CAEN
Tools for Discovery

Electronic Instrumentation

User Manual UM3185

CAENDPP Library

High level library for CAEN Digitizers running DPP firmware

Rev. 1 - July 5th, 2017

Purpose of this User Manual

This User Manual contains the full description of the C version of CAENDPP library, software rel. 1.7.1.

Change Document Record

Date	Revision	Changes
July 3rd, 2014	00	Initial release

July 5th, 2017	01	Updated description of some functions and types in
		`StartBoardParametersGuess`
		
		Added description of new functions
		`AttachBoards`
		`GetInputRange`
		`GetWaveformLength`
		`BoardADCCalibration`
		`GetChannelTemperature`
		`GetDAQInfo`
		`ResetConfiguration`
		
		Added **Appendix.**

Symbols, abbreviated terms and notation

ADC	Analog to Digital Converter
FPGA	Field Programmable Gate Array
MCA	Multi-Channel Analyzer
OS	Operating system
PHA	Pulse Height Analysis
SBC	Single-Board Computer

Reference Document

[RD1]	UM1934 - CAENComm User & Reference Manual
[RD2]	AN2472 - CONET1 to CONET2 migration
[RD3]	UM2784 - CAENDigitizer LabView User & Reference Manual
[RD4]	UM2606 - DT5780 User Manual
[RD5]	UM3904 – GammaStream User Manual
[RD6]	WP2081 - Digital Pulse Processing in Nuclear Physics
[RD7]	UM1935 - CAENDigitizer User & Reference Manual
[RD8]	UM3074 - Digital Detector Emulator User Manual

All CAEN documents can be downloaded from: http://www.caen.it/csite/LibrarySearch.jsp

CAEN
Tools for Discovery

Electronic Instrumentation

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

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 2
Reference Document 2
Index 4
List of Figures 5
List of Tables 5
1 Introduction 6
Drivers 7
Installation 8
Return Codes 9
2 Communication & Configuration 10
InitLibrary 10
EndLibrary 10
AddBoard 11
AttachBoards 12
GetDPPInfo 12
GetParameterInfo 15
StartBoardParametersGuess 16
StopBoardParametersGuess 20
GetBoardParametersGuessStatus 20
GetBoardParametersGuessResult 21
Set / GetBoardConfiguration 22
BoardADCCalibration 22
IsChannelEnabled 23
CheckBoardCommunication 23
SetInputRange 23
GetInputRange 24
GetChannelTemperature 24
GetDAQInfo 24
ResetConfiguration 25
3 Trigger 26
EnableSwTriggers 26
DisableSwTriggers 26
4 Acquisition 27
StartAcquisition 27
ArmAcquisition 27
StopAcquisition 27
IsBoardAcquiring 28
IsChannelAcquiring 28
Set / GetStopCriteria 29
GetAcqStats 30
5 Waves & Histograms 31
GetNumberOfCompleteHistograms 31
GetTotalNumberOfHistograms 31
GetListEvents 32
GetWaveform 32
GetWaveformLength 33
Set / GetHistogram 33
GetCurrentHistogram 34
Set / GetHistoSwitchMode 35
SaveHistogram 35

LoadHistogram.....36
ClearHistogram.....36
ClearCurrentHistogram.....37
ClearAllHistograms.....37
ResetHistogram.....37
ResetAllHistograms.....38
ForceNewHistogram.....38
Set / GetHistogramSize.....39
AddHistogram.....39
Set / GetCurrentHistogramIndex.....40
Set / GetHistogramStatus.....40
Set / GetHistogramRange.....41
6 HV Management.....42
Set / GetHVChannelConfiguration.....42
SetHVChannelVMax.....43
Set / GetHVChannelPowerOn.....43
ReadHVChannelMonitoring.....44
ReadHVChannelExternals.....44
GetHVChannelStatus.....45
GetHVStatusString.....45
7 CAENDPP Demo program.....46
CAENDPP Demo Installation.....46
Acquiring data with CAENDPP Demo.....46
8 Appendix.....55
9 Technical Support.....58

List of Figures

Fig. 1.1: Hardware and Software layers.....7
Fig. 7.1:The V1730 CAEN Digitizer.....46
Fig. 7.2: DPPDemo main menu at first run......47

List of Tables

Tab. 1.1: Return codes.....9

1 Introduction

CAEN has developed a family of Sampling ADCs modules (digitizers) with different form factors (VME, NIM and Desktop). They all can be handled and read out by a host PC via different communication channels.

Exploiting the powerful FPGAs currently avalilable on the market, and to satisfy requests coming from several applications in Nuclear Physics and related fields, CAEN can provide also special firmware implementing specific algorithms for the Digital Pluse Processing (DPP) compliant to specific digitizer families. Running the DPP firmware, the digitizer becomes a complete multi-channel data acquisition system for Nuclear Physics and other applications involving radiation detection.

Note: For information about the special DPP firmware available, please refer to CAEN web site www.caen.it or consult **[RD6]**.

The CAENDPP is a high level library of C functions designed to completely control exclusively the digitizers running the DPP-PHA firmware, that is to say the **724, 725 and 730 digitizer families** and the CAEN **Digital MCAs x770, x780x, x781, Gamma Stream**, and **Hexagon**.

CAENDPP LIBRARY SUPPORTS ONLY DPP-PHA FIRMWARE

The main features of the library can be resumed as follows:

- Multi-board management supporting all the provided communication links (USB, miniUSB, VMEbus, Optical Link, Ethernet)
- Board information reading
- Easy configuration of the boards, fast and customizable through generic writes to the bits of the board internal registers
- Coincidences
- Start Acquisition of specific channel or all the channels of the connected boards
- Arm Acquisition in multi-board synchronization
- Performing and automatic Data Readout management
- Automatic management of the spectra properties (Energy, Dead Time, Real Time, Total Counts, etc.)
- Multi-spectra memory storage (incoming new data are collected in the active spectrum)
- Switching among different spectra by software command or by external signal
- Possibility to set a starting spectrum from which to retrieve a previous acquisition
- Stop criteria configuration to implement time or statistics-driven acquisitions
- List operating mode (list of raw data being Time Stamps and Energies provided by the boards)
- Output list file saving
- Algorithm to automatically detect a preset of parameters for the Detector being used
- Waveforms operating mode (oriented only to the parameter preset)
- Automatic management of the acquisition statistics (trigger rate, bandwidth, etc.)
- HV channel management (if supported by the board)

Supported platforms are Windows and Linux OS (32 and 64 bit).

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 board and compliant with Winodws and Linux OS:

USB 2.0 and miniUSB drivers for the NIM/Desktop boards, the USB 2.0 driver for V1718 CAEN Bridge and the Optical Link driver for A2818 and A3818 optical controllers are downloadable on CAEN website (ww.caen.it) in the Software/Firmware area of the board/bridge/controller web page **(login required).**

Currently, the CAENDPP supports the following communication channels:

- PC → USB → Digitizer (either Desktop or NIM models) and Digital MCAs x780x, x781x
- PC → USB → V1718 → VME → Digitizers (VME models only)
- PC → PCI (A2818) → CONET → Digitizers and Digital MCAs x780x and x781
- PC → PCI (A2818) → CONET → V2718 → VME → Digitizers (VME models only)
- PC → PCI (A3818) → CONET → Digitizers and Digital MCAs x780x and x781
- PC → PCI (A3818) → CONET → V2718 → VME → Digitizers (VME models only)
- PC → miniUSB → Digital MCAs x770, GammaStream and Hexagon
- PC → Ethernet → Digital MCAs x770, GammaStream and Hexagon

CONET (Chainable Optical NETwork) indicates the CAEN proprietary protocol for communication on Optical Link. Refer to **[RD2]** for useful information.

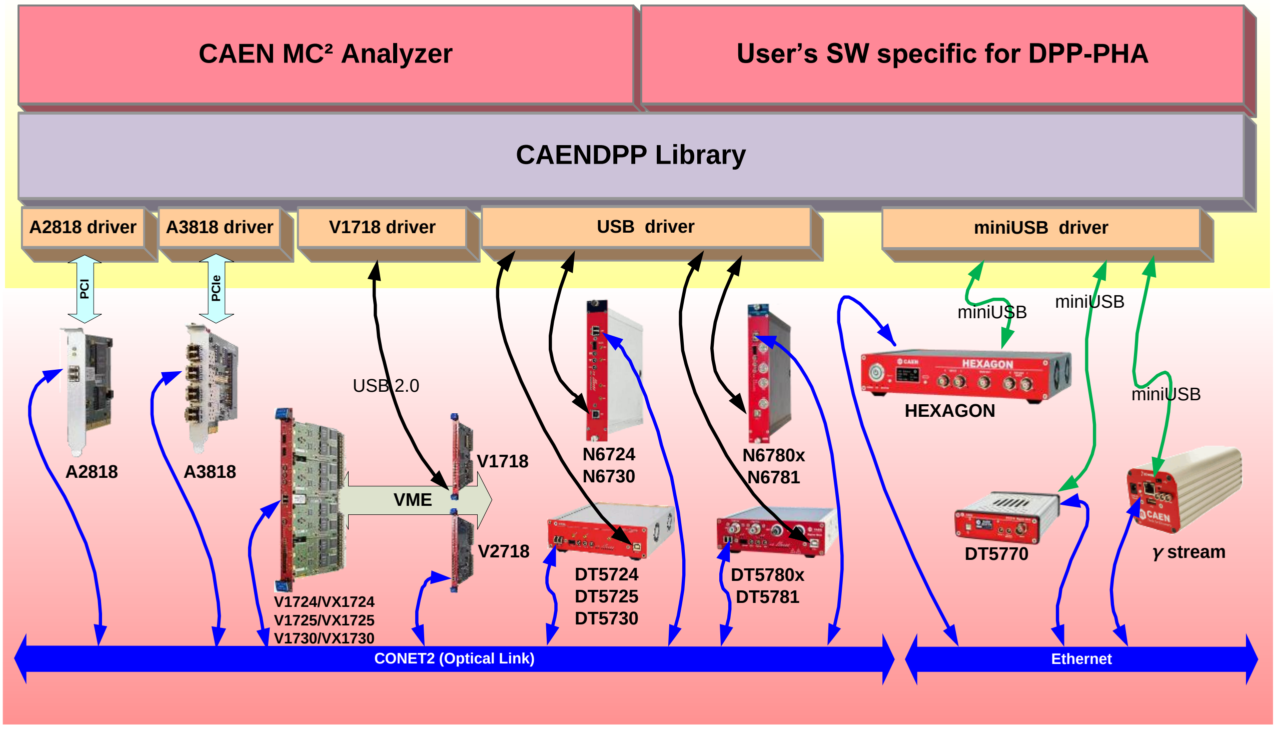


Fig. 1.1: Hardware and Software layers

Installation

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

Before installing CAENDPP library, perform the following steps:

- **Make sure** that your **hardware** (Digitizer, MCA, eventually Bridge and/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 on CAEN website (**login required**).

Then:

- Go to CAEN web site in the “Download” area of *CAENDPPlibrary* page.
- Download the **CAENDPP installation package** related to your OS.
- **Extract files** to your host.

For Windows users

- run the *CAENDPP* setup executable file and follow the installer instructions.

For Linux users:

- follow the instructions in the README file within the library package.

Note: Installation of *CAENDPP* library also includes a demo code as an example for users, provided with source files and the related Visual Studio Professional 2010 project as basis for custom developments (see **CAENDPP Demo program**).

Return Codes

Error code	Value	Meaning
CAENDPP_RetCode_Ok	0	Operation completed successfully
CAENDPP_RetCode_GenericError	-100	Unspecified error
CAENDPP_RetCode_TooManyInstances	-101	Too many instances
CAENDPP_RetCode_ProcessFail	-102	Process fail
CAENDPP_RetCode_ReadFail	-103	Read fail
CAENDPP_RetCode_WriteFail	-104	Write fail
CAENDPP_RetCode_BadMessage	-105	Invalid response
CAENDPP_RetCode_InvalidHandle	-106	Invalid library handle
CAENDPP_RetCode_ConfigError	-107	Configuration error
CAENDPP_RetCode_BoardInitFail	-108	Board Init failed
CAENDPP_RetCode_TimeoutError	-109	Timeout error
CAENDPP_RetCode_InvalidParameter	-110	Invalid parameter
CAENDPP_RetCode_NotInWaveMode	-111	Not in Waveforms Mode
CAENDPP_RetCode_NotInHistoMode	-112	Not in Histogram Mode
CAENDPP_RetCode_NotInListMode	-113	Not in List Mode
CAENDPP_RetCode_NotYetImplemented	-114	Not yet implemented
CAENDPP_RetCode_BoardNotConfigured	-115	Board not configured
CAENDPP_RetCode_InvalidBoardIndex	-116	Invalid board index
CAENDPP_RetCode_InvalidChannelIndex	-117	Invalid channel index
CAENDPP_RetCode_UnsupportedFirmware	-118	Invalid board firmware
CAENDPP_RetCode_NoBoardsAdded	-119	No board added
CAENDPP_RetCode_AcquisitionRunning	-120	Acquisition Status is not compliant with the function called
CAENDPP_RetCode_OutOfMemory	-121	Out of memory
CAENDPP_RetCode_BoardChannelIndex	-122	Invalid board channel index
CAENDPP_RetCode_HistoAlloc	-123	No valid histogram allocated
CAENDPP_RetCode_OpenDumper	-124	Error opening the list dumper
CAENDPP_RetCode_BoardStart	-125	Error starting acquisition for a board
CAENDPP_RetCode_ChannelEnable	-126	The given channel is not enabled
CAENDPP_RetCode_InvalidCommand	-127	Invalid command
CAENDPP_RetCode_NumBins	-128	Invalid number of bins
CAENDPP_RetCode_HistoIndex	-129	Invalid Hitogram Index
CAENDPP_RetCode_UnsupportedFeature	-130	The feature is not supported by the gve board/channel
CAENDPP_RetCode_BadHistoState	-131	The given histogram is an invalid state (e.g. 'done' while it shouldn't)
CAENDPP_RetCode_NoMoreHistograms	-132	Cannot switch to ext histo, no more histograms available
CAENDPP_RetCode_NotHVBoard	-133	The selected board doesn't support HV Channels
CAENDPP_RetCode_InvalidHVChannel	-134	Invalid HV channel index
CAENDPP_RetCode_SocketSend	-135	Error Sending Message through Socket
CAENDPP_RetCode_SocketReceive	-136	Error Receiving Message from Socket
CAENDPP_RetCode_BoardThread	-137	Cannot get Board's acquisition thread
CAENDPP_RetCode_DecodeWaveform	-138	Cannot decode waveform from buffer
CAENDPP_RetCode_OpenDigitizer	-139	Error Opening the digitizer
CAENDPP_RetCode_BoardModel	-140	Requested a feature incompatible with board's Manufacture
CAENDPP_RetCode_AutosetStatus	-141	Autoset Status is not compliant with the requested feature
CAENDPP_RetCode_Autoset	-142	Autoset error looking for signal parameters
CAENDPP_RetCode_Calibration	-143	Calibration Error
CAENDPP_RetCode_EventRead	-144	Event read error

Tab. 1.1: Return codes

2 Communication & Configuration

This set of functions manage CAENDPP library instances, board connection, information retrivial from a board, board configuration and channel enabling check.

InitLibrary

Description

Allows the user to open and initialize a library instance.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_InitLibrary(
                    int32_t * handle
                    );
```

Arguments

Name	I/O	Description
`handle`	Output	Pointer to the handle of the opened DPP instance

Return Values

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

EndLibrary

Description

Allows the user to close a specific instance of the DPP Library.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_EndLibrary(
                   int32_t handle
                   );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance

Return Values

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

AddBoard

Description

Allows the user to open and initialize the connection to a DPP Board.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_AddBoard(
                 int32_t handle,
                 CAENDPP_ConnectionParams_t connParams,
                 int32_t *boardId
                 );

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

typedef struct {
    CAENDPP_ConnectionType LinkType;
    int32_t LinkNum;
    int32_t ConetNode;
    uint32_t VMEBaseAddress;
    char ETHAddress[IP_ADDR_LEN + 1];
} CAENDPP_ConnectionParams_t;
//----------------

typedef enum {
    CAENDPP_USB                 = 0,
    CAENDPP_PCI_OpticalLink     = 1,
    CAENDPP_ETH                 = 2,
    CAENDPP_Serial              = 3,
} CAENDPP_ConnectionType;

//--------------------
//Constants Definition
//--------------------

#define IP_ADDR_LEN      255
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
connParams	Input	*CAENDPP_ConnectionParams_t* type structure specifying the connection paramters of the board to add.
boardId	Output	Pointer to the numeric value that identifies the opened board (used to retrieve boad information)

Return Values

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

AttachBoards

Description

Allows the user to attach to an already running system, whose readout is running on a remote host.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_AttachBoards(
                     char *IP,
                     uint16_t port,
                     int32_t *handle,
                     int32_t *numBrd,
                     int32_t *brdIds
                     );

//--------------------
//Constants Definition
//--------------------

#define MAX_NUMB     20 // Max Number of boards the user can connect
```

Arguments

Name	I/O	Description
IP	Input	Pointer to the host IP address where the acquisition is running
port	Input	The port on the remote host to use for connection
handle	Output	Pointer to the handle to be used with the boards running on the remote instance
numBrd	Output	Pointer to the number of boards managed by the server at the connection time
brdIds	Output	Pointer to the list of board IDs to be used for boards control, filled up to 'numBrd' valid elements. *IMPORTANT: 'brdIds' must be an array of AT LEAST 'MAX_NUMB' elements*

Return Values

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

GetDPPInfo

Description

Allows the user to retrieve general information of a specific DPP board.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetDPPInfo(
                   int32_t handle,
                   int32_t boardId,
                   CAENDPP_Info_t *info
                   );

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

//DPP board info parameters
typedef struct {
    char                     ModelName[MAX_BRDNAME_LEN];
    int32_t                  Model;
    uint32_t                 Channels;
    char                     ROC_FirmwareRel[MAX_FWVER_LENGTH];
    char                     AMC_FirmwareRel[MAX_FWVER_LENGTH];
    char                     License[MAX_LICENSE_LENGTH];
    uint32_t                 SerialNumber;
    uint8_t                  Status;
    int32_t                  FamilyCode;
    uint32_t                 HVChannels;
    uint32_t                 FormFactor;
    uint32_t                 PCB_Revision;
    uint32_t                 ADC_NBits;
    uint32_t                 Energy_MaxNBits;
    uint32_t                 USBOption;
    uint32_t                 ETHOption;
    uint32_t                 WIFIOption;
    uint32_t                 BTOption;
    uint32_t             POEOption;
    uint32_t             GPSOption;
    uint32_t             InputRangeNum;
    CAENDPP_InputRange_t InputRanges[MAX_INRANGES];
    double               Tsample;
    uint32_t             SupportedVirtualProbes1[MAX_PROBES_NUM];
    uint32_t             NumVirtualProbes1;
    uint32_t             SupportedVirtualProbes2[MAX_PROBES_NUM];
    uint32_t             NumVirtualProbes2;
    uint32_t             SupportedDigitalProbes1[MAX_PROBES_NUM];
    uint32_t             NumDigitalProbes1;
    uint32_t             SupportedDigitalProbes2[MAX_PROBES_NUM];
    uint32_t             NumDigitalProbes2;
    int32_t              DPPCodeMaj;
    int32_t              DPPCodeMin;
    CAENDPP_HVChannelInfo_t HVChannelInfo[MAX_HVCHB];
} CAENDPP_Info_t
//-----------------
//HV channels info parameters
typedef struct {
    CAENDPP_ParamInfo_t VSetInfo;
    CAENDPP_ParamInfo_t ISetInfo;
    CAENDPP_ParamInfo_t RampUpInfo;
    CAENDPP_ParamInfo_t RampDownInfo;
    CAENDPP_ParamInfo_t VMaxInfo;
    CAENDPP_ParamInfo_t VMonInfo;
    CAENDPP_ParamInfo_t IMonInfo;
    CAENDPP_ParamInfo_t VExtInfo;
    CAENDPP_ParamInfo_t RTMPInfo;
    CAENDPP_HVFamilyCode_t HVFamilyCode;
} CAENDPP_HVChannelInfo_t;
//-----------------

typedef enum {
    CAENDPP_HVFamilyCode_V6521  = 0  // 5 KV / 300 uA (DT5780)
    CAENDPP_HVFamilyCode_V6533  = 1, // 4 KV / 3mA (DT5790)
    CAENDPP_HVFamilyCode_V6519  = 2, // 500 V / 3 mA (DT5780SD)
    CAENDPP_HVFamilyCode_V6521H = 3, // 5 KV / 20 uA (NONE)
    CAENDPP_HVFamilyCode_V6534  = 4, // 5 KV / 1 mA (NONE)
} CAENDPP_HVFamilyCode_t;
//-----------------

typedef enum
{
    CAENDPP_V1724     = 0,  //The board is V1724
    CAENDPP_DT5724    = 6,  //The board is DT5724
    CAENDPP_N6724     = 12, //The board is N6724
    CAENDPP_DT5780    = 21, //The board is DT5780
    CAENDPP_N6780     = 22, //The board is N6780
    CAENDPP_V1780     = 23, //The board is V1780
    CAENDPP_DT5730    = 30, //The board is DT5730
    CAENDPP_N6730     = 31, //The board is N6730
    CAENDPP_V1730     = 32, //The board is V1730
    CAENDPP_DT5781    = 36, //The board is DT5781
    CAENDPP_N6781     = 37, //The board is N6781
    CAENDPP_V1781     = 38, //The board is V1781
    CAENDPP_DT5770    = -1, //The board is DT5770
    CAENDPP_N6770     = -2, //The board is N6770
    CAENDPP_V1770     = -3, //The board is V1770
    CAENDPP_DT57GS    = -4, //The board is GammaStream
    CAENDPP_DT5000    = 5000, //The board is Hexagon (desktop)
    CAENDPP_DT6000    = 6000, // The board is Hexagon (NIM)
} CAENDPP_BoardModel_t;
//-----------------

typedef enum {
    CAENDPP_FORM_FACTOR_VME64    = 0,
    CAENDPP_FORM_FACTOR_VME64X   = 1,
    CAENDPP_FORM_FACTOR_DESKTOP  = 2,
    CAENDPP_FORM_FACTOR_NIM      = 3,
} CAENDPP_BoardFormFactor_t;
//-----------------

typedef enum {
    CAENDPP_XX724_FAMILY_CODE  = 0,
    CAENDPP_XX780_FAMILY_CODE  = 7,
    CAENDPP_XX730_FAMILY_CODE  = 11,
    CAENDPP_XX781_FAMILY_CODE  = 13,
    CAENDPP_XX725_FAMILY_CODE  = 14,
    CAENDPP_XX000_FAMILY_CODE  = 5000, // Hexagon
    CAENDPP_XX770_FAMILY_CODE  = -1,
    CAENDPP_XX7Gs_FAMILY_CODE  = -2,
} CAENDPP_BoardFamilyCode_t;

//----------------
typedef enum {
    CAENDPP_InputRange_9_5Vpp   = 0, //X780
    CAENDPP_InputRange_3_7Vpp   = 1, //X780
    CAENDPP_InputRange_1_4Vpp   = 2, //X780
    CAENDPP_InputRange_0_6Vpp   = 3, //X780
    CAENDPP_InputRange_3_0Vpp   = 4, //X781
    CAENDPP_InputRange_1_0Vpp   = 5, //X781
    CAENDPP_InputRange_0_3Vpp   = 6, //X781
    CAENDPP_InputRange_10_0Vpp  = 7, //X781 / X770
    CAENDPP_InputRange_5_0Vpp   = 8, //X770
    CAENDPP_InputRange_2_0Vpp   = 9, //X724 / X730
    CAENDPP_InputRange_0_5Vpp   = 10, //X730
    CAENDPP_InputRange_2_5Vpp   = 11, //X770
    CAENDPP_InputRange_1_25Vpp  = 12, //X770
    CAENDPP_InputRange_0_1Vpp   = 13, //Hexagon
    CAENDPP_InputRange_0_21Vpp  = 14, //Hexagon
    CAENDPP_InputRange_0_45Vpp  = 15, //Hexagon
    CAENDPP_InputRange_0_83Vpp  = 16, //Hexagon
    CAENDPP_InputRange_1_6Vpp   = 17, //Hexagon
    CAENDPP_InputRange_3_3Vpp   = 18, //Hexagon
    CAENDPP_InputRange_6_6Vpp   = 19, //Hexagon
    CAENDPP_InputRange_13_3Vpp  = 20, //Hexagon

//NOTE: GammaStream doesn't use Input Ranges
//but gains (X1, X2, X4, X8). It is not pos-
//sible to directly correlate them to an
//input dynamic since there is the charge
//amplifier beneath.

    CAENDPP_InputRange_X0_25    = 93,  // XGS
    CAENDPP_InputRange_X0_5     = 94,  // XGS
    CAENDPP_InputRange_X1       = 95,  // XGS
    CAENDPP_InputRange_X2       = 96,  // XGS
    CAENDPP_InputRange_X4       = 97,  // XGS
    CAENDPP_InputRange_X8       = 98,  // XGS
    CAENDPP_InputRange_X16      = 99,  // XGS
    CAENDPP_InputRange_X32      = 100, // XGS
    CAENDPP_InputRange_X64      = 101, // XGS
    CAENDPP_InputRange_X128     = 102, // XGS
    CAENDPP_InputRange_X256     = 103, // XGS

    CAENDPP_InputRange_UNKN     = -1,
} CAENDPP_InputRange_t;
//--------------------

typedef enum {
    CAENDPP_CODE_PHA_X724  = 0x80, // The code for the DPP-PHA for x724 boards
    CAENDPP_CODE_PHA_X730  = 0x8B, // The code for the DPP-PHA for x730 boards
    CAENDPP_CODE_CI_X720   = 0x82, // The code for the DPP-CI for x720 boards
    CAENDPP_CODE_PSD_X720  = 0x83, // The code for the DPP-PSD for x720 boards
    CAENDPP_CODE_PSD_X751  = 0x84, // The code for the DPP-PSD for x751 boards
    CAENDPP_CODE_ZLE_X751  = 0x85, // The code for the DPP-ZLE for x751 boards
    CAENDPP_CODE_CI_X743   = 0x86, // The code for the DPP-PSD for x743 boards
    CAENDPP_CODE_PSD_X730  = 0x88, // The code for the DPP-PSD for x730 boards
    CAENDPP_CODE_PHA_XHEX  = 0xFF, // The code for the DPP-PHA for Hexagon
} CAENDPP_DPPCode_t;

//--------------------
//Constants Definition
//--------------------

#define MAX_BRDNAME_LEN         12
#define MAX_FWVER_LENGTH        20
#define MAX_LICENSE_LENGTH      17 /* The maximum length of License is uint8_t[8];
                                      to plot it as an hex number on a string we need
                                      2 chars for each uint8_t digit, so 8*2=16. With
                                      the trailing NULL char we need 17 chars. */
#define MAX_INRANGES            4
#define MAX_PROBES_NUM          20 //Maximum Number of supported probes
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`boardId`	Input	Numeric value that identifies the opened board
`info`	Output	Pointer to *CAENDPP_Info_t* type structure containing the information about the opened DPP board

Return Values

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

GetParameterInfo

Description

Allows the user to get the given parameters information for the given channel, basing on its current configuration.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetParameterInfo(
                         int32_t handle,
                         int32_t ch,
                         CAENDPP_ParamID_t param,
                         CAENDPP_ParamInfo_t *info
                         );

//-----------------
//Types Definition
//-----------------
//Details on parameters description can be found in the DPP-PHA User Manual on CAEN website
typedef enum {
    CAENDPP_ParamID_RecordLength  = 0,
    CAENDPP_ParamID_PreTrigger    = 1,
    CAENDPP_ParamID_Decay         = 2,
    CAENDPP_ParamID_TrapRise      = 3,
    CAENDPP_ParamID_TrapFlat      = 4,
    CAENDPP_ParamID_TrapFlatDelay = 5,
    CAENDPP_ParamID_Smoothing     = 6,
    CAENDPP_ParamID_InputRise     = 7,
    CAENDPP_ParamID_Threshold     = 8,
    CAENDPP_ParamID_NSBL          = 9,
    CAENDPP_ParamID_NSPK          = 10,
    CAENDPP_ParamID_PKHO          = 11,
    CAENDPP_ParamID_BLHO          = 12,
    CAENDPP_ParamID_TRGHO         = 13,
    CAENDPP_ParamID_DGain         = 14,
    CAENDPP_ParamID_ENF           = 15,
    CAENDPP_ParamID_Decimation    = 16,
    CAENDPP_ParamID_TWWDT         = 17,
    CAENDPP_ParamID_TRGWin        = 18,
    CAENDPP_ParamID_PulsePol      = 19,
    CAENDPP_ParamID_DCOffset      = 20,
    CAENDPP_ParamID_IOLev         = 21,
    CAENDPP_ParamID_TRGain        = 22,
} CAENDPP_ParamID_t;
//-----------------

typedef struct {
CAENDPP_InfoType_t type;
    double minimum;
    double maximum;
    double resolution;
    double values[MAX_LIST_VALS];
    uint32_t valuesCount;
    CAENDPP_Units_t units;
} CAENDPP_ParamInfo_t;
//-----------------

typedef enum {
    CAENDPP_InfoType_Range = 0,
    CAENDPP_InfoType_List  = 1
} CAENDPP_InfoType_t;
//-----------------

typedef enum {
    CAENDPP_Units_NanoSeconds      =  0,
    CAENDPP_Units_Samples           =  1,
    CAENDPP_Units_Adimensional      =  2,
    CAENDPP_Units_MicroAmpere       =  3,
    CAENDPP_Units_Volt              =  4,
    CAENDPP_Units_VoltAtSecond      =  5,
    CAENDPP_Units_Ohm               =  6,
} CAENDPP_Units_t;

//---------------------
//Constants Definition
//---------------------

#define MAX_LIST_VALS   15
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
Ch	Input	Number identifying the channel index
param	Input	The *CAENDPP_ParamInfo_t* type structure containing the current values of the channel parameters
info	Output	Pointer to the *CAENDPP_ParamInfo_t* type structure containing the requested information

Return Values

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

StartBoardParametersGuess

Description

Allows the user to fill *Params* structure with guessed values using an automatic algorithm based on the waveforms acquired from the digitizer. This function will start the acquisition for the specified board in Waveforms mode and run the algorithm for the channels specified in *channelMask*. The algorithm runs on a dedicated thread, so that the acquisition for the other boards is not affected.

Once the user has called this method, he must use the function **GetBoardParametersGuessStatus** to monitor the algorthm status and, once the status equals *CAENDPP_GuessConfigStatus_Ready*, he must fetch the result using **GetBoardParametersGuessResult**. Alternatively, he can use the function **StopBoardParametersGuess** to stop the algorithm thread and ignore the results. See the corresponding functions description for more details.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_StartBoardParametersGuess(
                                  int32_t handle,
                                  int32_t
                                  boardId,
                                  uint32_t channelMask,
                                  const CAENDPP_DgtzParams_t *Params
                                  )

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

//Parameters for a single digitizer
typedef struct {
//Board Settings
//Generic Write
    int32_t                 GWn;
    uint32_t                GWaddr[MAX_GW];
    uint32_t                GWdata[MAX_GW];
    uint32_t                GWmask[MAX_GW];

//Channel settings
    int32_t                 ChannelMask;
    CAENDPP_PulsePolarity_t  PulsePolarity[MAX_NUMCHB];
    int32_t DCoffset[MAX_NUMCHB];
    CAENDPP_TempCorrParams_t TempCorrParameters[MAX_NUMCHB]; // Only for GammaStream
    CAENDPP_INCoupling_t InputCoupling[MAX_NUMCHB]; // Only for Hexagon

    int32_t EventAggr;
    CAENDPP_PHA_Params_t DPPParams;
    CAENDPP_IOLevel_t IOlev;
```

```
// Waveform Mode Settings, they only affect waveforms acquisition mode
    CAENDPP_WaveformParams_t WFParams;

// List Mode Settings
    CAENDPP_ListParams_t ListParams;

// Run Specifications
    CAENDPP_RunSpecs_t RunSpecifications; // Only for GammaStream

// Parameters for coincidence mode
    CAENDPP_CoincParams_t CoincParams[MAX_NUMCHB_COINCIDENCE];

//Spectrum Control setting
    CAENDPP_SpectrumControl    SpectrumControl[MAX_NUMCHB]; // Only for X770

// Transistor Reset settings
    CAENDPP_TRReset ResetDetector[MAX_NUMCHB]; // Transistor Reset Detector settings
} CAENDPP_DgtzParams_t
//----------------

See Appendix for the definition of CAENDPP_TempCorrParams_t
                                   CAENDPP_INCoupling_t
                                   CAENDPP_RunSpecs_t
                                   CAENDPP_SpectrumControl
//----------------

typedef enum {
    CAENDPP_PulsePolarityPositive = 0,
    CAENDPP_PulsePolarityNegative = 1,
} CAENDPP_PulsePolarity_t;
//----------------

typedef struct {
    int32_t M           [MAX_NUMCHB]; //Signal Decay Time Constant
    int32_t m           [MAX_NUMCHB]; //Trapezoid Flat Top
    int32_t k           [MAX_NUMCHB]; //Trapezoid Rise Time
    int32_t ftd         [MAX_NUMCHB]; //Trapezoid Peaking Delay
    int32_t a           [MAX_NUMCHB]; //Trigger Filter smoothing factor
    int32_t b           [MAX_NUMCHB]; //Input Signal Rise time
    int32_t thr         [MAX_NUMCHB]; //Trigger Threshold
    int32_t nsbl        [MAX_NUMCHB]; //Number of Samples for Baseline Mean
    int32_t nspk        [MAX_NUMCHB]; //Number of Samples for Peak Mean Calculation
    int32_t pkho        [MAX_NUMCHB]; //Peak Hold Off
    int32_t blho        [MAX_NUMCHB]; //Base Line Hold Off
    int32_t trgho       [MAX_NUMCHB]; //Trigger Hold Off
    int32_t dgain       [MAX_NUMCHB]; //Digital Probe Gain
    float   enf         [MAX_NUMCHB]; //Energy Normalization Factor
    int32_t decimation  [MAX_NUMCHB]; //Decimation of Input Signal
    int32_t enskim      [MAX_NUMCHB]; // Enable energy skimming
    int32_t eskimlld    [MAX_NUMCHB]; // LLD    energy skimming
    int32_t eskimuld    [MAX_NUMCHB]; // ULD    energy skimming
    int32_t blrclip     [MAX_NUMCHB]; // Enable baseline restorer clipping
    int32_t dcomp       [MAX_NUMCHB]; // tt_filter compensation
    int32_t trapbsl     [MAX_NUMCHB]; // trapezoid baseline adjuster
    uint32_t pz_dac     [MAX_NUMCHB]; // DAC value used for PoleZero Cancellation
    uint32_t inh_length [MAX_NUMCHB]; // Inhibit length
    CAENDPP_ExtraParameters X770_extraparameters[MAX_NUMCHB]; //parameters for X770 only
} CAENDPP_PHA_Params_t;
//----------------

typedef enum {
    CAENDPP_IOLevel_NIM  = 0,
    CAENDPP_IOLevel_TTL  = 1,
} CAENDPP_IOLevel_t;

//----------------
// Waveform mode config parameters
//----------------
typedef struct {
    int32_t dualTraceMode; // if true dual trace is enabled
    CAENDPP_PHA_VirtualProbe1_t vp1; // First Analog Probe
    CAENDPP_PHA_VirtualProbe2_t vp2; // Second Analog Probe, ignored if dualTraceMode=false
    CAENDPP_PHA_DigitalProbe1_t dp1; // First Digital probe
    CAENDPP_PHA_DigitalProbe2_t dp2; // Second Digital probe

    int32_t recordLength;
    int32_t preTrigger;
```

```
// Only for X770
    CAENDPP_PHA_ProbeTrigger_t probeTrigger;
    int32_t probeSelfTriggerVal;
} CAENDPP_WaveformParams_t;
//-----------------

* \brief Defines the signals that can be carried by the virtual analog probe 1
 *  in the readout data of the DPP-PHA
 */
typedef enum {
    CAENDPP_PHA_VIRTUALPROBE1_Input             = 0,  // BOTH X724 AND X770
    CAENDPP_PHA_VIRTUALPROBE1_Delta             = 1,  // BOTH X724 AND X770
    CAENDPP_PHA_VIRTUALPROBE1_Delta2            = 2,  // BOTH X724 AND X770
    CAENDPP_PHA_VIRTUALPROBE1_Trapezoid         = 3,  // BOTH X724 AND X770
    CAENDPP_PHA_VIRTUALPROBE1_FastTrap          = 4,  // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE1_TrapBaseline      = 5,  // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE1_EnergyOut         = 6,  // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE1_TrapBLCorr        = 7,  // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE1_None              = 8,
    CAENDPP_PHA_VIRTUALPROBE1_Deconvolved       = 9,  // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE1_Dev2FastTrap      = 10, // X770 ONLY
} CAENDPP_PHA_VirtualProbe1_t;


/*!
 * \brief Defines the signals that can be carried by the virtual analog probe 2
 *  in the readout data of the DPP-PHA
 */
typedef enum {
    CAENDPP_PHA_VIRTUALPROBE2_Input             = 0, // BOTH X724 AND X770
    CAENDPP_PHA_VIRTUALPROBE2_S3                = 1, // X724 ONLY
    CAENDPP_PHA_VIRTUALPROBE2_TrapBLCorr        = 2, // BOTH X724 AND X770
    CAENDPP_PHA_VIRTUALPROBE2_TrapBaseline      = 3, // BOTH X724 AND X770
    CAENDPP_PHA_VIRTUALPROBE2_None              = 4, // X724 ONLY
    CAENDPP_PHA_VIRTUALPROBE2_Delta             = 5, // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE2_FastTrap          = 6, // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE2_Delta2            = 7, // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE2_Trapezoid         = 8, // X770 ONLY
    CAENDPP_PHA_VIRTUALPROBE2_EnergyOut         = 9, // X770 ONLY
} CAENDPP_PHA_VirtualProbe2_t;

/*!
 * \brief Defines the digital signals that can be carried by the digital probe 1
 *  in the readout data of the DPP-PHA
 */
typedef enum {
    CAENDPP_PHA_DigitalProbe1_TrgWin            = 0,
    CAENDPP_PHA_DigitalProbe1_Armed             = 1,
    CAENDPP_PHA_DigitalProbe1_PkRun             = 2,
    CAENDPP_PHA_DigitalProbe1_PURFlag           = 3,
    CAENDPP_PHA_DigitalProbe1_Peaking           = 4,
    CAENDPP_PHA_DigitalProbe1_TVAW              = 5,
    CAENDPP_PHA_DigitalProbe1_BLHoldoff         = 6,
    CAENDPP_PHA_DigitalProbe1_TRGHoldoff        = 7,
    CAENDPP_PHA_DigitalProbe1_TRGVal            = 8,
    CAENDPP_PHA_DigitalProbe1_ACQVeto           = 9,
    CAENDPP_PHA_DigitalProbe1_BFMVeto           = 10,
    CAENDPP_PHA_DigitalProbe1_ExtTRG            = 11,
    CAENDPP_PHA_DigitalProbe1_Trigger           = 12,
    CAENDPP_PHA_DigitalProbe1_None              = 13,
    CAENDPP_PHA_DigitalProbe1_EnergyAccepted    = 14,
    CAENDPP_PHA_DigitalProbe1_Saturation        = 15,
    CAENDPP_PHA_DigitalProbe1_Reset             = 16,
    CAENDPP_PHA_DigitalProbe1_BLFreeze          = 17,
    CAENDPP_PHA_DigitalProbe1_Busy              = 18,
    CAENDPP_PHA_DigitalProbe1_PrgVeto           = 19,
} CAENDPP_PHA_DigitalProbe1_t;

/*!
 * \brief Defines the digital signals that can be carried by the digital probe 2
 *  in the readout data of the DPP-PHA
 */
typedef enum {
    CAENDPP_PHA_DigitalProbe2_Trigger          = 0,
    CAENDPP_PHA_DigitalProbe2_None             = 1,
    CAENDPP_PHA_DigitalProbe2_Peaking          = 2,
    CAENDPP_PHA_DigitalProbe2_BLHoldoff        = 3,
    CAENDPP_PHA_DigitalProbe2_PURFlag          = 4,
    CAENDPP_PHA_DigitalProbe2_EnergyAccepted   = 5,
    CAENDPP_PHA_DigitalProbe2_Saturation       = 6,
    CAENDPP_PHA_DigitalProbe2_Reset            = 7,
} CAENDPP_PHA_DigitalProbe2_t;


//----------------
// List mode config parameters
//----------------
typedef struct {
    uint8_t                 enabled;    // 1 = ListMode Enabled, 0 = ListMode Disabled
    CAENDPP_ListSaveMode_t  saveMode;
    char fileName[MAX_LISTFILE_LENGTH]; // the filename used for binary writing
    uint32_t maxBuffNumEvents; // the maximum number of events to keep in the buffer if in
                                memory mode
    uint32_t saveMask; //The mask of the object to be dumped as defined from 'DUMP_MASK_*'
                                macros.
} CAENDPP_ListParams_t;
//----------------

typedef enum {
    CAENDPP_ListSaveMode_Memory     = 0, //Keep the list events in a memory buffer of
                                           maximum size = MAX_LIST_BUFF_NEV
    CAENDPP_ListSaveMode_FileBinary = 1, //Save list events in a binary file.
    CAENDPP_ListSaveMode_FileASCII  = 2, //Save list events in a ASCII file.
} CAENDPP_ListSaveMode_t;


//----------------
// Parameters for coincidence mode
//----------------
typedef struct {
    uint32_t                CoincChMask;
    uint32_t                MajLevel;
    uint32_t                TrgWin;
    CAENDPP_CoincOp_t       CoincOp;
    CAENDPP_CoincLogic_t    CoincLogic;
} CAENDPP_CoincParams_t;
//----------------

typedef enum {
    CAENDPP_CoincOp_OR  = 0,
    CAENDPP_CoincOp_AND = 1,
    CAENDPP_CoincOp_MAJ = 2,
} CAENDPP_CoincOp_t;
//----------------
typedef enum {
    CAENDPP_CoincLogic_None             = 0,
    CAENDPP_CoincLogic_Coincidence      = 2,
    CAENDPP_CoincLogic_Anticoincidence = 3,
} CAENDPP_CoincLogic_t;


//----------------
// Transistor Reset settings
//----------------
typedef struct {
    uint32_t                       Enabled;             // Enable TRReset mode
                                                           (HEXAGON: requires AC coupling)
    CAENDPP_ResetDetectionMode_t   ResetDetectionMode;  // How to detect a reset
    uint32_t                       thrhold;             // Reset negative threshold
                                                           (X770 only)
    uint32_t                       reslenmin;           // Minimum length of the reset
                                                           spike to trigger the reset
                                                           inhibit(X770 only)
    uint32_t                       reslength;           // Inhibit length
} CAENDPP_TRReset;
//----------------
// Only for X770
typedef enum {
    CAENDPP_ResetDetectionMode_Internal = 0,
    CAENDPP_ResetDetectionMode_GPIO     = 1,
    CAENDPP_ResetDetectionMode_Both     = 2
} CAENDPP_ResetDetectionMode_t;
```

```
//---------------------
//Constants Definition
//---------------------
#define MAX_GW              1000 //Max number of generic write register in the Config File
#define MAX_NUMCHB_COINCIDENCE  (MAX_NUMCHB+1)  // Max number of channels for coincidences
                                                   (add external channel)
#define MAX_NUMCHB          16    //Max number of channels per board
#define MAX_LISTFILE_LENGTH 155

//-----------------------
//HV Related definitions
//-----------------------
//Dump masks
#define DUMP_MASK_TTT           (0x1)
#define DUMP_MASK_ENERGY        (0x2)
#define DUMP_MASK_EXTRAS        (0x4)
#define DUMP_MASK_ENERGYSHORT   (0x8)
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
boardId	Input	Numeric value that identifies the opened board
channelMask	Input	The channel mask of channels for which the guess must be extecuted.
Params	Input	Pointer to *DgtzParams_t* type structure containing the board parameters which must be guessed. The function tries to guess only the parameters initialized to '-1', and assume the others to be providedby the user and uses them without changes

Return Values. Negative numbers are error codes (see Return Codes).

StopBoardParametersGuess

Description

This function can be used to stop the autoCalibration algorithm started with function **StartBoardParametersGuess**. The user must have called such function before calling this one, otherwise the function will return an error.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_StopBoardParametersGuess(
                                 int32_t handle,
                                 int32_t boardId
                                 );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
boardId	Input	Numeric value that identifies the opened board

Return Values

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

GetBoardParametersGuessStatus

Description

Allows the user to get the status of the algorithm used to guess the acquisition parameters, which can be started with function **StartBoardParametersGuess**. Once the status equals *CAENDPP_GuessConfigStatus_Ready*, he can use function **GetBoardParametersGuessResult** to get the Guess result.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetBoardParametersGuessStatus(
                                      int32_t handle,
                                      int32_t boardId,
                                      CAENDPP_GuessConfigStatus_t *status
                                      );
```

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

typedef enum {
    CAENDPP_GuessConfigStatus_NotRunning    = 0,
    CAENDPP_GuessConfigStatus_Started       = 1,
    CAENDPP_GuessConfigStatus_PulsePolarity = 2,
    CAENDPP_GuessConfigStatus_DCOffset      = 3,
    CAENDPP_GuessConfigStatus_SignalRise    = 4,
    CAENDPP_GuessConfigStatus_Threshold     = 5,
    CAENDPP_GuessConfigStatus_DecayTime     = 6,
    CAENDPP_GuessConfigStatus_Trapezoid     = 7,
    CAENDPP_GuessConfigStatus_Baseline      = 8,
    CAENDPP_GuessConfigStatus_Ready         = 9
} CAENDPP_GuessConfigStatus_t;
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`boardId`	Input	Numeric value that identifies the opened board
`status`	Output	Pointer to the *GuessConfigStatus_t* type structure which will be filled with the algorithm status

Return Values

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

GetBoardParametersGuessResult

Description

Allows the user to get the resulting parameters found from the autoCalibration algorithm, which can be started using function **StartBoardParametersGuess**. This function must be called only when the algortihm status, which can be fetched using function **GetBoardParametersGuessStatus**, equals *CAENDPP_GuessConfigStatus_Ready*. This function also gives the channelMask of board's channels for which the autoCalibration succeeded. The parameters of the other channels are in an undefined state, so the user must manually adjust them before using them to set again the board configuration. The user must call this function before starting again the acquisition of the board, otherwise the board will not acquire new data.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetBoardParametersGuessResult(
                                      int32_t handle,
                                      int32_t boardId,
                                      CAENDPP_DgtzParams_t *Params,
                                      uint32_t *succMask
                                      );

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

//see StartBoardParametersGuess for CAENDPP_DgtzParams_t definition
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`boardId`	Input	Numeric value that identifies the opened board
`Params`	Output	Pointer to the *DgtzParams_t* value containing the parameters found by the algorithm
`succMask`	Output	Pointer to the channel mask which indicates for which channels the algorithm succeded

Return Values

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

Set / GetBoardConfiguration

Description

Allow to configure a specific board and to get the configuration of that board. The GetBoardConfiguration return an error if the board is not configured.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetBoardConfiguration(
                              int32_t handle,
                              int32_t boardId,
                              CAENDPP_AcqMode_t acqMode,
                              CAENDPP_DgtzParams_t dgtz_params
                              );

int32_t CAENDPP_API
CAENDPP_GetBoardConfiguration(
                              int32_t handle,
                              int32_t boardId,
                              CAENDPP_AcqMode_t *acqMode,
                              CAENDPP_DgtzParams_t *dgtz_params
                              );

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

typedef enum {
    CAENDPP_AcqMode_Waveform    = 0,
    CAENDPP_AcqMode_Histogram   = 1,
} CAENDPP_AcqMode_t;
//----------------
//see StartBoardParametersGuess function for CAENDPP_DgtzParams_t definition
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
boardId	Input	Numeric value that identifies the opened board
acqMode	Input (Set)/Output(Get)	*CAENDPP_AcqMode_t* type structure (or the pointer to, in case of Get) defining the acquisition mode to set/get (Waveform or Histogram)
dgtz_params	Input (Set)/Output(Get)	*CAENDPP_DgtzParams_t* type structure (or the pointer to, in case of Get) setting the configuration parameters to write/read on/from the board

Return Values

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

BoardADCCalibration

Description

Allows the user to calibrate the board’s ADC to get the best performances. The calibration must be done when the ADC temperature gets stabilized (see **GetChannelTemperature** function).

Synopsis

```
int32_t CAENDPP_API
CAENDPP_BoardADCCalibration(
                            int32_t handle,
                            int32_t brd
                            );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
brd	Input	Board index

Return Values

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

IsChannelEnabled

Description

Allows the user to know if a specified channel of the DPP board is enabled.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_IsChannelEnabled(
                         int32_t handle,
                         int32_t channel,
                         int32_t* enabled
                         );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index
enabled	Output	Pointer to the channel giving “1” if the channel is enabled, “0” if not

Return Values

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

CheckBoardCommunication

Description

Allows the user to know if the communication with the given board works properly.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_CheckBoardCommunication(
                                int32_t handle,
                                int32_t boardId
                                );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
boardId	Input	The numeric value identifying the opened board

Return Values

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

SetInputRange

Description

Allows the user to set the Input Voltage Level of the given channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetInputRange(
                      int32_t handle,
                      int32_t channel,
                      CAENDPP_InputRange_t iputLevel
                      );

//-----------------
//Types Definition
//-----------------
//see GetDPPInfo for CAENDPP_InputRange_t definition
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	The number identifying the channel index
iputLevel	Input	The input level to set for the specified channel

Return Values

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

GetInputRange

Description

Allows the user to get the Input Voltage Level of the given channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetInputRange(
                      int32_t handle,
                      int32_t channel,
                      CAENDPP_InputRange_t *iputLevel
                      );

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

//see GetDPPInfoStartBoardParametersGuess for CAENDPP_InputRange_t definition
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	The number identifying the channel index
iputLevel	Input	Pointer to the input level value for the specified channel

Return Values

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

GetChannelTemperature

Description

Allows the user to get the temperature in °C of the given channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetChannelTemperature(
                              int32_t handle,
                              int32_t channel,
                              double *temp
                              );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	The number identifying the channel index
temp	Output	Pointer to the value of the read temperature

Return Values

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

GetDAQInfo

Description

Allows the user to get the DAQ information for the given channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetDAQInfo(
                   int32_t handle,
                   int32_t channel,
                   CAENDPP_DAQInfo_t *infos
                   );
```

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

typedef struct {
    CAENDPP_RunState_t ACQStatus;
    int32_t RunLoop;
    CAENDPP_RunState_t RunState;
    int64_t RunElapsedTimeSec;
    int32_t TotEvtCount;
    int64_t DeadTimeNs;
    CAENDPP_GainStabilizationStatus_t GainStatus;
    int32_t RunID;
} CAENDPP_DAQInfo_t;
//----------------
typedef enum {
    CAENDPP_RunState_Stop  = 0,
    CAENDPP_RunState_Start = 1,
    CAENDPP_RunState_Pause = 2
} CAENDPP_RunState_t;
//----------------
typedef enum {
    CAENDPP_GainStatus_OFF       = 0,
    CAENDPP_GainStatus_Searching = 1,
    CAENDPP_GainStatus_Found     = 2,
    CAENDPP_GainStatus_Lost      = 3,
    CAENDPP_GainStatus_Following = 4
} CAENDPP_GainStabilizationStatus_t;
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	The number identifying the channel index
`infos`	Output	Pointer to the *CAENDPP_DAQInfo_t* structure, cointaining the DAQ information

Return Values

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

ResetConfiguration

Description

Allows the user to reset the board configuration to default.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ResetConfiguration(
                           int32_t handle,
                           int32_t boardId
                           );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`boardId`	Input	The numeric value identifying the opened board

Return Values

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

3 Trigger

EnableSwTriggers

Description

Allows the user to enable the issuing of a software trigger simultaneously to all the channels of all DPP boards.

Synopsis

```
DPP_DEPRECATED(
               int32_t CAENDPP_API
               CAENDPP_EnableSwTriggers(
                                         int32_t handle
                                         )
               );

//-----------------
#ifdef __GNUC__
#define DPP_DEPRECATED(func) func __attribute__ ((deprecated))
#elif defined(_MSC_VER)
#define DPP_DEPRECATED(func) __declspec(deprecated) func
#else
#pragma message("WARNING: DEPRECATED marking not supported on this compiler")
#define DPP_DEPRECATED(func) func
#endif
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance

Return Values

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

DisableSwTriggers

Description

Allows the user to disable the issuing of the software trigger.

Synopsis

```
DPP_DEPRECATED(
               int32_t CAENDPP_API
               CAENDPP_DisableSwTriggers(
                                         int32_t handle
                                         )
               );
//-----------------
see EnableSwTriggers for DPP_DEPRECATED function definition.
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance

Return Values

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

4 Acquisition

The functions in this section allow to manage, control and program the acquisition.

StartAcquisition

Description

Allows the user to start the acquisition on the given channel. The channel must be enabled.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_StartAcquisition(
                         int32_t handle,
                         int32_t channel
                         );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels of all boards)

Return Values

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

ArmAcquisition

Description

Allows the user to arm the acquisition on the enabled channels of the DPP board. The acquisition starts in correspondence of an external signal.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ArmAcquisition(
                       int32_t handle,
                       int32_t channel
                       );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)

Return Values

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

StopAcquisition

Description

Allows the user to stop the acquisition on the given channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_StopAcquisition(
                        int32_t handle,
                        int32_t channel
                        );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)

Return Values

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

IsBoardAcquiring

Description

Allows the user to know the acquisition status of the DPP board.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_IsBoardAcquiring(
                         int32_t handle,
                         int32_t boardId,
                         int32_t* isAcquiring
                         );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`boardId`	Input	Numeric value that identifies the opened board
`isAcquiring`	Output	Pointer to the acquisition flag: “1”, the board is acquiring; “0”, the board is not acquiring

Return Values

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

IsChannelAcquiring

Description

Allows the user to know if the acquisition for a specified channel is ON

Synopsis

```
int32_t CAENDPP_API
CAENDPP_IsChannelAcquiring(
                           int32_t handle,
                           int32_t channel,
                           CAENDPP_AcqStatus_t *acquiring
                           );

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

typedef enum {
    CAENDPP_AcqStatus_Stopped = 0,
    CAENDPP_AcqStatus_Running = 1,
    CAENDPP_AcqStatus_Armed   = 2,
    CAENDPP_AcqStatus_Unknown = 3,
} CAENDPP_AcqStatus_t;
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	The number identifying the channel index
`*acquiring`	Output	Pointer to the acquisition flag: “1”, the board is acquiring; “0”, the board is not acquiring

Return Values

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

Set / GetStopCriteria

Description

Allows the user to set the condition that stops the acquisition and to get the condition that stopped the acquisition.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetStopCriteria(
                        int32_t handle,
                        int32_t channel,
                        CAENDPP_StopCriteria_t stopCrit,
                        uint64_t value
                        );

int32_t CAENDPP_API
CAENDPP_GetStopCriteria(
                        int32_t handle,
                        int32_t channel,
                        CAENDPP_StopCriteria_t *stopCrit,
                        uint64_t *value
                        );

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

typedef enum {
    CAENDPP_StopCriteria_Manual    = 0,
    CAENDPP_StopCriteria_LiveTime  = 1,
    CAENDPP_StopCriteria_RealTime  = 2,
    CAENDPP_StopCriteria_Counts    = 3,
} CAENDPP_StopCriteria_t;
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)
stopCrit	Input (Set)/Output (Get)	*CAENDPP_StopCriteria_t* type structure (or the pointer to, in case of Get) defining the stop acquisition criterium
value	Input (Set)/Output (Get)	Time in nanoseconds/counts (or the pointer to, in case of Get) after which the acquisition is stopped

Return Values

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

GetAcqStats

Description

Allows the user to get the statistics of the acquisition for a specified channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetAcqStats(
                    int32_t handle,
                    int32_t channel,
                    statistics_t* stats
                    );

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

typedef struct {
    double ThroughputRate;   //Data throughput rate in MB/s
    uint32_t SaturationFlag; //Flag set to 1 if there were saturations from the last call
    double SaturationPerc;   //Total percentage of saturations
    double PulseDeadTime;    // Extimated dead time after a single pulse (ns)
    double RealRate;         // Extimated real interaction rate in detector (Hz)
    uint32_t PeakingTime;    // Extimated peaking time (ns)
} statistics_t;
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	Number identifying the channel index (“-1” for all channels)
`stats`	Output	Pointer to *statistics_t* type structure reporting the statistics of the specified channel

Return Values

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

5 Waves & Histograms

This section includes those functions strictly deputed to manage the waveforms or histograms being acquired according to the operating mode set.

GetNumberOfCompleteHistograms

Description
Allows the user to know the number of completed histograms available in memory.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetNumberOfCompleteHistograms(
                                       int32_t handle,
                                       int32_t channel,
                                       int32_t *numHisto
                                       );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)
numHisto	Input	Pointer to the number of completed histograms. A preallocated array of *int32_t* must be supplied in case *channel = -1*.

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

GetTotalNumberOfHistograms

Description
Allows the user to know the total number of histograms (completed or not).

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetTotalNumberOfHistograms(
                                   int32_t handle,
                                   int32_t channel,
                                   int32_t *numHisto
                                   );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)
numHisto	Input	Pointer to the total number of histograms available. A preallocated array of *int32_t* must be supplied in case *channel = -1*.

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

GetListEvents

Description

Allows the user to get list events from the DPP board up to *MAX_LIST_BUFF_NEV* events for the specified channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetListEvents(
                      int32_t handle,
                      int32_t channel,
                      CAENDPP_ListEvent_t *listEvents,
                      uint32_t *nEvts
                      );
//----------------
//Types Definition
//----------------
typedef struct {
    uint64_t   TimeTag;
    uint16_t   Energy;
    uint32_t   Extras;
} CAENDPP_ListEvent_t;
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	Number identifying the channel index
`listEvents`	Output	Pointer to the list event structure *CAENDPP_ListEvent_t*.
`nEvts`	Output	Pointer to the number of events in the structure

Return Values

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

GetWaveform

Description

Allows the user to get the first waveform read in the next readout. If no wave is received in the next readout for the specified channel, the number of samples is 0.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetWaveform(
                    int32_t handle,
                    int32_t channel,
                    int16_t AUTO
                    int16_t *analogTrace1,
                    int16_t *analogTrace2,
                    uint8_t *digitalTrace1,
                    uint8_t *digitalTrace2,
                    uint32_t *ns
                    double *tsample
                    );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	Number identifying the channel index ("-1" for all channels)
`AUTO`	Input	If set to "1", emulates 'AUTO' trigger mode of a common oscilloscope. If no waveform is available, AUTO sends a SW trigger and gives back the acquired wave.
`analogTrace1`	Output	Pointer to the first analog trace samples values
`analogTrace2`	Output	Pointer to the second analog trace samples values
`digitalTrace1`	Output	Pointer to the first digital trace samples values
`digitalTrace2`	Output	Pointer to the second digital trace samples values
`ns`	Output	Pointer to the number of samples for the relevant wave. If no wave is read for that channel, then ns = 0;
`tsample`	Output	Pointer to the value in nanoseconds (ns) of a single sample

Return Values

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

GetWaveformLength

Description

Allows the user to get the length, in samples, of the waveform for a specified channel

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetWaveformLength(
                          int32_t handle,
                          int32_t channel,
                          uint32_t* length
                          );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index ("-1" for all channels)
length	Output	Pointer to the length value of the waveform in samples

Return Values

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

Set / GetHistogram

Description

GetHistogram allows the user to get a histogram from the memory (histogram data are not erased), while SetHistogram allows the user to store a histogram, at the index *histoIndex*, for the specified channel (any previously acquired data for that histogram are erased).

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHistogram(
                     int32_t handle,
                     int32_t channel,
                     int32_t histoIndex,
                     uint64_t realTime_ns,
                     uint64_t deadTime_ns,
                     uint32_t nbins,
                     uint32_t *histo
                     );

int32_t CAENDPP_API
CAENDPP_GetHistogram(
                     int32_t handle,
                     int32_t channel,
                     int32_t histoIndex,
                     void *histo,
                     uint32_t *counts,
                     uint64_t *realTime_ns,
                     uint64_t *deadTime_ns
                     );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index
histoIndex	Input	Index of the histogram to store/get ("- 1" refers to the current histogram)
realTime_ns	Input (Set)/Output (Get)	Real time (or the pointer to, in case of Get) of the histogram expressed in nanoseconds
deadTime_ns	Input (Set)/Output (Get)	Dead time (or the pointer to, in case of Get) of the histogram expressed in nanoseconds
nbins	Input (Set only)	Number of bins in the histogram
histo	Input (Set)/Output (Get)	The pointer to the histogram to store / get
counts	Output (Get only)	Number of counts in the histogram

Return Values

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

GetCurrentHistogram

Description

Allows the user to get the active histogram from the memory and the status of the Acquisition for the specified channel. This function doesn't erase the histogram data.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetCurrentHistogram(
                            int32_t handle,
                            int32_t channel,
                            void *histo,
                            uint32_t *counts,
                            uint64_t *realTime_ns,
                            uint64_t *deadTime_ns,
                            CAENDPP_AcqStatus_t *acqStatus
                            );

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

typedef enum {
    CAENDPP_AcqStatus_Stopped = 0,
    CAENDPP_AcqStatus_Running = 1,
    CAENDPP_AcqStatus_Armed   = 2,
    CAENDPP_AcqStatus_Unknown = 3,
} CAENDPP_AcqStatus_t;
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)
histo	Output	Pointer to the requested histogram
counts	Output	Pointer to the number of counts in the histograms
realTime_ns	Output	Pointer to the real time of the histogram expressed in nanoseconds
deadTime_ns	Output	Pointer to the dead time of the histogram expressed in nanoseconds
acqStatus	Output	Pointer to the *CAENDPP_AcqStatus_t* type structure defining the acquisition status of the specified channel

Return Values

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

Set / GetHistoSwitchMode

Description

Allows the user to switch the histogram when a specified condition is satisfied (Set) and to know which condition has been set (Get).

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHistoSwitchMode(
                           int32_t handle,
                           CAENDPP_MultiHistoCondition_t condition
                           );

int32_t CAENDPP_API
CAENDPP_GetHistoSwitchMode(
                           int32_t handle,
                           CAENDPP_MultiHistoCondition_t *condition
                           );

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

typedef enum {
    CAENDPP_MultiHisto_SoftwareOnly      = 1,
    CAENDPP_MultiHisto_TimeStampReset    = 2,
} CAENDPP_MultiHistoCondition_t;
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
condition	Input (Set)/Output (Get)	*CAENDPP_MultiHistoCondition_t* type structure (or the pointer to, in case of Get) defining the condition for the histogram switching

Return Values

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

SaveHistogram

Description

Allows the user to save the histogram to disk, at index *histoIndex*, for the specified channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SaveHistogram(
                      int32_t handle,
                      int32_t channel,
                      int32_t histoIndex,
                      char *filename
                      );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)
histoIndex	Input	Index of the histogram to store (the most recent histogram has the index “-1”)
filename	Input	Pointer to the name of the file to save the histogram data in

Return Values

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

LoadHistogram

Description
Allows the user to load the histogram at index *histoIndex*, for the specified channel, from disk. This function erases any previously acquired data for that histogram.

Synopsis

```
nt32_t CAENDPP_API
CAENDPP_LoadHistogram(
                      int32_t handle,
                      int32_t channel,
                      int32_t histoIndex,
                      char *filename
                      );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)
histoIndex	Input	Index of the stored histogram to be selected (the most recent histogram has the index “-1”)
filename	Input	Pointer to the file which contains histogram data to load

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

ClearHistogram

Description
Allows the user to clear data of a specified histogram.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ClearHistogram(
                       int32_t handle,
                       int32_t channel,
                       int32_t histoIndex
                       );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	The number identifying the channel index (“-1” for all channels)
histoIndex	Input	Index of the stored histogram to be selected (“- 1 ” refers to the current histogram)

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

ClearCurrentHistogram

Description

Allows the user to clear data of the current histogram.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ClearCurrentHistogram(
                              int32_t handle,
                              int32_t channel
                              );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	The number identifying the channel index (“-1” for all channels)

Return Values

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

ClearAllHistograms

Description

Allows the user to clear data of all histograms of the given channel.

Note: if some histograms of the given channel are in 'completed' state, this function will return the error 'CAENDPP_RetCode_BadHistoState', and the corresponding histogram will not be cleared. Manage the error outside the library if you want to ignore it.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ClearAllHistograms (
                            int32_t handle,
                            int32_t channel
                            );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	The number identifying the channel index (“-1” clears the histograms of all channels)

Return Values

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

ResetHistogram

Description

Allows the user to reset the histogram at the given index. The datas are cleared and the completed flag is set to ‘false’.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ResetHistogram(
                       int32_t handle,
                       int32_t channel,
                       int32_t histoIndex
                       );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP library
`channel`	Input	Number identifying the channel index
`histoIndex`	Input	The index of the stored histogram to be selected (“- 1 ” refers to the current histogram)

Return Values

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

ResetAllHistograms

Description

Allows the user to reset all the histograms of a given channel. This command must be executed only when the acquisition is stopped.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ResetAllHistograms(
                           int32_t handle,
                           int32_t channel
                           );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)

Return Values

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

ForceNewHistogram

Description

Allows the user to force the switch to a new histogram for a given channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ForceNewHistogram(
                          int32_t handle,
                          int32_t channel
                          );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index (“-1” for all channels)

Return Values

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

Set / GetHistogramSize

Description

Allows the user to set/get the size in bins of the histogram at the given index.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHistogramSize(
                         int32_t handle,
                         int32_t channel,
                         int32_t histoIndex,
                         int32_t size
                         );

int32_t CAENDPP_API
CAENDPP_GetHistogramSize(
                         int32_t handle,
                         int32_t channel,
                         int32_t histoIndex,
                         int32_t *size
                         );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	Number identifying the channel index (“-1” for all channels)
`histoIndex`	Input	The index of the stored histogram to be selected. In case of Get function, “-1“ means that no histogram is currently active, other negative values mean that an error occurred
`size`	Input (Set)/Output (Get)	The histogram size in bins to set/get. In case of Set,. It must be a power of 2 until up 16384

Return Values

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

AddHistogram

Description

Allows the user to add a new histogram of the given size.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_AddHistogram(
                     int32_t handle,
                     int32_t channel,
                     int32_t size
                     );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	Number identifying the channel index (“-1” for all channels)
`size`	Input	The histogram size in bins. It must be a power of 2 until up 16384

Return Values

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

Set / GetCurrentHistogramIndex

Description

Allows the user to select the index of the histogram to be set as the currently active histogram. The selected histogram must not be already completed. If you want to set an already completed histogram as active, call **ResetHistogram** or **Set / GetHistogramStatus** to set its status to not completed.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetCurrentHistogramIndex(
                                 int32_t handle,
                                 int32_t channel,
                                 int32_t histoIndex
                                 );

int32_t CAENDPP_API
CAENDPP_GetCurrentHistogramIndex(
                                 int32_t handle,
                                 int32_t channel,
                                 int32_t *histoIndex
                                 );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	Number identifying the channel index (“-1” for all channels)
`histoIndex`	Input (Set)/Output (Get)	The index of the stored histogram to be selected. In case of Get function, “-1“ means that no histogram is currently active, other negative values mean that an error occurred

Return Values

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

Set / GetHistogramStatus

Description

Allows the user to set/get if a histogram is completed or not. The selected index cannot be the one of the current histogram.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHistogramStatus(
                           int32_t handle,
                           int32_t channel,
                           int32_t histoIndex,
                           int32_t completed
                           );

int32_t CAENDPP_API
CAENDPP_GetHistogramStatus(
                           int32_t handle,
                           int32_t channel,
                           int32_t histoIndex,
                           int32_t *completed
                           );
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`channel`	Input	Number identifying the channel index
`histoIndex`	Input	The index of the stored histogram to be selected. In case of Get function , “-1“ means that no histogram is currently active, other negative values means that an error occurred
`completed`	Input (Set)/Output (Get)	The value (or the pointer to, in case of Get function) of the completed flag

Return Values

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

Set / GetHistogramRange

Description

Allows the user to set/get the interesting range (in bins) of a channel's histograms.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHistogramRange(
                          int32_t handle,
                          int32_t channel,
                          int32_t lower,
                          int32_t upper
                          );

int32_t CAENDPP_API
CAENDPP_GetHistogramRange(
                          int32_t handle,
                          int32_t channel,
                          int32_t* lower,
                          int32_t* upper
                          );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
channel	Input	Number identifying the channel index
lower	Input (Set)/Output (Get)	The Lower Level Discriminator (or the pointer to, in case of Get function)
upper	Input (Set)/Output (Get)	The Upper Level Discriminator (or the pointer to, in case of Get function)

Return Values

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

6 HV Management

CAENDPP Library includes a set of functions managing the configuration and control of the HV channels for those Digital MCAs supporting that (e.g. the DT5780 **[RD4],** Gamma Stream **[RD5]** or Hexagon).

Set / GetHVChannelConfiguration

Description
Allows the user to set/get the current configuration of a HV channel.

Note: VMax must be set through the **SetHVChannelVMax** function.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHVChannelConfiguration(
                                  int32_t handle,
                                  int32_t bId,
                                  int32_t ch,
                                  CAENDPP_HVChannelConfig_t config
                                  );

int32_t CAENDPP_API
CAENDPP_GetHVChannelConfiguration(
                                  int32_t handle,
                                  int32_t bId,
                                  int32_t ch,
                                  CAENDPP_HVChannelConfig_t *config
                                  );

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

typedef struct {
    double               VSet;
    double               ISet;
    double               RampUp;
    double               RampDown;
    double               VMax;
    CAENDPP_PWDownMode_t PWDownMode;
} CAENDPP_HVChannelConfig_t;
//----------------

typedef enum {
    CAENDPP_PWDownMode_Ramp = 0,
    CAENDPP_PWDownMode_Kill = 1,
} CAENDPP_PWDownMode_t;
```

Arguments

Name	I/O	Description
`handle`	Input	Handle of the opened DPP instance
`bId`	Input	The numeric value representing the opened board
`ch`	Input	The number identifying the channel
`config`	Input (Set)/Output (Get)	*CAENDPP_HVChannelConfig_t* type structure (or the pointer to, in case of Get) defining the configuration to set/get for the specified channel

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

SetHVChannelVMax

Description

Allows the user to set the VMax of a HV channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHVChannelVMax(
                         int32_t handle,
                         int32_t bId,
                         int32_t ch,
                         double VMax
                         );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
bId	Input	The numeric value representing the opened board
ch	Input	The number identifying the channel index
VMax	Input	The Vmax value to set for the specified HV channel

Return Values

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

Set / GetHVChannelPowerOn

Description

SetHVChannelPowerOn function allows the user to switch on/off a HV channel, while *GetHVChannelPowerOn* function allows the user to get whether a HV channel is switched on/off.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_SetHVChannelPowerOn(
                            int32_t handle,
                            int32_t bId,
                            int32_t ch,
                            int32_t on
                            );

int32_t CAENDPP_API
CAENDPP_GetHVChannelPowerOn(
                            int32_t handle,
                            int32_t bId,
                            int32_t ch,
                            uint32_t *on
                            );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
bId	Input	The numeric value representing the opened board
ch	Input	The number identifying the channel
on	Input (Set)/Output (Get)	Set: "0" = switch off the channel "1" = switch on the channel Get: "0" = the channel is switched off "1" = the channels is switched on

Return Values

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

ReadHVChannelMonitoring

Description

Allows the user to get the monitored bias voltage (VMon) and current (IMon) of the specified HV channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ReadHVChannelMonitoring(
                                int32_t handle,
                                int32_t bId,
                                int32_t ch,
                                double *VMon,
                                double *IMon
                                );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP library
bId	Input	Numeric value representing the opened board
ch	Input	Number identifying the channel
VMon	Output	The value (or the pointer to, in case of Get) in Volts (V) of the monitored bias voltage for the specified channel
IMon	Output	The value (or the pointer to, in case of Get) in microAmpere (µA) of the monitored bias current for the specified channel

Return Values

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

ReadHVChannelExternals

Description

Allows the user to get the monitored external voltage (VExt) and PT100 (TRes) of the specified channel.

Note: This function may take up to 10 ms to be executed.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_ReadHVChannelExternals(
                               int32_t handle,
                               int32_t bId,
                               int32_t ch,
                               double *VExt,
                               double *TRes
                               );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
bId	Input	Numeric value representing the opened board
ch	Input	Number identifying the channel
VExt	Output	The value (or the pointer to, in case of Get) in Volts (V) of the monitored external voltage for the specified channel
TRes	Output	The value (or the pointer to, in case of Get) in Ohm (Ω) of the external PT100 for the specified channel

Return Values

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

GetHVChannelStatus

Description

Allows the user to get the current status of a HV channel.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetHVChannelStatus(
                           int32_t handle,
                           int32_t bId,
                           int32_t ch,
                           uint16_t *status
                           );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP library
bId	Input	The numeric value representing the opened board
ch	Input	The number identifying the channel
status	Input	Pointer to the encoded status of the specified HV channel

Return Values

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

GetHVStatusString

Description

Allows the user to convert the encoded channel status to a Human Readable string.

Synopsis

```
int32_t CAENDPP_API
CAENDPP_GetHVStatusString(
                          int32_t handle,
                          int32_t bId,
                          uint16_t status,
                          char *statusString
                          );
```

Arguments

Name	I/O	Description
handle	Input	Handle of the opened DPP instance
bId	Input	The numeric value representing the opened board
status	Input	The encoded status
statusString	Output	Pointer to the Human Readable string

Return Values

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

7 CAENDPP Demo program

The *DemoDPPLib* folder installed with CAENDPP library includes a hardcoded program as an example for developers who can base their software customization on it. The source codes and the Visual Studio Pro 2010 project file are provided; a Makefile is available to compile the program in Linux environment.

By default, this sample program provides the following main functionalities:

- CAENDPP instance management
- Multi-board management
- Multi-histogram-per-channel management
- Connection setting
- DPP parameter setting and Acquistion management
- Waveforms and Histogram operating mode management
- Plotting of waveforms and histograms
- Saving of histograms to file

Note: Default settings and parameters values in the code must not be considered universally valid. The user is recommened to tune them, or part of them, accordingly to the specific case.

CAENDPP Demo Installation

WINDOWS USERS: once installed the CAENDPP library, the *DemoDPPLib* files can be found in the default system path

C:\Program Files\CAEN\Digitizers\DPPLibrary\DemoDPPLib\build

Run the DemoDPPLib.sln file and compile the VS solution.

LINUX USERS: once the CAENDPP library installation package has been unpacked:

- Change directory to the DemoDPPLib subfolder
- Execute 'make'
- Change directory to \bin\x86_x64
- Execute ./DemoDPPLib.bin (DemoDPPLib.bin is executable under every path)

Acquiring data with CAENDPP Demo

In this section we provide step-by-step instructions on how to work with the CAENDPP demo, from the board connection and configuration, to the visualization of the waveform and histogram plots.
In this example we use a V1730 CAEN Digitizer running the DPP-PHA firmware version 4.11_139.6. The firmware is available in the V1730 page on the CAEN website.

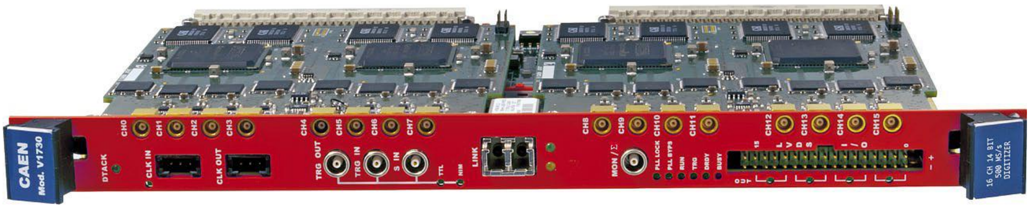
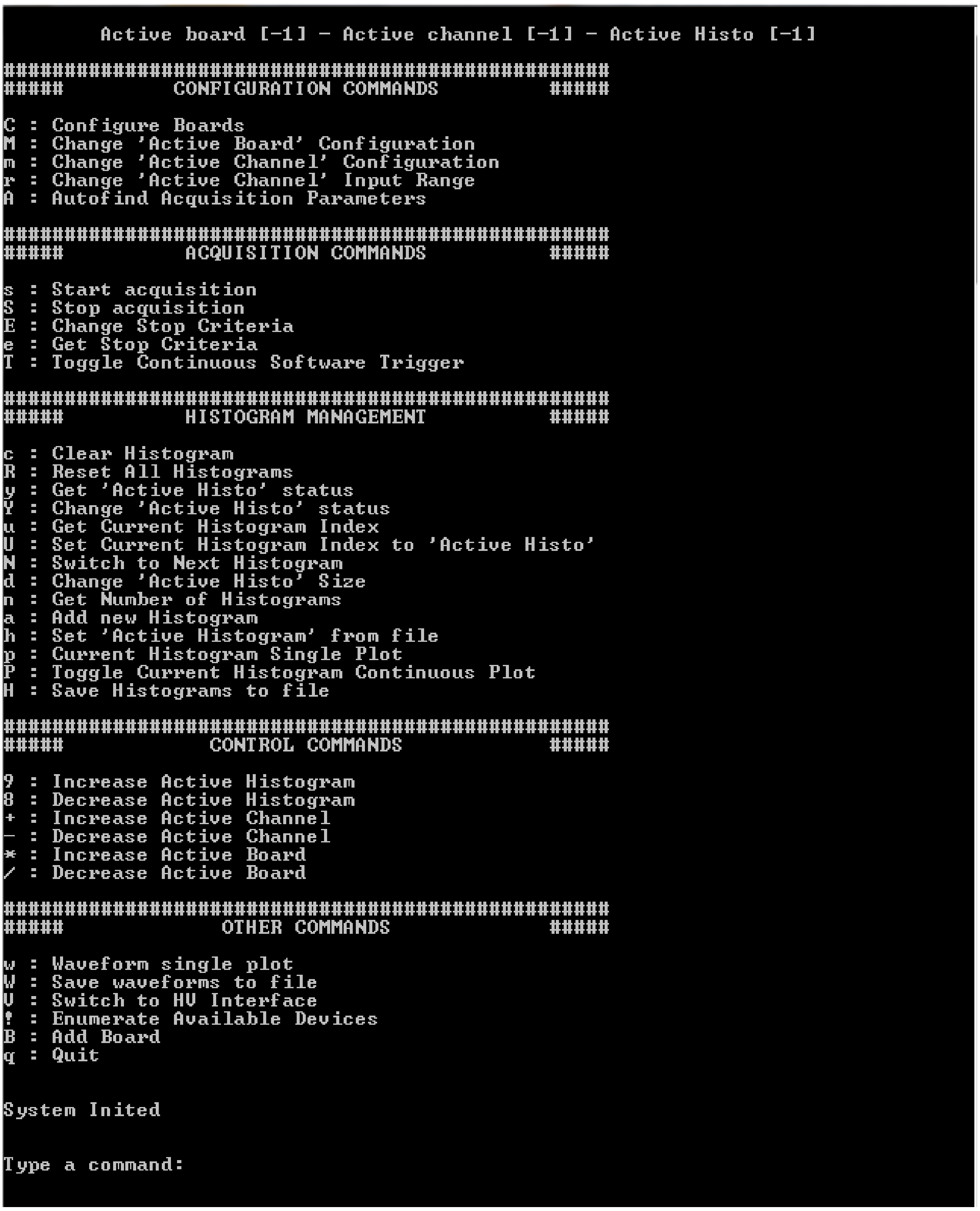


Fig. 7.1: The V1730 CAEN Digitizer.

Launch the DPP Demo according to the instructions given in Section **CAENDPP Demo Installation.** In our example we use the DPPDemo under Windows10 OS, therefore we launch it in VisualStudio.

The Demo main shell should appear, as shown in the following picture. The main shell is divided in 8 sections:

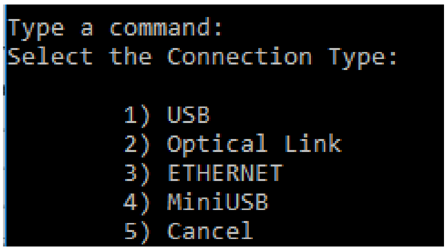
1. A header where the Active Board/Channel/Histo index are shown;
2. A “Configuration Commands” menu;
3. An “Acquisition Commands” menu;
4. A “Histogram Management” menu;
5. A “Control Commands” menu;
6. An “Other Commands” menu;
7. The message area, where all return messages and errors are written. At the first run of the Demo, you should read the message “System Inited”;
8. The command line, where commands must be typed.



```
        Active board [-1] - Active channel [-1] - Active Histo [-1]

##################################################
#####         CONFIGURATION COMMANDS         #####

C : Configure Boards
M : Change 'Active Board' Configuration
m : Change 'Active Channel' Configuration
r : Change 'Active Channel' Input Range
A : Autofind Acquisition Parameters

##################################################
#####          ACQUISITION COMMANDS          #####

s : Start acquisition
S : Stop acquisition
E : Change Stop Criteria
e : Get Stop Criteria
T : Toggle Continuous Software Trigger

##################################################
#####          HISTOGRAM MANAGEMENT          #####

c : Clear Histogram
R : Reset All Histograms
y : Get 'Active Histo' status
Y : Change 'Active Histo' status
u : Get Current Histogram Index
U : Set Current Histogram Index to 'Active Histo'
N : Switch to Next Histogram
d : Change 'Active Histo' Size
n : Get Number of Histograms
a : Add new Histogram
h : Set 'Active Histogram' from file
p : Current Histogram Single Plot
P : Toggle Current Histogram Continuous Plot
H : Save Histograms to file

##################################################
#####            CONTROL COMMANDS            #####

9 : Increase Active Histogram
8 : Decrease Active Histogram
+ : Increase Active Channel
- : Decrease Active Channel
* : Increase Active Board
/ : Decrease Active Board

##################################################
#####             OTHER COMMANDS             #####

w : Waveform single plot
W : Save waveforms to file
V : Switch to HV Interface
! : Enumerate Available Devices
B : Add Board
q : Quit


System Inited


Type a command:
```

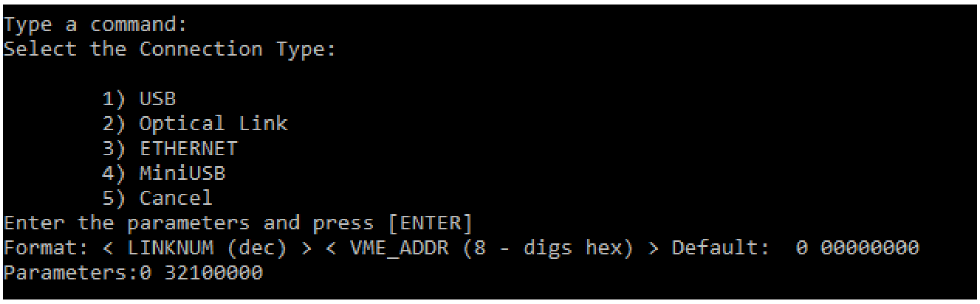
Fig. 7.2: DPPDemo main menu at first run.

First of all we **connect to our board typing the ‘B’ command.** A **Connection Type menu** will appear



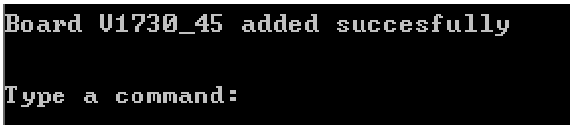
```
Type a command:
Select the Connection Type:

        1) USB
        2) Optical Link
        3) ETHERNET
        4) MiniUSB
        5) Cancel
```

Since we use a **USB** bridge connection with V1718 bridge, we **type ‘1’.** A message will appear requiring to enter the **address parameters**. We **type ‘0 32100000’**, which is the VME address of our board and we **press ‘Enter’**.



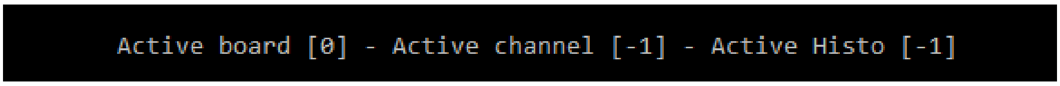
```
Type a command:
Select the Connection Type:

        1) USB
        2) Optical Link
        3) ETHERNET
        4) MiniUSB
        5) Cancel
Enter the parameters and press [ENTER]
Format: < LINKNUM (dec) > < VME_ADDR (8 - digs hex) > Default:  0 00000000
Parameters:0 32100000
```

If the address parameters are correct for our connection configuration, a message confirming the added board wil appear.



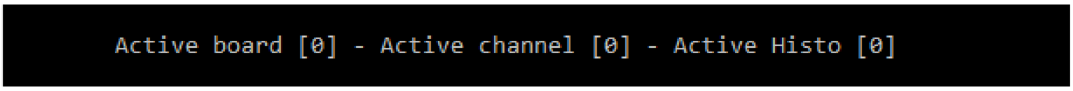
```
Board V1730_45 added succesfully

Type a command:
```

We can now set a correct **Active Board Index,** which is shown in the **top part of the command shell.** At first run, the default value is [-1], so we want to set it to [0]. In order to do this, we **type the command ‘*’**. As explained in the Control Command menu, this command will increase the Active Board index. It is possible to check in the top part of the main menu that the **Active Board Index is [0].**



```
        Active board [0] - Active channel [-1] - Active Histo [-1]
```

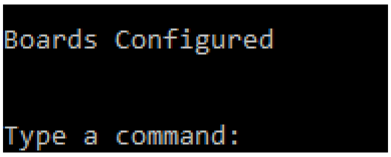
We can do the same thing with the Active Channel Index and the Active Histo Index. According to the Control Commands menu, **we type ‘+’ and then ‘9’ to change the indexes to [0].** Again we can check in the top part of the command shell that alla indexes are now [0].



```
        Active board [0] - Active channel [0] - Active Histo [0]
```

Note: The channel index value must be the same of the Digitizer or MCA channel number we want to acquire.

We can now **configure the board typing ‘C’** command. A confirm message will appear:

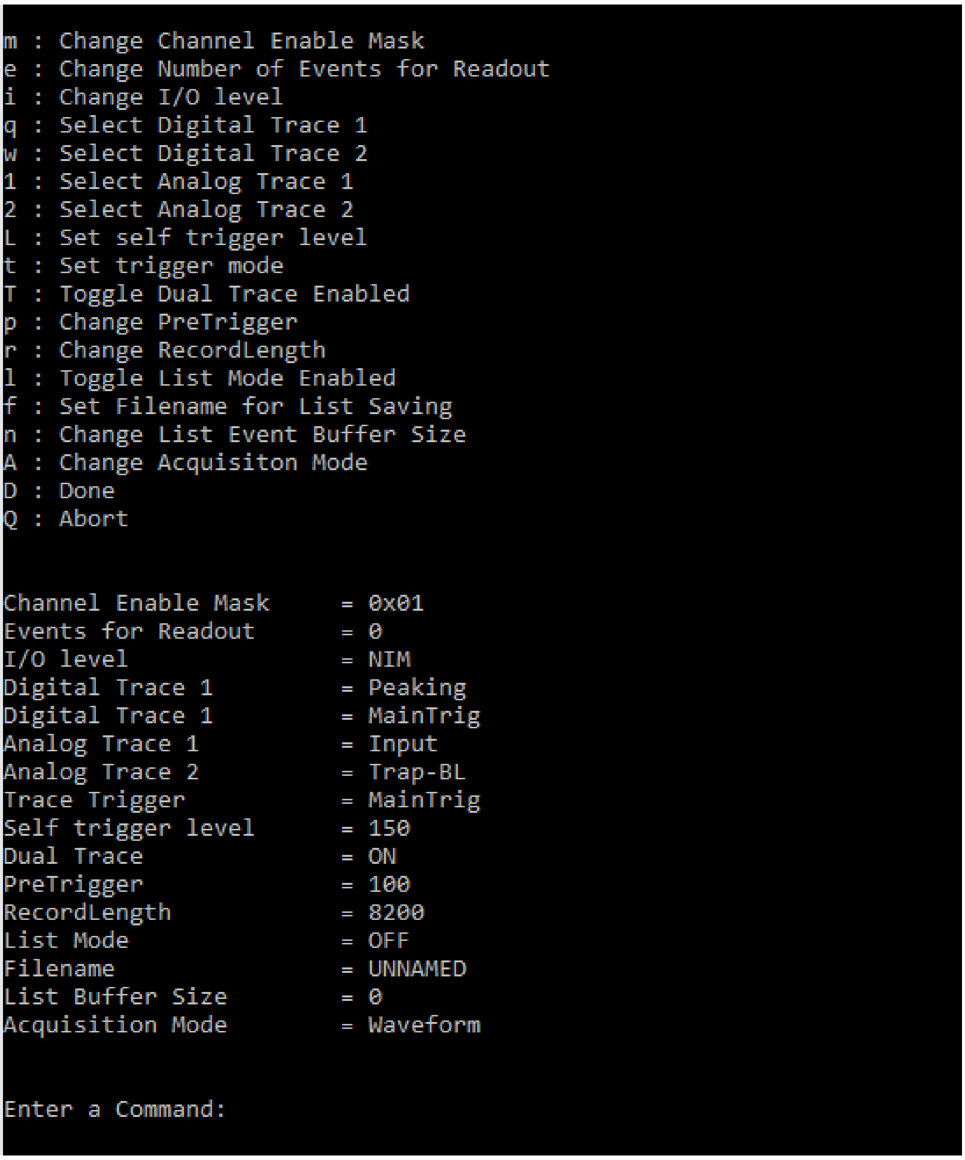


```
Boards Configured

Type a command:
```

We are now ready to set the board and channel parameters, suitable for acquiring our input signal.

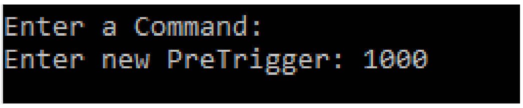
In our specific case we use an **exponential decay signal** on Channel 0 at a **Poissonian rate** of **1 kHz,** with **positive polarity, decay time** of **50 us, rise time** of **0.1 us** and the **energy spectrum of ^{60}Co.** We used a DT5800D CAEN Emulator **[RD8]** to emulate the ^{60}Co energy spectrum acquired with a Germanium (HPGe) detector with a Charge-Sensitive Preamplifier.

We start setting the **Board Parameters.** In order to do this, we **type the 'M' command**, which will open the Board Parameters menu.



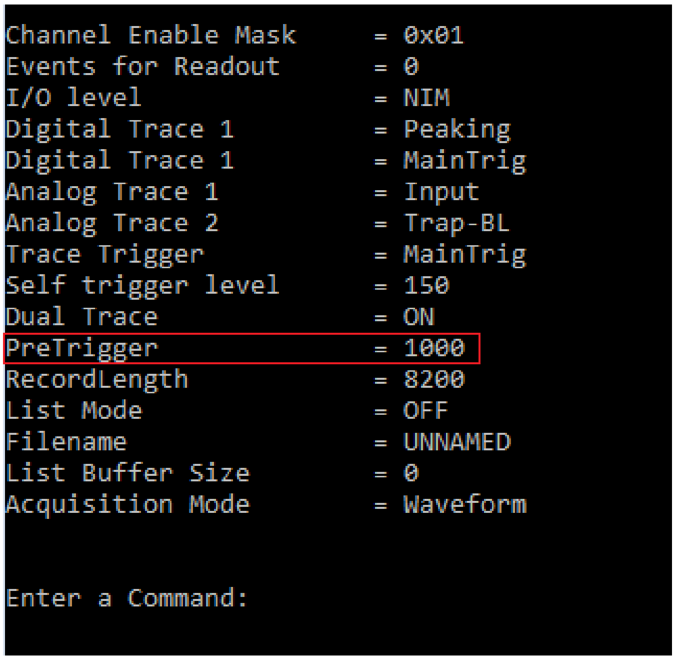
```
m : Change Channel Enable Mask
e : Change Number of Events for Readout
i : Change I/O level
q : Select Digital Trace 1
w : Select Digital Trace 2
1 : Select Analog Trace 1
2 : Select Analog Trace 2
L : Set self trigger level
t : Set trigger mode
T : Toggle Dual Trace Enabled
p : Change PreTrigger
r : Change RecordLength
l : Toggle List Mode Enabled
f : Set Filename for List Saving
n : Change List Event Buffer Size
A : Change Acquisiton Mode
D : Done
Q : Abort


Channel Enable Mask     = 0x01
Events for Readout      = 0
I/O level               = NIM
Digital Trace 1         = Peaking
Digital Trace 1         = MainTrig
Analog Trace 1          = Input
Analog Trace 2          = Trap-BL
Trace Trigger           = MainTrig
Self trigger level      = 150
Dual Trace              = ON
PreTrigger              = 100
RecordLength            = 8200
List Mode               = OFF
Filename                = UNNAMED
List Buffer Size        = 0
Acquisition Mode        = Waveform


Enter a Command:
```

The default value of each parameter is shown in the screenshot above. We only change the **PreTrigger** length. Therefore we **type the 'p' command**, we **insert the desired value** (in our case 1000 samples) and we **press 'Enter'**.



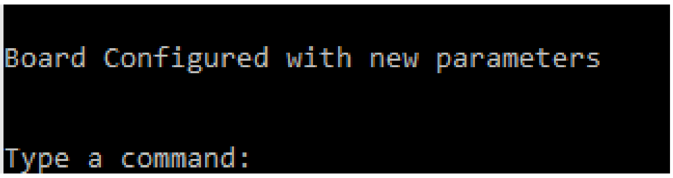
```
Enter a Command:
Enter new PreTrigger: 1000
```

We will see the changed value in the Board Parameters menu



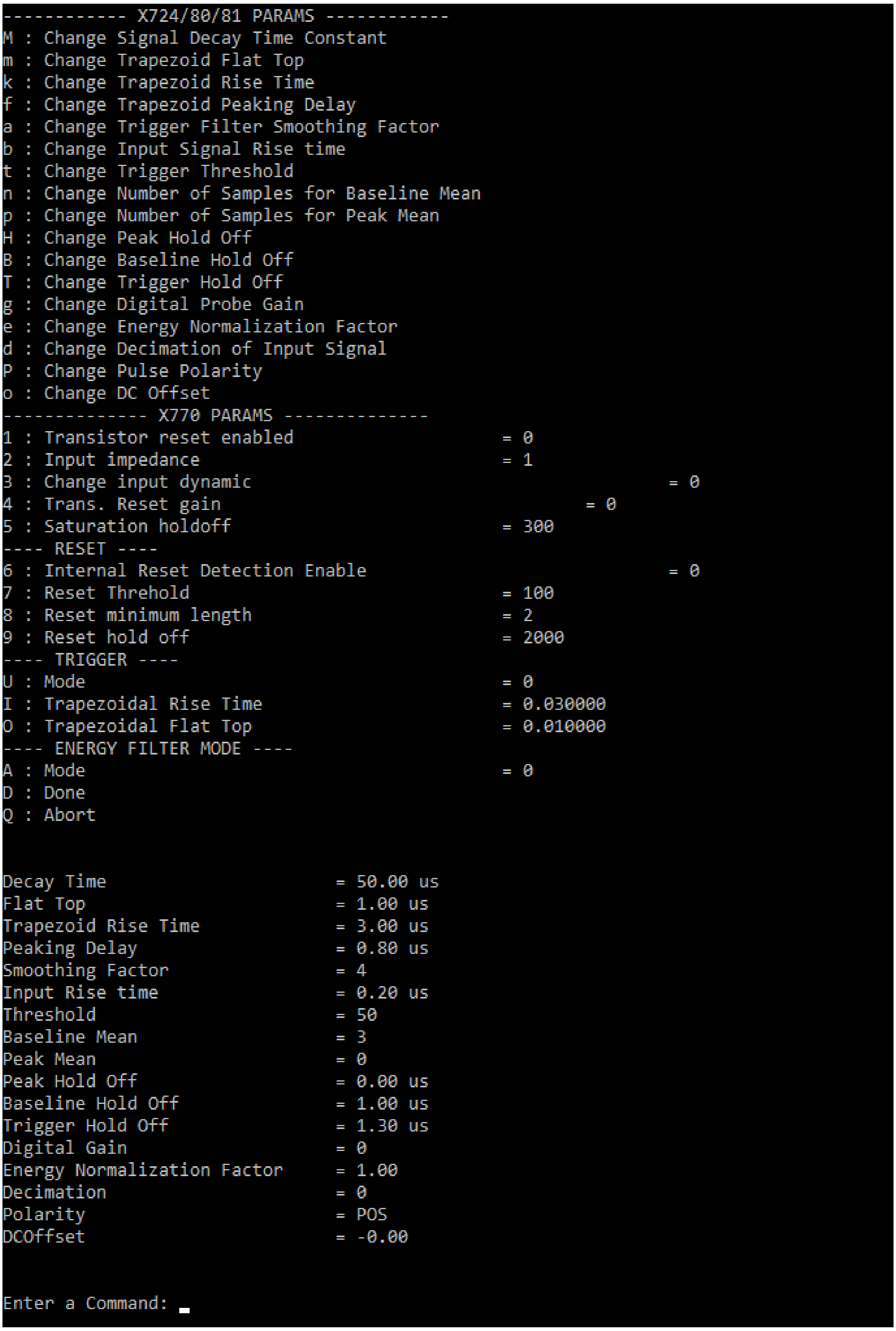
```
Channel Enable Mask      = 0x01
Events for Readout       = 0
I/O level                = NIM
Digital Trace 1          = Peaking
Digital Trace 1          = MainTrig
Analog Trace 1           = Input
Analog Trace 2           = Trap-BL
Trace Trigger            = MainTrig
Self trigger level       = 150
Dual Trace               = ON
PreTrigger               = 1000
RecordLength             = 8200
List Mode                = OFF
Filename                 = UNNAMED
List Buffer Size         = 0
Acquisition Mode         = Waveform


Enter a Command:
```

We **confirm the set Board Parameters typing 'D'** (Done)**.** The Demo will go back to the main menu and a confirm message will appear:



```
Board Configured with new parameters


Type a command:
```

We now set the **Channel 0 Parameters.** In order to do this we **type the 'm' command**, which will open the Channel Parameters menu.

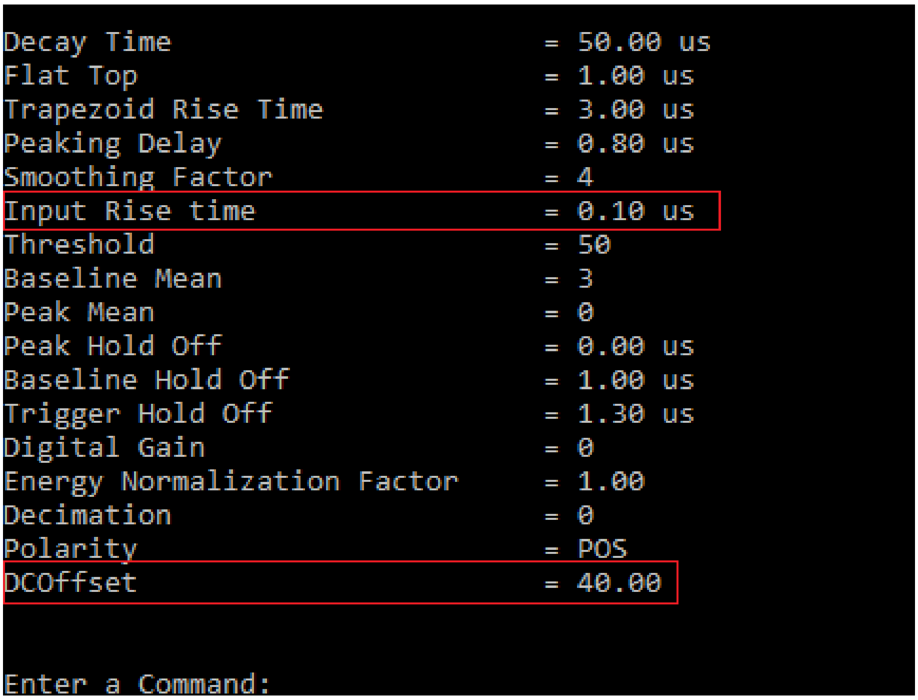


```
------------ X724/80/81 PARAMS ------------
M : Change Signal Decay Time Constant
m : Change Trapezoid Flat Top
k : Change Trapezoid Rise Time
f : Change Trapezoid Peaking Delay
a : Change Trigger Filter Smoothing Factor
b : Change Input Signal Rise time
t : Change Trigger Threshold
n : Change Number of Samples for Baseline Mean
p : Change Number of Samples for Peak Mean
H : Change Peak Hold Off
B : Change Baseline Hold Off
T : Change Trigger Hold Off
g : Change Digital Probe Gain
e : Change Energy Normalization Factor
d : Change Decimation of Input Signal
P : Change Pulse Polarity
o : Change DC Offset
-------------- X770 PARAMS --------------
1 : Transistor reset enabled                    = 0
2 : Input impedance                             = 1
3 : Change input dynamic                                        = 0
4 : Trans. Reset gain                                   = 0
5 : Saturation holdoff                          = 300
---- RESET ----
6 : Internal Reset Detection Enable                             = 0
7 : Reset Threhold                              = 100
8 : Reset minimum length                        = 2
9 : Reset hold off                              = 2000
---- TRIGGER ----
U : Mode                                        = 0
I : Trapezoidal Rise Time                       = 0.030000
O : Trapezoidal Flat Top                        = 0.010000
---- ENERGY FILTER MODE ----
A : Mode                                        = 0
D : Done
Q : Abort


Decay Time                      = 50.00 us
Flat Top                        = 1.00 us
Trapezoid Rise Time             = 3.00 us
Peaking Delay                   = 0.80 us
Smoothing Factor                = 4
Input Rise time                 = 0.20 us
Threshold                       = 50
Baseline Mean                   = 3
Peak Mean                       = 0
Peak Hold Off                   = 0.00 us
Baseline Hold Off               = 1.00 us
Trigger Hold Off                = 1.30 us
Digital Gain                    = 0
Energy Normalization Factor     = 1.00
Decimation                      = 0
Polarity                        = POS
DCOffset                        = -0.00


Enter a Command: _
```

Given the characteristics of our input signal, we set

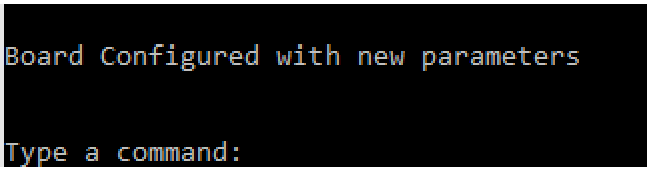
- **Input Rise time = 0.1 us (type ‘b’ → 0.1 → Enter)**
- **DCOffset = 40% (type ‘o’ → 40 → Enter)**

Again you will see the changed parameters in the menu.



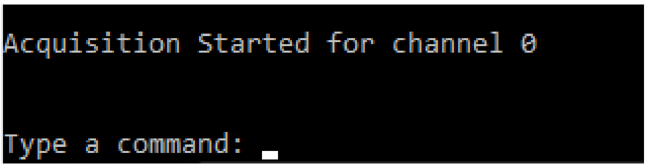
```
Decay Time                         = 50.00 us
Flat Top                           = 1.00 us
Trapezoid Rise Time                = 3.00 us
Peaking Delay                      = 0.80 us
Smoothing Factor                   = 4
Input Rise time                    = 0.10 us
Threshold                          = 50
Baseline Mean                      = 3
Peak Mean                          = 0
Peak Hold Off                      = 0.00 us
Baseline Hold Off                  = 1.00 us
Trigger Hold Off                   = 1.30 us
Digital Gain                       = 0
Energy Normalization Factor        = 1.00
Decimation                         = 0
Polarity                           = POS
DCOffset                           = 40.00


Enter a Command:
```

Confirm the set Channel Parameters typing 'D' (Done)**.** The Demo will go back to the main menu and a confirm message will appear:



```
Board Configured with new parameters


Type a command:
```

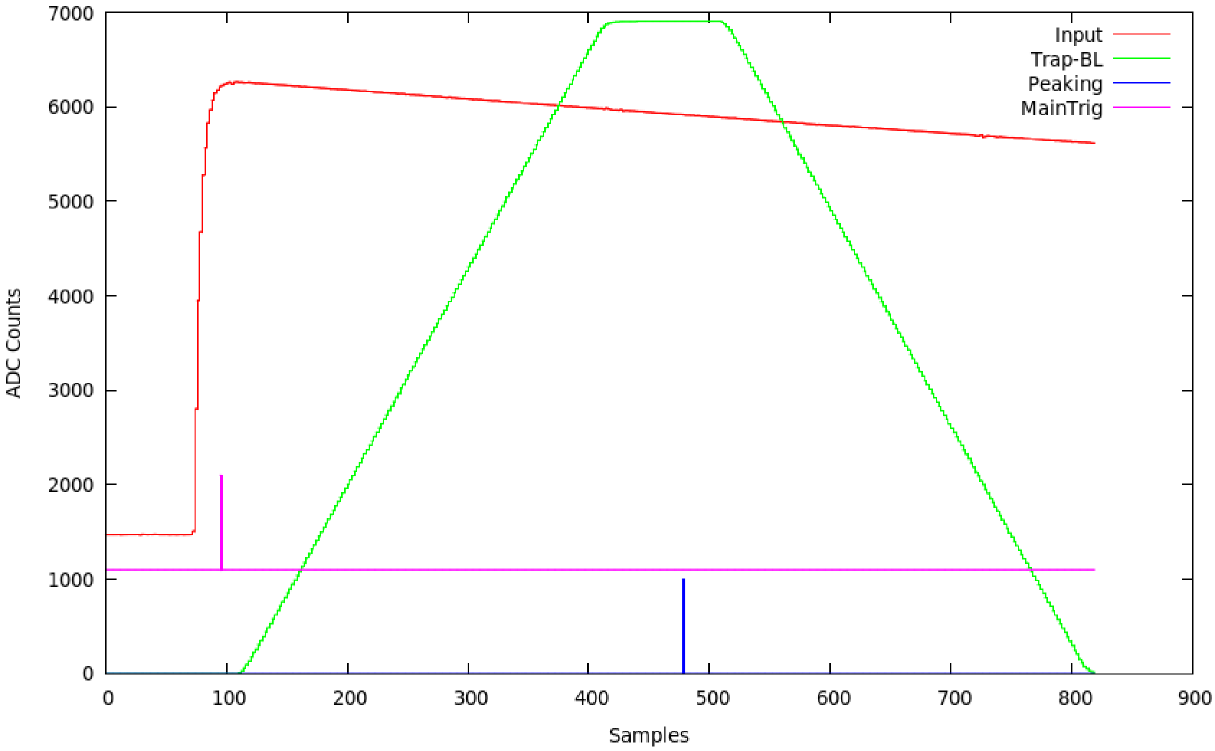
We are now ready to **start the acquisition** on channel 0 and **visualize the acquired waveform. Type 's' to start the acquisition** on channel 0. A confirm message will appear:



```
Acquisition Started for channel 0


Type a command: _
```

Type 'w' to acquire a waveform single plot. The gnuplot window will be opened, showing the acquired plot. Our result is shown in the picture below. The Input signal is shown in red, together with the Main Trigger, the Trapezoid Energy filter and the Peaking signal. The energy is calculated through the coincidence of Trapezoid flat top and the Peaking signal.

Note: Even if your input signal has negative polarity, it is shown with positive polarity in the waveform plot. This is due to the PHA algorithm.

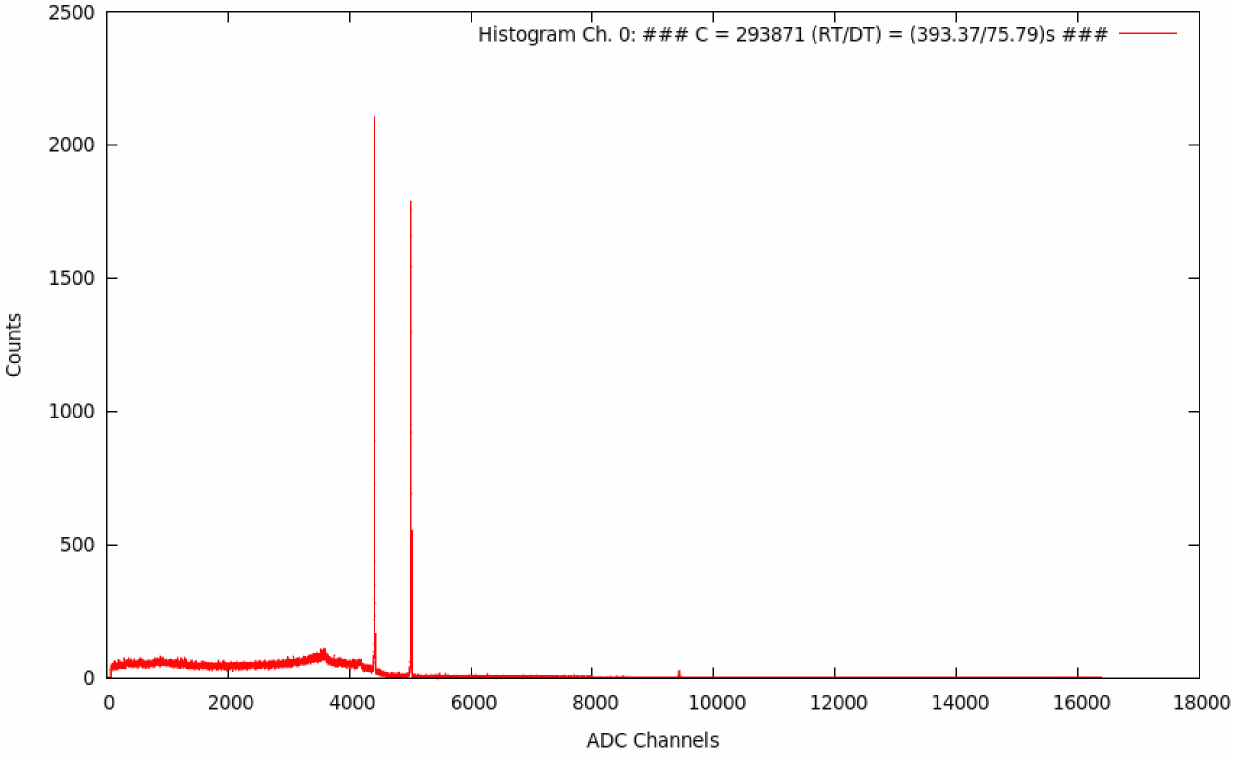


We can save the waveform to file typing 'W'. The output file is saved under the path

C:\Program Files\CAEN\Digitizers\DPPLibrary\DemoDPPLib\build\waveform_ch0.txt

Linux users can find the output file under the directory from which the Demo was launched.

It is possible to **switch to the Histogram plot mode by typing 'P'.** This command enables the **current index histogram continuous plot.** According to the emulated imput signal, we obtain the typical ^{60}Co spectrum acquired with a HPGe detector, as shown in the picture below.



We can save the histograms to file typing 'H'. The output file is saved under the path

C:\Program Files\CAEN\Digitizers\DPPLibrary\DemoDPPLib\build\histo_ch0.txt

Linux users can find the output file under the directory from which the Demo was launched.

We can decide in every moment to **Disable the Histogram Continous plot** by retyping **'P'**, **stop the acquisition** typing **'S'**, or we can directly **quit the Demo** typing **'q'**.

8 Appendix

In this Appendix we give the definitions and brief descriptions of some specific CAENDPPLib types used for Gamma Stream, Hexagon and X770 modules. All these definitions refer to the function **StartBoardParametersGuess**.

```
//----------------
//Types Definition for Gamma Stream
//----------------

typedef struct {
    int32_t enabled;
    int32_t LLD;
    int32_t ULD;
} CAENDPP_TempCorrParams_t;
//----------------

typedef struct {
    char RunName[MAX_RUNNAME]; // run name
    int32_t RunDurationSec; // duration time (s)
    int32_t PauseSec; // pause duration (s)
    int32_t CyclesCount; // number of cycles to do
    uint8_t SaveLists;
    uint8_t GPSEnable;
    uint8_t ClearHistos;
} CAENDPP_RunSpecs_t;
//----------------

typedef struct {
    CAENDPP_RunSpecs_t RunSpecs; // Run specifications
    int32_t RunID;               // run ID (from '0' to 'RunSpecs.CyclesCount - 1')
} CAENDPP_RunInfo_t;



//----------------
//Types Definition for Hexagon boards
//----------------

// Input Coupling codification.
typedef enum {
    CAENDPP_INCoupling_DC      = 0,
    CAENDPP_INCoupling_AC_5us  = 1,
    CAENDPP_INCoupling_AC_10us = 2,
    CAENDPP_INCoupling_AC_20us = 3,
} CAENDPP_INCoupling_t;



//----------------
//Types Definition for X770
//----------------

typedef struct {
    int32_t trigK;      // trigger fast trapezoid rising time
    int32_t trigm;      // trigger fast trapezoid flat top
    int32_t trigMODE;   // 0 threshold on fast trapeziodal
                        // 1 For future use
    int32_t energyFilterMode; // 0 trapezoidal
                              // 1 For future use

    CAENDPP_InputImpedance_t InputImpedance;

    uint32_t CRgain;          // Continuous Reset Analog Gain
    uint32_t TRgain;          // Pulsed Reset Analog Gain

    uint32_t SaturationHoldoff;    // Saturation detector holdoff

    CAENDPP_GPIOConfig_t GPIOConfig; // GPIO Configuration for X770
} CAENDPP_ExtraParameters;
//----------------

typedef enum {
    CAENDPP_InputImpedance_50O = 0,
    CAENDPP_InputImpedance_1K = 1,
} CAENDPP_InputImpedance_t;
//----------------
```

```
//-----------------
typedef struct {
    CAENDPP_GPIO_t GPIOs[MAX_GPIO_NUM];
    CAENDPP_TriggerControl_t TRGControl;
    CAENDPP_GPIOLogic_t GPIOLogic;
    uint32_t TimeWindow;
    uint32_t TransResetLength;
    uint32_t TransResetPeriod;
} CAENDPP_GPIOConfig_t
//-----------------
// NOTE: Analog signals in enum 'CAENDPP_OUTSignal_t' and parameters
// 'DACInvert' and 'DACOffset' are only supported in GPIO1
typedef struct {
    CAENDPP_GPIOMode_t Mode;
    CAENDPP_OUTSignal_t SigOut;
    uint8_t DACInvert; // 0 -> not inverted; 1 -> Inverted
    uint32_t DACOffset;
} CAENDPP_GPIO_t;
//-----------------

typedef enum {
    CAENDPP_GPIOMode_OUTSignal = 0, // GPIO used as OUTPUT for an analog or digitial signal
                                       (see enum 'CAENDPP_OUTSignal_t')
    CAENDPP_GPIOMode_INTrigger = 2, // GPIO used as INPUT to implement trigger logic
                                       (see enums 'CAENDPP_GPIOLogic_t' and
                                       'CAENDPP_TriggerControl_t')
    CAENDPP_GPIOMode_INReset    = 3 // GPIO used as INPUT for external reset signal
                                        (Trans. Reset only)
} CAENDPP_GPIOMode_t;
//-----------------

// NOTE: Analog signals are only supported by GPIO1
typedef enum {
    CAENDPP_OUTSignal_OFF = 0,
    CAENDPP_OUTSignal_Digital_Trigger               = 1,
    CAENDPP_OUTSignal_Digital_ESample               = 2,
    CAENDPP_OUTSignal_Digital_BLSample              = 3,
    CAENDPP_OUTSignal_Digital_ResetDetected         = 4,
    CAENDPP_OUTSignal_Digital_Running               = 5,
    CAENDPP_OUTSignal_Digital_Saturation            = 6,
    CAENDPP_OUTSignal_Digital_PUR                   = 7, // PileUp Rejection
    CAENDPP_OUTSignal_Digital_PUI                   = 8, // PileUp Inhibit
    CAENDPP_OUTSignal_Digital_TRESETPeriodic        = 9, // Reset Over Threshold
    CAENDPP_OUTSignal_Digital_CLKHALF               = 10,
    CAENDPP_OUTSignal_Digital_BLInhibit             = 11,
    CAENDPP_OUTSignal_Digital_SCA1                  = 12, // Single channel analyzer 1
    CAENDPP_OUTSignal_Digital_SCA2                  = 13, // Single channel analyzer 1
    CAENDPP_OUTSignal_Analog_Input                  = 100,
    CAENDPP_OUTSignal_Analog_FastTrap               = 101,
    CAENDPP_OUTSignal_Analog_Baseline               = 102,
    CAENDPP_OUTSignal_Analog_Trapezoid              = 103,
    CAENDPP_OUTSignal_Analog_Energy                 = 104,
    CAENDPP_OUTSignal_Analog_TrapCorrected          = 105,
} CAENDPP_OUTSignal_t;
//-----------------

// Defines the role played by the internal signal (SIG_IN)
// on the trigger logic (see also enums 'CAENDPP_GPIOMode_t' and 'CAENDPP_GPIOLogic_t')
typedef enum {
    CAENDPP_TriggerControl_Internal    = 0,       // Threshold on Fast Trapezoid is used as
                                                   trigger(normal mode)

    CAENDPP_TriggerControl_ON          = 5,      // SIG_IN EDGE is used as internal trigger
    CAENDPP_TriggerControl_OFF         = 7,       // Trigger is inhibited
    CAENDPP_TriggerControl_Gate         = 1,      // SIG_IN is used as GATE for the internal
                                                   Trigger

    CAENDPP_TriggerControl_GateWin     = 3,      // SIG_IN edge opens a programmable GATE
                                                   window for the internal trigger

    CAENDPP_TriggerControl_Coincidence  = 6,    //like 'CAENDPP_TriggerControl_GateWin' but
                                                   only the first trigger is kept.

    CAENDPP_TriggerControl_Veto         = 2,      // SIG_IN is used as VETO for the internal
                                                   trigger

    CAENDPP_TriggerControl_VetoWin      = 4         // SIG_IN edge opens a programmable VETO
                                                    window for the internal trigger

} CAENDPP_TriggerControl_t;
//-----------------
```

```
//----------------
// Set how to generate the internal signal (SIG_IN) used for trigger
// logic (see enums 'CAENDPP_GPIOMode_t' and 'CAENDPP_TriggerControl_t')
typedef enum {
    CAENDPP_GPIOLogic_AND = 0, // logic AND of GPIO1/2 generates [SIG_IN]
    CAENDPP_GPIOLogic_OR = 1   // logic OR of GPIO1/2 generates [SIG_IN]
} CAENDPP_GPIOLogic_t;

//----------------
//Spectrum Control settings
typedef struct {
    CAENDPP_SpectrumMode_t SpectrumMode;       // 0 Energy
                                               // 1 Time distribution
    uint32_t TimeScale;       // Scale in time distribution
} CAENDPP_SpectrumControl;
//----------------

typedef enum {
    CAENDPP_SpectrumMode_Energy     = 0,
    CAENDPP_SpectrumMode_Time       = 1
} CAENDPP_SpectrumMode_t;


//----------------
//Constants Definition
//----------------

#define MAX_GPIO_NUM        2
#define MAX_RUNNAME         128
```

9 Technical Support

CAEN experts can provide technical support at the e-mail addresses below:

support.nuclear@caen.it
(for questions about the hardware)

support.computing@caen.it
(for questions about software and libraries)

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 Vetraia, 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

CAEN Tools for Discovery **Electronic Instrumentation**

UM3185 - CAENDPP Library User Manual Rev. 1 - July 5th, 2017

00000-00-DPP00-MUTX