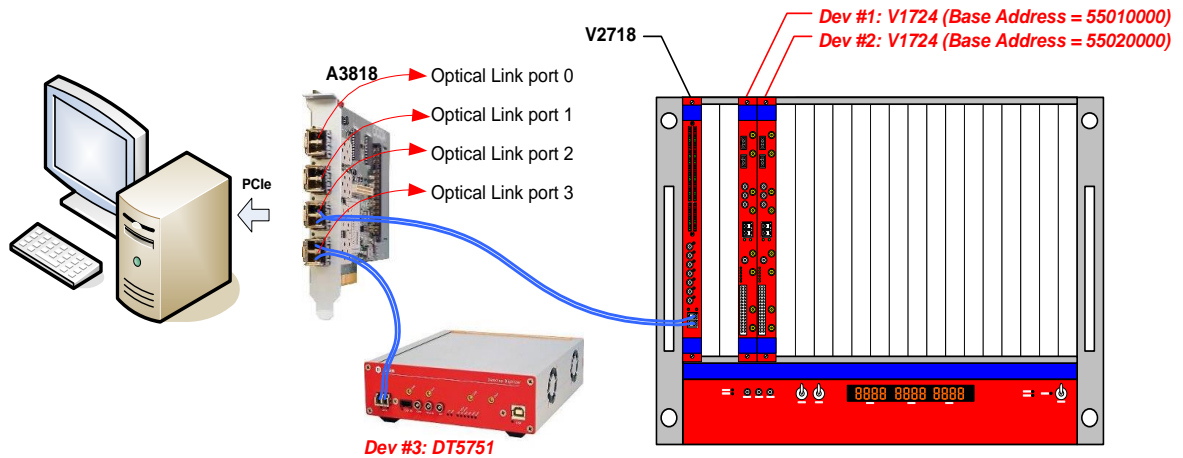


Fig. 9.5: Connection example no.4

The host PC houses one CAEN A3818C PCIe CONET Controller with 4 Optical Link;

- port#3 is connected to Dev#3 (DT5751 - 2/4 Channel 10 bit 2/1 GS/s Digitizer)
- port#2 is connected to a V2718 VME-PCI Optical Link Bridge housed in a VME crate that contains the following boards:
 - Dev#1: V1724 - 8 Channel 14 bit 100 MS/s Digitizer (Base address = 0x55010000)
 - Dev#2: V1724 - 8 Channel 14 bit 100 MS/s Digitizer (Base address = 0x55020000)

Open Dev#1: V1724 (VME base address 0x55010000) accessed via VMEbus through the V2718 connected to A3818 port#2:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,
    2,
    0,
    0x55010000,
    &handleV1724_1
);
```

LinkType: physical communication channel = Optical Link via A3818 (PCIe Controller)
LinkNum: unique device, Link number =A3818 port number: 2
ConetNode: unique device in the chain =0
VMEBaseAddress
Pointer to the handler returned by function

Open Dev#2: V1724 (VME base address 0x55020000) accessed via VMEbus through the V2718 connected to A3818 port#2:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,
    2,
    0,
    0x55020000,
    &handleV1724_2
);
```

LinkType: physical communication channel = Optical Link via A3818 (PCIe Controller)
LinkNum: unique device, Link number =A3818 port number: 2
ConetNode: unique device in the chain =0
VMEBaseAddress
Pointer to the handler returned by function

Open Dev#3: DT5751 first device in Daisy chain of A3818 port#2

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,
    3,
    0,
    0,
    &handleDT5751_3
);
```

LinkType: physical communication channel = Optical Link via A3818 (PCIe Controller)
LinkNum: unique device, Link number =A3818 port number: 3
ConetNode: unique device in the chain =0
VMEBaseAddress: must be = 0
Pointer to the handler returned by function

Arguments description

Name	Description
LinkType	= CAEN_DGTZ_OpticalLink. Indicates A3818 -> Optical Link, either direct connection or VME through V2718 as the physical communication channel. Note: the function CAEN_DGTZ_PCIE_OpticalLink is now deprecated, though it is still possible to use it.
LinkNum	Link number: For the CONET, the link number indicates which link of A2818 or A3818 is used. For A3818 refers to the PCI slot and depends on the motherboard of the PC used. Link index start from 0 (1 st Optical link port in the 1 st slot used). It is not known a priori which slot is used for first. IMPORTANT Note: if also A2818s are installed, these ones have lower index assigned.
ConetNode	The CONET node identifies which device in the Daisy chain is being addressed. The node is 0 for the first device in the chain, 1 for the second and so on.
VMEBaseAddress	used only for model accessed via VME. Must be 0 in other cases

Example No.5

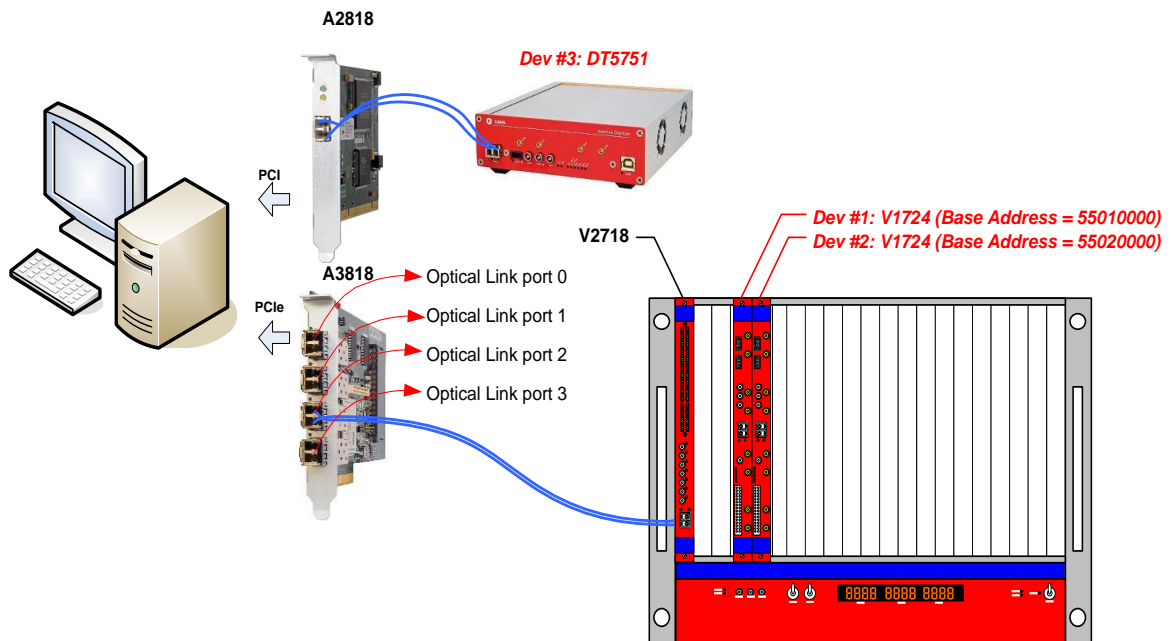


Fig. 9.6: Connection example no.5

The host PC houses

- one A2818 PCI CONET Controller connected to Dev#3 (DT5751 - 2/4 Channel 10 bit 2/1 GS/s Digitizer)
- one CAEN A3818C PCIe CONET Controller with 4 Optical Link; with port#2 connected to a V2718 VME-PCI Optical Link Bridge housed in a VME crate that contains the following boards:
 - Dev#1: V1724 - 8 Channel 14 bit 100 MS/s Digitizer (Base address = 0x55010000)
 - Dev#2: V1724 - 8 Channel 14 bit 100 MS/s Digitizer (Base address = 0x55020000)

Open Dev#1: V1724 (VME base address 0x55010000) accessed via VMEbus through the V2718 connected to A3818 port#2:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,
    3,
    0,
    0x55010000,
    &handleV1724_1
);
```

LinkType: physical communication channel = Optical Link via A3818 (PCIe Controller)
LinkNum: 3 = A3818 port number+1 (to A2818 is assigned the first link =0)
ConetNode: unique device in the chain =0
VMEBaseAddress
Pointer to the handler returned by function

Open Dev#2: V1724 (VME base address 0x55020000) accessed via VMEbus through the V2718 connected to A3818 port#2:

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,
    3,
    0,
    0x55020000,
    &handleV1724_1
);
```

LinkType: physical communication channel = Optical Link via A3818 (PCIe Controller)
LinkNum: 3 = A3818 port number+1 (to A2818 is assigned the first link =0)
ConetNode: unique device in the chain =0
VMEBaseAddress
Pointer to the handler returned by function

Open Dev#3: DT5751 first device in Daisy chain of A2818

```
CAEN_DGTZ_OpenDigitizer (
    CAEN_DGTZ_OpticalLink,
    0,
    0,
    0,
    &handleDT5751_3
);
```

LinkType: physical communication channel = Optical Link via A2818 (PCI Controller)
LinkNum: A2818 has lower index assigned = 0
ConetNode: unique device in the chain =0
VMEBaseAddress: must be = 0
Pointer to the handler returned by function

Arguments description

Name	Description
LinkType	= CAEN_DGTZ_OpticalLink. Indicates A3818 (A2818) -> Optical Link, either direct connection or VME through V2718 as the physical communication channel. Note: functions CAEN_DGTZ_PCI_OpticalLink and CAEN_DGTZ_PCIE_OpticalLink are now deprecated, though it is still possible to use them.
LinkNum	Link number: For the CONET, the link number indicates which link of A2818 or A3818 is used. For A3818/A2818 refers to the PCI slot and depends on the motherboard of the PC used. Link index start from 0 (1 st Optical link port in the 1 st slot used). It is not known a priori which slot is used for first. IMPORTANT Note: if also A2818s are installed, these ones have lower index assigned.
ConetNode	The CONET node identifies which device in the Daisy chain is being addressed. The node is 0 for the first device in the chain, 1 for the second and so on
VMEBaseAddress	Used only for model accessed via VME. Must be 0 in other cases

10 Technical Support

CAEN makes available the technical support of its specialists for requests concerning the software and hardware. Use the support form available at the following link:

<https://www.caen.it/support-services/support-form/>





CAEN SpA is acknowledged as the only company in the world providing a complete range of High/Low Voltage Power Supply systems and Front-End/Data Acquisition modules which meet IEEE Standards for Nuclear and Particle Physics. Extensive Research and Development capabilities have allowed CAEN SpA to play an important, long term role in this field. Our activities have always been at the forefront of technology, thanks to years of intensive collaborations with the most important Research Centres of the world. Our products appeal to a wide range of customers including engineers, scientists and technical professionals who all trust them to help achieve their goals faster and more effectively.

**CAEN S.p.A.**

Via Vetraria, 11
55049 Viareggio
Italy
Tel. +39.0584.388.398
Fax +39.0584.388.959
info@caen.it
www.caen.it

CAEN GmbH

Klingenstraße 108
D-42651 Solingen
Germany
Tel. +49 (0)212 254 4077
Mobile +49 (0)151 16 548 484
Fax +49 (0)212 25 44079
info@caen-de.com
www.caen-de.com
CAEN GmbH

CAEN Technologies, Inc.

1140 Bay Street - Suite 2 C
Staten Island, NY 10305
USA
Tel. +1.718.981.0401
Fax +1.718.556.9185
info@caentechnologies.com
www.caentechnologies.com