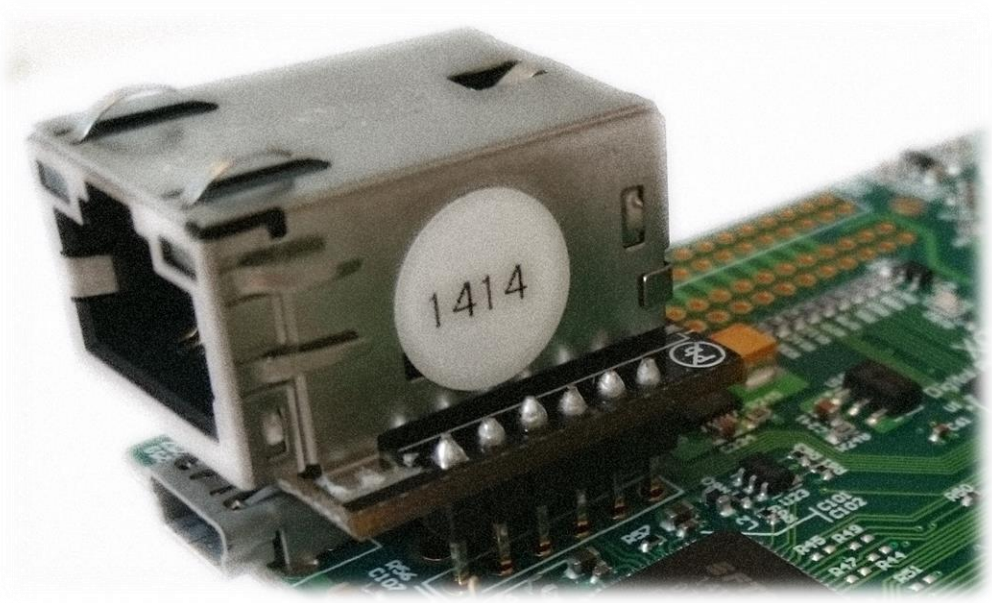
XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)
e-mail: info@xglab.it
ph: +39 02 49660460
P. IVA/C.F. 06557660963
Capitale Sociale 50.000 euro I.V., R.E.A. MI-1899304

Milan, July 28th, 2021
Manual revision: 4.1

DANTE Application Programming Interface



Contents

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001
BUREAU VERITAS
Certification

- confidential -

Intended audience

This manual details the use of the library to control DANTE Digital Pulse Processor Systems connected using the USB or TCP-IP Ethernet protocols.

It is intended for software developers needing to integrate DANTE systems in a custom acquisition software.

System requirements

The software is available for Windows Vista, Windows 7, 8, 8.1, 10, both x32 and x64. The library is available also for Linux 64 bit and requires gcc version 4.4.6 or newer.

A USB 2.0 port or an Ethernet connection is required to operate the system.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Linux installation

Some prerequisites are required to operate the library under Linux. In particular, USB drivers have to be manually installed because Linux by default loads a wrong type of driver for USB communication with our hardware.

1) Inside the software package you will find a directory named Drivers/libftd2xx-x86_64-1.4.8. Execute the steps described in section 2.1.1 Native Compiling of the "AN_220_FTDI_Drivers_Installation_Guide_for_Linux.pdf" guide, located in such folder.

2) If not already installed, install the libusb package with the following commands:

   sudo apt-get install libusb-1.0-0-dev

   sudo apt-get install libusb-dev

   or equivalent commands if not in a Debian environment.

3) Launch the script ftdi_config.sh that is located in the directory libftd2xx-x86_64-1.4.8.

4) In order to automatically load the device drivers without root permissions, the user has to added to a special group, so launch this command:

   sudo usermod -aG usb <username>

   where <username> it's the name of your Linux user.

5) Reboot.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

Python

The DANTE library is compatible with Python (2.7.15 tested, newer 2.x version are compatible, 3.x versions instead is currently supported only for Windows). The .dll (Windows) or .so (Linux) files can be renamed to *.pyd* Python modules and imported in Python like any other module (by using: *import XGL_DPP*).

All the conversions between C++ types and Python types are handled by the library, so that a Python user only needs to call the library functions by using standard Python syntax and types.

However, the standard interface of the library makes extensive use of strings or integer arguments passed by reference, and of arrays. In Python, strings and integers are *immutable* types, and so they cannot (easily) be passed as reference arguments for functions.

Therefore, specialized versions of each function have been declared to better match the Python philosophy. For each function *Xxx*, a *pyXxx* equivalent is also declared (for example *pyInitLibrary*) in which the main difference is that arguments are only passed by value, or when a reference is passed it will be a const reference, in the sense that won't be modified (like string inputs). In other words, all the arguments are inputs to the function.

The outputs of the function will be returned in the return value, that will be just a bool for some functions, while others that needs to return more data types will return a tuple (i.e. a Python standard type that implements an immutable list). In this case, reference or raw pointers arguments of the standard interface are moved in the return value for Python interface.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Introduction

The provided library is used to control the main system functions to integrate it into a customer system through a C/C++ program (or any other language that may use a library with an ANSI-C API).

The library is also compatible with Python (2.7.15 or newer 2.x versions, 3.x currently supported in windows only), see 'Python' section for details.

The library is available for dynamic inclusion into the customer software. The supported compilers for Windows are Microsoft C++ and Embarcadero C++ Builder. An import library (.lib) is included in the software for both of them. The supported compiler for Linux is GCC. Other compilers may be used although the export library may not be compatible. It should be always possible to include the library through dynamic loading of the library and declaration of the required functions. Please consider that other compilers are not supported.

Technical notes:
The calling convention used for the functions is __cdecl. If required for any reason, feel free to contact us for a specialized version with a different convention.

For integers, the library interface uses the intXX_t types of the standard library, introduced from C++ 11. This avoids any type of misunderstandings about integers size because they are independent on the OS architecture, e.g. 32 vs. 64 bit.

Size of these types must be respected to correctly interface the library with your software (for example a Boolean value in .NET is 4 bytes and needs to be manually marshaled).

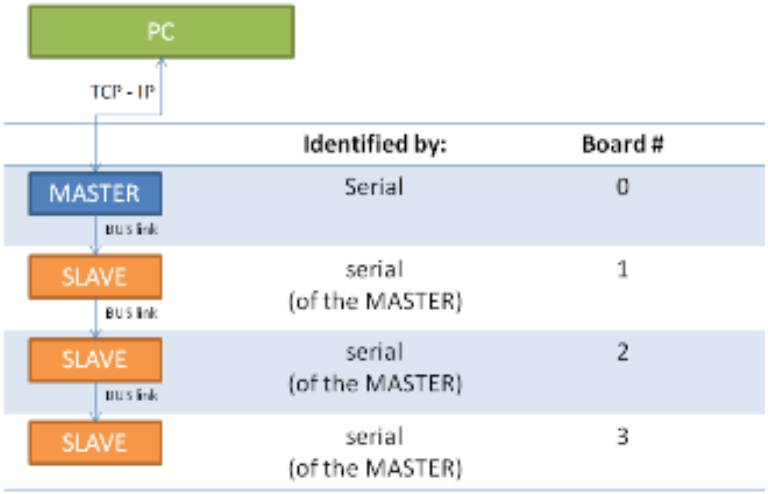
The library can control multiple systems connected to the same host PC, either in Ethernet or USB. Each board is identified through its IP (TCP devices) or serial number (USB devices). For this reason, in most calls the "identifier" (that is the IP or the serial number) is the variable used to identify which of the connected boards (or chains of boards) is the target of the requested operation.

If more than one board is connected in daisy-chain mode, the library will control the full chain through a single identifier that is the "identifier" of the master board. The other boards are identified through a progressive number (starting from '0' for the first board – *see following image*).

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -



The library functions can be grouped in two categories: functions directly communicating with the boards (which are asynchronous) and functions communicating only with the library (which are instead synchronous). Indeed, it is very convenient to use asynchronous calls in a LAN environment to interact with the device because, if the answer takes long time or in presence of network issues, the caller thread does not remain blocked waiting. Instead other functions will not talk directly with the boards but will write or read internal library data. So, in this case the functions will be blocking and will immediately return data or indicate if the operation was successful.

The asynchronous functions will return immediately a call ID (always incremented) and will give back the answer to the caller in two ways: using callbacks or adding the answers in a queue. In the former case, the callback function will include the call ID among the parameters, so that the user can combine requests and answers. In the latter case, instead, the call ID will be added to the queue for the same purpose.

We distribute two versions of the library: Callback library and Polling library, reflecting these two different behaviors.

The synchronous functions will return a boolean value indicating whether the function was successful or not. If the return value is FALSE, the required operation may have failed to complete. Specialized functions can then be called to detect the error occurred.

The library does not automatically recognize DANTE devices in the network, but it needs to know their IP first. **The IP configuration of DANTE boards can be set through USB, so refer to the USB library manual.** The user can add the IP of DANTE boards to a query, which will be used to poll for connected devices and start communicating with them, when available. The library is able to detect hot-plugged devices, without the need for a library or software reload. However, errors may occur if a device is removed from the system during a command execution. The library should be able to recover this type of errors, anyway.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

Broadcast command

Standard way to use the library is by sending command to each board separately. For some commands, the library supports a "broadcast" address 0xFF which can be used to configure all the boards of the chain with a single command. In this case, only the slave board of the chain will respond to the command, to indicate that all the previous boards have been successfully configured.

Currently, the broadcast address is supported for the following API calls:

- configure()
- configure_gating()
- configure_input()

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

The callback function

If using the Callback library, the user must register, before any other operation, a callback function with this signature:

```
void callback(uint16_t type,
              uint32_t call_id,
              uint32_t length,
              uint32_t* data)
```

The response of a preceding asynchronous function is given by the library calling this function.

The parameter *type* indicates the type of operation to which the answer refers. It can have three values: 1 if it refers to a read operation, 2 if it refers to a write operation, 0 if an error occurred.

The parameter *call_id* returns the call ID of the corresponding request, so that the user can combine answers with requests.

The parameter *length* indicates the number of elements contained in the *data* array.

The array *data* contains the read data, in case of a read command, or a 1, in case of a successful write command.

A more detailed description of the *data* array content will be provided in the definition of each asynchronous function, starting at page 15 of this manual.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

The answer queue

When using the Polling library, answers are inserted into a queue called the answers queue. Its content reflects the same parameters of the callback function: in particular, the parameters are inserted in this order: *call_id*, *type*, *length*, *data*.

Again, a description of the contents of the *data* field will be provided separately for each asynchronous call, starting at page 15 of this manual.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Acquisition modes of the DANTE system

The system may work in different acquisition modes. This section contains details about them and shows what functions are relevant for each mode. Please refer to functions detailed descriptions for other details.

Normal DPP acquisition mode (single spectrum)

Parameter	Description	Condition
Acquisition Time	Min: 1 ms Max: free-running, no limit.	
Bins	1024, 2048, 4096	

The normal DPP acquisition mode is the default acquisition mode: the system measures the energy of each detected event and returns an energy spectrum.

To start operation in this mode, first set the desired DPP parameters by calling the **`configure`** function, then start the acquisition by calling the **`start`** function and eventually retrieve the spectrum (together with acquisition statistics) with the **`getData`** function.

Waveform acquisition mode

The waveform acquisition mode is useful for debug purposes, allowing the user to acquire the raw analog input signal of the system with no further processing.

Start the acquisition with a call to **`start_waveform`** and retrieve waveform samples with **`getData`**.

List mode

Parameter	Description	Condition
Acquisition Time	Min: 1 ms Max: free-running, no limit.	
Bins	1024, 2048, 4096	
Max OCR	2.5 Mcps	* Only if network allows full 100Mbit/s bandwidth *Overall OCR among all the channels controlled by a single USB-TCP/IP

In list mode, the DPP returns the energy and the timestamp (8 ns resolution) of each single detected event.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

Configure the DPP with **configure** and start the acquisition with **start_list**. To readout, first check the number of available events with a call to **getAvailableData** and retrieve data with **getData**.

List Wave mode

In list-wave mode the DPP, for each event, returns:

- energy
- timestamp (8ns resolution)
- waveform triggered by the event detection

Waveform is continuously stored in a circular buffer therefore an *offset* can be used to shift the starting point of the acquired waveform being sure to capture the rising-edge of the event.

Configure the DPP with **configure** and start the acquisition with **start_listwave**. Check how many events are available for readout with **getAvailableData** and retrieve them with **getListWaveData**.

Mapping mode (multiple spectra)

Parameter	Description	Condition
Maximum frame Rate	1ms * N	* Only if network allows full 100Mbit/s bandwidth. * N: number of devices in chain controlled by a single USB-TCP/IP connection.
Bins	4096	
Time between frame	No deadtime	
Maximum number of frames	No limit	Only if frame-rate is sustained by the network
Gating/Trigger	Supported	CMOS, TTL compatible

In mapping mode, the DPP can record multiple spectra of programmable length and return them along with the corresponding statistics.

First set the desired DPP parameters with **configure** function and start the acquisition with **start_map**. Check how many spectra are available for reading with **getAvailableData** in order to allocate enough space. Then retrieve data (together with corresponding statistics) with **getAllData**.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

API function reference

Managing controllable systems and library

The library must be initialized with a call to the InitLibrary function first, otherwise it will not be able to work and to detect any system.

The library supports an arbitrary number of DANTE systems connected to the same PC. The library assigns a unique name to each detected system. To get the number of connected systems and other information, call **`get_dev_number`** function and **`get_ids`** function. The board identifiers, returned by get_ids, are used by many function calls to select the system on which the operations are to be applied. Calls not requiring an identifier will instead operate on all the connected systems.

The synchronous library functions return a boolean value (**`true`**/**`false`**) to signal if the operation completed successfully or if a problem occurs. To get more information about the error originating the issue, the **`getLastError`** function returns the last error detected by the library. Instead, asynchronous functions handle problems by returning the error code using the callback function or by adding it to the queue. A description for returned error codes and other details will be provided in the following descriptions.

The library is thread-safe. If multiple calls by multiple threads happen, the library return an error code **`DLL_MULTI_THREAD_ERROR`** and only the first call is served.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
ACCREDIA

- confidential -

initLibrary

Used to initialize the library. Call this function before any other function, otherwise the other functions will always return false and set **DLL_NOT_INITIALIZED** error.

C++ Declaration:

```
bool InitLibrary ( void )
```

Parameters:

None

Return value:

A boolean, true, if the library is initialized, false if a problem occurs.

Python Declaration:

```
def pyInitLibrary ():

return bool(ret_val)
```

Parameters:

None

Return value:

A boolean, true, if the library is initialized, false if a problem occurs.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

closeLibrary

Used to close the internal library resources. Call this function before unloading the library from memory, otherwise with some softwares (e.g. with NI LabVIEW) you will not be able to re-initialize the library without a software restart.

C++ Declaration:

```
bool CloseLibrary ( void )
```

Parameters:

None

Return value:

A boolean, true, if the library closed successfully, false if a problem occurs.

Python Declaration:

```
def pyCloseLibrary ():

return bool(ret_val)
```

Parameters:

None

Return value:

A boolean, true, if the library closed successfully, false if a problem occurs.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

getLastError

Returns the last detected error. Note that the error code is not reset between function calls. It always reflects the last detected error. For example, if a function fails and the next function works correctly, the returned error code is the one set by the first function call.

C++ Declaration:

```
bool getLastError ( uint16_t& error_code )
```

Parameters:

uint16_t& error_code

An integer reporting the last error detected by the library. The *error_code* variable must be instantiated by the user program.

A simplified C example:

```
#include "XGL_DPP.h" // Include the library header.
uint16_t error_code = DLL_NO_ERROR;
bool result = false;
result = InitLibrary(); // Initialize the library.

if (result) {
            // Do some wrong stuff here...
            // Detect error:
            result = getLastError(error_code); // Get last error code.
            if (result) {
                        switch (error_code) {
                        case DLL_MULTI_THREAD_ERROR:
                        // Do something...
                        break;
                        case DLL_CLOSED:
                        // etc...
}}}
```

Return values:

A boolean, true, if the returned error_code is valid. Returns false and **DLL_MULTI_THREAD_ERROR** if **getLastError** is called while another thread is working with the library.

Python Declaration:

```
def pygetLastError ():

tuple(bool(ret_val), uint16_t(error_code))
```

Parameters:

None

Return values:

```
bool(ret_val)
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
ACCREDIA

- confidential -

A boolean, true, if the returned error_code is valid. Returns false and **DLL_MULTI_THREAD_ERROR** if **getLastError** is called while another thread is working with the library.

Uint16(error_code)

An integer containing the error code.

Return value description

Standard error codes are described in the following table:

Error code	Value	Description
DLL_NO_ERROR	0	No errors occurred.
DLL_MULTI_THREAD_ERROR	1	Another thread has a lock on the library functions.
DLL_NOT_INITIALIZED	4	The library is not initialized. Call InitLibrary before calling anything else.
DLL_CHAR_STRING_SIZE	5	Supplied char buffer size is too short. Return value updated with minimum length.
DLL_ARRAY_SIZE	6	An array passed as parameter to a function has not enough space to contain all the data that the function should return.
DLL_ALREADY_INITIALIZED	42	The library has been already initialized.
DLL_COM_ERROR	57	Communication error or timeout.
DLL_ARGUMENT_OUT_OF_RANGE	60	An argument supplied to the library is out of valid range.
DLL_WRONG_SERIAL	62	The supplied serial is not present in the system.
DLL_TIMEOUT	64	An operation timed out. Result is unspecified.
DLL_CLOSING	67	Error during library closing.
DLL_RUNNING	68	The operation cannot be completed while the system is running.
DLL_WRONG_MODE	69	The function called is not appropriate for the current mode.
DLL_NO_DATA	70	No data to be read.
DLL_DECRYPT_FAILED	71	An error occured during decryption.
DLL_INVALID_BITSTREAM	72	Trying to upload an invalid bistream file.
DLL_FILE_NOT_FOUND	73	The specified file hasn't been found.
DLL_INVALID_FIRMWARE	74	Invalid firmware detected on one board. Upload a new one with load_firmware function.
DLL_UNSUPPORTED_BY_FIRMWARE	75	Function not supported by current firmware of the board. Or firmware on the board is not present or corrupted.
DLL_THREAD_COMM_ERROR	76	Error during communication between threads.
DLL_MISSED_SPECTRA	78	One or more spectra missed.
DLL_MULTIPLE_INSTANCES	79	Library open by multiple processes.
DLL_THROUGHPUT_ISSUE	80	Download rate from boards at least 15% lower than expected, check your connection speed.
DLL_INCOMPLETE_CMD	81	A communication error caused a command to be truncated.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

DLL_MEMORY_FULL	82	The hardware memory became full during the acquisition. Likely the effective throughput is not enough to handle all the data.
DLL_SLAVE_COMM_ERROR	83	Communication error with slave board
DLL_SOFTWARE_MEMORY_FULL	84	Allowed memory for the library became full. Likely some data/event have been discarded.

If instead, the error code is one of the following, please contact us for further investigation:

Error code	Value	Description
DLL_WIN32API_FAILED_INIT	3	Win32 errors - Debugging.
DLL_WIN32API_GET_DEVICE	7	Win32 errors - Debugging.
DLL_WIN32API	41	Generic WIN32 API error.
DLL_WIN32API_INIT_EVENTS	43	Win32 errors - Debugging.
DLL_WIN32API_RD_INIT_EVENTS	44	Win32 errors - Debugging.
DLL_WIN32API_REPORTED_FAILED_INIT	45	Win32 errors - Debugging - Possible hardware/driver error.
DLL_WIN32API_UNEXPECTED_FAILED_INIT	46	Win32 errors - Debugging.
DLL_WIN32API_MULTI_THREAD_INIT_SET	47	Win32 errors - Debugging.
DLL_WIN32API_LOCK_HMODULE_SET	48	Win32 errors - Debugging.
DLL_WIN32API_HWIN_SET	49	Win32 errors - Debugging.
DLL_WIN32API_WMSG_SET	50	Win32 errors - Debugging.
DLL_WIN32API_READ_SET	51	Win32 errors - Debugging.
DLL_WIN32API_GET_DEVICE_SIZE	52	Win32 errors - Debugging.
DLL_WIN32API_DEVICE_UPD_LOCK_G	53	Win32 errors – Debugging.
DLL_FT_CREATE_IFL	54	FT errors - Debugging.
DLL_FT_GET_IFL	55	FT errors - Debugging.
DLL_CREATE_DEVCLASS_RUNTIME	56	Library errors - Debugging - Possible hardware/driver error.
DLL_CREATE_DEVCLASS_ARGUMENT	58	Library errors - Debugging.
DLL_CREATE_DEVCLASS_COMM	59	Library errors - Debugging.
DLL_RUNTIME_ERROR	61	Generic runtime error.
DLL_WIN32API_HMODULE	65	Win32 errors - Debugging.
DLL_WIN32API_DEVICE_UPD_LOCK_F	66	Win32 errors - Debugging.
DLL_INIT_EXCEPTION	77	Library errors - Debugging.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
ACCREDIA

resetLastError

Resets the "last error" variable to **DLL_NO_ERROR**.

C++ Declaration:

```
bool resetLastError ( void )
```

Parameters:

None

Return values:

A boolean, true, if the function succeeds.

Python Declaration:

```
def pyresetLastError ():

return bool(ret_val)
```

Parameters:

None

Return values:

A boolean, true, if the function succeeds.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

libVersion

Returns the library version string.

C++ Declaration:

```
bool libVersion ( char* version,
                  uint32_t& version_size )
```

Parameters:

char* version

A pointer to a null-terminated C string (array of chars). The memory for this array has to be initialized by the user program.

uint32_t& version_size

An integer passed by reference, set to the maximum length of the version string. If the length is not enough the function returns false and version will be updated with the correct minimum length value.

C example:

```
// Initialize library first!

uint32_t size = 20; // Enough space for the string.
char* version = new char[size];
bool result = false;

result = libVersion(version,size);

If (result) {
       // Output library version to the command prompt:
       printf("Library version is: %s\r\n",version);
}

delete[] version; // Free memory.
```

Outputs: "**Library version is: 2.0.2**".

Return values:

True if the returned value is correct, false if a problem occurs (e.g. string size is not enough).

Python Declaration:

```
def pylibVersion (uint32_t(version_size)):

tuple(bool(ret_val), string(version))
```

Parameters:

Uint32_t(version_size)

An integer passed by reference, set to the maximum length of the version string. If the length is not enough the function returns false and version will be updated with the correct minimum length value.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Return values:

bool(ret_val)

True if the returned value is correct, false if a problem occurs (e.g. string size is not enough).

string(version)

Version of the library

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

add_to_query

Add an IP address to the library to be use for establishing the TCP/IP connection

C++ Declaration:

```
bool add_to_query ( char* address )
```

Parameters:

char* address

A string containing the IP to be added (for example **"192.168.1.120"**).

Return values:

True if the parameter is correct, false if a problem occurs.

Python Declaration:

```
def pyadd_to_query ( string(address) ):

return bool(ret_val)
```

Parameters:

String(address)

A string containing the IP to be added (for example **"192.168.1.120"**).

Return values:

True if the parameter is correct, false if a problem occurs.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

remove_from_query

Remove an IP address from the query.

C++ Declaration:

```
Bool remove_from_query ( char* address )
```

Parameters:

char* address

A string containing the IP to be removed.

Return values:

True if the parameter is correct, false if a problem occurs.

Python Declaration:

```
def pyremove_from_query ( string(address) ):

return bool(ret_val)
```

Parameters:

String(address)

A string containing the IP to be removed.

Return values:

True if the parameter is correct, false if a problem occurs.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

flush_local_eth_conn

Communication with the device is based on 4 sockets. The function can restore the connection in case of broken TCP/IP link between computer and device.

C++ Declaration

```
Bool flush_local_eth_conn ( char* address )
```

Parameters:

char* address

A string containing the IP to be flushed.

Return values:

True if the parameter is correct, false if a problem occurs.

Python Declaration:

```
def pyflush_local_eth_conn ( string( address) )

return bool(ret_val)
```

Parameters:

String(address)

A string containing the IP to be flushed.

Return values:

True if the parameter is correct, false if a problem occurs.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

autoScanSlaves

Enable (or disable) the automatic searching for slave boards.

Recommended: after discovered the connected boards the automatic searching should be deactivated to guarantee a full bandwidth of communication.

C++ Declaration:

```
bool autoScanSlaves ( bool enable )
```

Parameters:

bool enable

True if the automatic searching is active, false if otherwise.

Return values:

True if the command succeeded, false if a problem occurs.

Python Declaration:

```
def pyautoScanSlaves( bool enable ):

return bool(ret_val)
```

Parameters:

bool enable

True if the automatic searching is active, false if otherwise.

Return values:

bool(ret_val)

True if the command succeeded, false if a problem occurs.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

get_dev_number

Get the number of connected devices (chain masters only).

C++ Declaration:

```
bool get_dev_number ( uint16_t& devs )
```

Parameters:

uint16_t& devs

The number of master devices connected.

Return values:

True if the returned value is correct, false if a problem occurs.

Python Declaration:

```
def pyget_dev_number () :

return tuple(bool(ret_val), uint16(devs))
```

Parameters:

None

Return values:

bool(ret_val)

True if the returned value is correct, false if a problem occurs.

uint16(devs)

The number of master devices connected.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

get_ids

Returns devices serial from progressive number 'nb' of connected devices.

C++ Declaration:

```
bool get_ids ( char* identifier,
               uint16_t& nb,
               uint16_t& id_size)
```

Parameters:

char* identifier

A string that contains the identifier of the selected connected device.

uint16_t& nb

The progressive number of connected devices. nb starts from '0'.

uint16_t& id_size

An integer passed by reference, set to the maximum length of the serial string. If the length is not enough the function returns with false and sets the correct size in this variable.

Return values:

True if all the parameters are filled with correct values, false otherwise.

Python Declaration:

```
def pyget_ids (uint16(nb),
               uint16(id_size)) :

return tuple(bool(ret_val),string(identifier))
```

Parameters:

uint16(nb)

The progressive number of connected devices; nb starts from '0'.

uint16(id_size)

An integer set to the maximum length of the serial string. If the length is not enough the function returns false and does not set the output string.

Return values:

bool(ret_val)

True if all the parameters are filled with correct values, false otherwise.

string(identifier)

A string that contains the identifier of the selected connected device.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

getFirmware

Gather information about the firmware version of a specific system. Asynchronous call.

C++ Declaration:

```
uint32_t getFirmware ( const char* identifier,
                       uint16_t Board)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function sets the Major, Minor, Build, Type version of the firmware, in this order, in the data array of the callback function or in the answers queue. Type can be either 1 (firmware optimized for Low Energy acquisitions) or 2 (firmware integrating the high-rate functionality). Otherwise, if the call was not successful, the response will be 0.

Python Declaration:

```
def pygetFirmware ( string(identifier),
                       uint16(Board)):

return uint32(call_id)
```

Parameters:

string(identifier)

A string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function sets the Major, Minor, Build, Type version of the firmware, in this order, in the data array of the callback function or in the answers queue. Type can be either 1 (firmware optimized for Low Energy acquisitions) or 2 (firmware integrating the high-rate functionality). Otherwise, if the call was not successful, the response will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

get_boards_in_chain

Get the number of devices in the chain controlled by the specified master board.

C++ Declaration:

```
bool get_boards_in_chain ( const char* identifier,
                           uint16_t& devs )
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t& devs

The number of devices in the chain.

Return values:

True if all the parameters are filled with correct values, false otherwise.

Python Declaration:

```
def pyget_boards_in_chain ( string(identifier),
                            uint16(devs)):

return tuple(bool(ret_val),uint16(devs))
```

Parameters:

string(identifier)

A string with the identifier of the system to query.

Return values:

bool(ret_val)

True if all the parameters are filled with correct values, false otherwise.

uint16(devs)

The number of devices in the chain.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

write_IP_configuration

Write a new IP configuration to the specified DPP board. The function can be used only if the device is connected with USB.

C++ Declaration:

```
uint32_t write_IP_configuration (  const char* identifier,
                                   const char* IP,
                                   const char* subnet_mask,
                                   const char* gateway)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

const char* IP

String of the IP.

const char* subnet_mask

String of the subnet mask

const char* gateway

String of the gateway.

Return values:

ID code of the reply.

Python Declaration:

```
def pywrite_IP_configuration ( string(identifier),
                               string(IP),
                               string(subnet_mask),
                               string(gateway)):

return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier of the system to query.

string(IP)

String of the IP.

string(subnet_mask)

String of the subnet mask

string(gateway)

String of the gateway.

Return values:

ID code of the reply.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

load_new_firmware

Loads a new firmware via either Ethernet or USB. Supported by firmware version 4 or higher; if an older firmware is present on the board to be updated, please use the legacy load_firmware function.

C++ Declaration:

```
uint32_t load_new_firmware ( const char* identifier,
                             const char* filename
                             uint16_t board_num)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

const char* filename

The filename must contain the path in this format: C:\\ArbDirectory\\firmware_name.bitc

uint16_t board_num

Optional parameter to flash the firmware only on a specific board of a multichannel system. If omitted or 255, a broadcast firmware configuration is invoked, and the firmware is downloaded to all the boards of the system. An out-of-range error is returned in case broadcast configuration is not selected and the number of connected boards is lower than board_num.

Return values:

ID code of the reply.

Python Declaration:

```
def pyload_new_firmware (string(identifier),
                         string(filename),
                         uint16(board_num)):

return uint32(call_id)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

const char* filename

The filename must contain the path in this format: C:\\ArbDirectory\\firmware_name.bitc

uint16_t board_num

Optional parameter to flash the firmware only on a specific board of a multichannel system. If omitted or 255, a broadcast firmware configuration is invoked, and the firmware is downloaded to all the boards of the system. An out-of-range error is returned in case broadcast configuration is not selected and the number of connected boards is lower than board_num.

Return values:

ID code of the reply.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

load_firmware

Loads a new firmware via USB. Legacy function, only to be used in case of firmware corruption.

C++ Declaration:

```
uint32_t load_firmware (   const char* identifier,
                           bool store,
                           const char* filename
                           uint16_t board_num)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

bool store

With store to false, it will load to the system a temporary firmware that will be lost if rebooted. Otherwise with store to true it will store the firmware on memory, and it will persist even after power cycles.

const char* filename

The filename must contain the path in this format: C:\\ArbDirectory\\firmware_name.bitc

uint16_t board_num

Optional parameter only used in case the routine is called to recovery the firmware on a multichannel system. If omitted or 0, the normal firmware procedure is performed, autodetecting the number of devices in the chain; if >0, the recovery procedure is enabled, and the firmware is forced onto the specified number of boards. An out-of-range error is returned in case the number of connected boards is lower than board_num.

Return values:

ID code of the reply.

Python Declaration:

```
def pyload_firmware ( string(identifier),
                      bool(store)
                      string(filename),
                      uint16(board_num)):

return uint32(call_id)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

bool store

With store to false, it will load to the system a temporary firmware that will be lost if rebooted. Otherwise with store to true it will store the firmware on memory, and it will persist even after power cycles.

const char* filename

The filename must contain the path in this format: C:\\ArbDirectory\\firmware_name.bitc

`uint16_t board_num`

Optional parameter only used in case the routine is called to recovery the firmware on a multichannel system. If omitted or 0, the normal firmware procedure is performed, autodetecting the number of devices in the chain; if >0, the recovery procedure is enabled, and the firmware is forced onto the specified number of boards. An out-of-range error is returned in case the number of connected boards is lower than board_num.

Return values:

ID code of the reply.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

get_load_fw_progress

Get the progress of the load firmware operation.

C++ Declaration:

```
bool get_load_fw_progress (double& progress)
```

Parameters:

Double& progress

Floating number indicating the progress state of the firmware load operation, ranging from 0 (operation just begun) to 1 (operation is complete).

Return values:

A boolean true, if the operation succeeded, false otherwise.

Python Declaration:

```
def pyget_load_fw_progress ():

return tuple(bool(ret_val),double(progress))
```

Parameters:

Return values:

bool(ret_val)

A boolean true, if the operation succeeded, false otherwise.

double(progress)

Floating number indicating the progress state of the firmware load operation, ranging from 0 (operation just begun) to 1 (operation is complete).

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

global_reset

Resets the communication on the entire chain.

C++ Declaration:

```
bool global_reset (const char* identifier)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to reset.

Return values:

A boolean true, if the returned number is correct, false otherwise.

Python Declaration:

```
def pyglobal_reset (string(identifier))
```

Parameters:

string(identifier)

A string with the identifier of the system to reset.

Return values:

bool(ret_val)

A boolean true, if the returned number is correct, false otherwise.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

Configuration section

configure_input

Configures the front-end stage: AC/DC coupling, DC input resistance and AC time constant. Asynchronous call.

Recommendation: It is suggested to disable autoScanSlaves during the configuration.

C++ Declaration:

```
uint32_t configure_input ( char* identifier,
                           uint16_t Board,
                           const InputMode mode)
```

Parameters:

char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain. If board is set to 0xFF the same configuration command is sent to all the boards of the chain (broadcast configuration). In such case, all boards in the chain will be configured in parallel and only the last slave board will provide a response.

const InputMode mode

A constant pointer to an enum that defines the front-end configuration.

```
enum InputMode {
       DC_HighImp, // DC coupling, 10 K Ohm input R
       DC_LowImp,  // DC coupling, 1  K Ohm input R
       AC_Slow,    // AC coupling, 22  us time constant
       AC_Fast     // AC coupling, 2 us time constant

};
```

Python Declaration:

```
def pyconfigure_inputmode ( const char* identifier,,
                        uint16(Board),
                        const char* InputmodeStr)
```

Parameters:

char*(identifier)

A null-terminated string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain. If board is set to 0xFF the same configuration command is sent to all the boards

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

of the chain (broadcast configuration). In such case, all boards in the chain will be configured in parallel and only the last slave board will provide a response.

InputmodeStr

A string which defines the input front-end configuration.

```
"DC_RinHighImp" // DC coupling, 10 K Ohm input R
"DC_RinLowImp"  // DC coupling, 1  K Ohm input R
"AC_Slow"     // AC coupling, 22  us time constant
"AC_Fast"     // AC coupling, 2 us time constant
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

configure

Configures the system with the required acquisition configuration. Asynchronous call.

Recommendation: It is suggested to disable autoScanSlaves during the configuration.

C++ Declaration:

```
uint32_t configure ( char* identifier,
                     uint16_t Board,
                     const configuration cfg)
```

Parameters:

char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain. If board is set to 0xFF the same configuration command is sent to all the boards of the chain (broadcast configuration). In such case, all boards in the chain will be configured in parallel and only the last slave board will provide a response.

const configuration cfg

A constant pointer to a constant structure that defines the base system configuration. Memory for this structure must be allocated and maintained by the caller.

The structure is defined with the following fields:

```
 struct configuration {
       uint32_t fast_filter_thr;
       uint32_t energy_filter_thr;
       uint32_t energy_baseline_thr;
       double   max_risetime;
       double   gain;
       uint32_t peaking_time;
       uint32_t max_peaking_time;
       uint32_t flat_top;
       uint32_t edge_peaking_time;
       uint32_t edge_flat_top;
       uint32_t reset_recovery_time;
       double   zero_peak_freq;
       uint32_t baseline_samples;
       bool     inverted_input;
       double   time_constant;
       uint32_t base_offset;
       uint32_t overflow_recovery;
       uint32_t reset_threshold;
       double   tail_coefficient;
       uint32_t other_param;
};
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
- confidential -

Python Declaration:

```
def pyconfigure ( string(identifier),
                  uint16(Board),
                  configuration(cfg)):

return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain. If board is set to 0xFF the same configuration command is sent to all the boards of the chain (broadcast configuration). In such case, all boards in the chain will be configured in parallel and only the last slave board will provide a response.

configuration(cfg)

An object of configuration class which defines the base system configuration.

The structure is initialized with this syntax:

```
cfg = XGL_DPP.configuration()
cfg.fast_filter_thr = 100
...
```

And it is defined with these fields:

```
class configuration:
       uint32(fast_filter_thr)
       uint32(energy_filter_thr)
       uint32(energy_baseline_thr)
       double(max_risetime)
       double(gain)
       uint32(peaking_time)
       uint32(max_peaking_time)
       uint32(flat_top)
       uint32(edge_peaking_time)
       uint32(edge_flat_top)
       uint32(reset_recovery_time)
       double(zero_peak_freq)
       uint32(baseline_samples)
       bool(inverted_input)
       double(time_constant)
       uint32(base_offset)
       uint32(overflow_recovery)
       uint32(reset_threshold)
       double (tail_coefficient)
       uint32(other_param)
```

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Parameter Description

Parameter	Description	Unit	Range
fast_filter_thr	detection threshold for the fast filter	Spectrum BIN	0 - 4096
energy_filter_thr	detection threshold for the energy filter. If 0, energy threshold is disabled.	Spectrum BIN	0 – 4096
energy_baseline_thr	threshold for inhibit baseline calculation if *energy filter threshold* is enabled (higher than zero)	Spectrum BIN	0 – 4096
max_risetime	maximum expected risetime of the detector, used for pileup rejection. Set to '0' to disable fast pileup rejection.	8ns sample	0 – 127
gain	digital gain	Spectrum BIN or ADC LSB	0.01 – peaking_time*2 TP use max_peaking_time in case of high-rate firmware
peaking_time	main energy filter peaking time. with high-rate FW, it represents the min_peaking_time of the filter	32ns sample	range: (1 to 511-flat_top) if standard DANTE FW is used range: (1 to 128-flat_top) if high-rate DANTE FW is used
max_peaking_time	max energy filter peaking time. must be fixed to 0 if standard DANTE FW is used	32ns sample	range: (1 to 128-flat_top) if high-rate DANTE FW is used
flat_top	main energy filter flat top time	32ns sample	1 - 15
edge_peaking_time	peaking time of the fast filter	8ns sample	1 - 31
edge_flat_top	Flattop of the fast filter	8ns sample	1 - 15
reset_recovery_time	reset recovery time	8ns sample	0 - 2^24-1
zero_peak_freq	frequency of zero-peak	kcps	1 – 501
baseline_samples	set number of samples to be used for baseline correction	32ns sample	0,8,16,32,64,128,256,512
inverted_input	'FALSE' for positive ramp/pulses. 'TRUE' for negative ramp/pulses	/	true, false
time_constant	time constant of incoming exponential pulses. To be used for deconvolution. If time_constant is set to '0', deconvolution is disabled.	µs	0 – 100
Base_offset	analog value read by the ADC in AC coupling without incoming events from	ADC BIN	0 – 2^16-1

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

	the radiation source. To be used for deconvolution		
Overflow_recovery	not used, set to 0	/	/
Reset_threshold	reset detection threshold.	ADC BIN	0 to 2^16-1
Tail coefficient	not used, set to 0	/	/
Other_param	bit 5 to bit 2: OCR limit setting. Supported only by ListWave acquisition mode.	See table	See table

Other param description				
Bit 5 to bit 2	**OCR limit**		**Bit 5 to bit 2**	**OCR limit**
0x0	Disabled		**0x8**	8 KHz
0x1	100Hz		**0x9**	10KHz
0x2	200 Hz		**0xA**	20 KHz
0x3	400 Hz		**0xB**	40 KHz
0x4	800 Hz		**0xC**	80 KHz
0x5	1 KHz		**0xD**	100 KHz
0x6	2 KHz		**0xE**	200 KHz
0x7	4 KHz		**0xF**	Disabled

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

configure_offset

Offset the analog waveform to match the ADC dynamic range of $2V_{PP}$. Each board is equipped with two digital potentiometers that can be used for shifting the input signal. Standard approach is to configure both offset_val fields with the same integer value. Asynchronous call.

Recommendation: It is suggested to disable autoScanSlaves during the configuration.

Declaration:

```
uint32_t configure_offset ( char* identifier,
                            uint16_t Board,
                            const configuration_offset cfg_offset)
```

Parameters:

char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

const configuration_offset const cfg

A constant pointer to a structure that defines the base system configuration. Memory for this structure must be allocated and maintained by the caller.

The structure is defined with these fields:

```
struct configuration_offset {
               uint32_t offset_val1;
               uint32_t offset_val2;
};
```

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def pyconfigure_offset ( string(identifier),
                         uint16(Board),
                         configuration_offset(cfg)):

return uint32(call_id)
```

Parameters:

string(identifier)

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

A string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

configuration_offset(cfg)

An object of a structure that defines the base system configuration.

The structure is defined with these fields:

```
class configuration_offset :
                uint32(offset_val1)
                uint32(offset_val2)
```

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Parameter Description

Parameter	Description	Unit	Range
offset_val1	Offset of digpot channel 1	arbitrary	0 - 255
offset_val2	Offset of digpot channel 2	arbitrary	0 - 255

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

configure_gating

Configures gating and triggering functionalities for map (multiple spectra) acquisitions. It has no effect on all other acquisition modalities. Asynchronous call.

Recommendation: It is suggested to disable autoScanSlaves during the configuration.

C++ Declaration:

```
uint32_t configure_gating ( char* identifier,
                            const GatingMode GatingMode,
                            uint16_t Board)
```

Parameters:

char* identifier

A null-terminated string with the identifier of the system to query.

const GatingMode_GatingMode

A constant enum which defines the gating or triggering modality.

```
enum GatingMode {
       FreeRunning,
       TriggerRising,
       TriggerFalling,
       TriggerBoth,
       GatedHigh,
       GatedLow
};
```

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain. If board is set to 0xFF, gating configuration is invoked on all the boards of the chain with the given parameters. In such case, only the last slave board will provide a response to the command.

Return values:

The call ID if the operation succeeds, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def pyconfigure_gating ( string(identifier),
                         string(GatingMode)
                         uint16(Board)
                         ):

return uint32(call_id)
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

Parameters:

string(identifier)

A null-terminated string with the identifier of the system to query.

string(GatingMode)

A string which defines the gating or triggering modality.

```
FreeRunning
TriggerRising
TriggerFalling
TriggerBoth
GatedHigh
GatedLow
```

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain. If board is set to 0xFF, gating configuration is invoked on all the boards of the chain with the given parameters. In such case, only the last slave board will provide a response to the command.

Return values:

The call ID if the operation succeeds, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Parameter Description

Parameter	Description
FreeRunning	the DPP ignores the input digital signal and triggering and gating functionalities are disabled
TriggerRising	in this mode, a new spectrum is acquired whenever a rising edge is detected on the digital input signal of the DPP
TriggerFalling	in this mode, a new spectrum is acquired whenever a falling edge is detected on the digital input signal of the DPP
TriggerBoth	in this mode, a new spectrum is acquired whenever an edge (either falling or rising) is detected on the digital input signal of the DPP
GatedHigh	in this mode, a new spectrum is acquired whenever a high (5 V TTL - 3.3 V CMOS) signal is applied to the digital input of the DPP, and the DPP is disabled when a low (0 V) signal is applied
GatedLow	in this mode, a new spectrum is acquired whenever a low (0 V) signal is applied to the digital input of the DPP, and the DPP is disabled when a high (5V TTL – 3.3V CMOS) signal is applied.

configure_timestamp_delay

Configures timestamp clock delay between adjacent boards in a chain, so that events happening at the same time on different boards receive the same timestamp value. In fact, due to the timestamp clock distribution among the chain, concurrent events detected by different boards suffer from skew if this function is not invoked. Calling configure_timestamp_delay allows to compensate for this skew. Asynchronous call.

Recommendation: It is suggested to disable autoScanSlaves during the configuration.

C++ Declaration:

```
uint32_t configure_timestamp_delay ( char* identifier,
                                     uint16_t Board,
                                     uint16_t CkPulses,
                                     uint16_t TimeShift)
```

Parameters:

char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint16_t CkPulses

Number of integer clock pulses to be subtracted from the detected timestamp value. Due to clock distribution delays, each board accumulates some skew (in the order of few tens of nanoseconds) compared to the one preceding it in the chain; setting this parameter allows to subtract an integer number of clock pulses in order to roughly adjust the timestamps for the selected board compared to the one that precedes in the chain. Fine adjustment can then be applied by means of the TimeShift parameter.

uint16_t TimeShift

This parameter introduces a further delay to the timestamp clock in order to fine-tune the skew with sub-nanosecond resolution. Incrementing by one the TimeShift parameter corresponds to delaying the clock by about 40 ps; a complete calibration procedure can therefore be established, effectively nulling the board-to-board skew.

Return values:

The call ID if the operation succeeds, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def pyconfigure_timestamp_delay ( string(identifier),
                                  uint16(Board)
                                  uint16(CkPulses)
                                  uint16(TimeShift)):
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

```
return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint16_t CkPulses

Number of integer clock pulses to be subtracted from the detected timestamp value. Due to clock distribution delays, each board accumulates some skew (in the order of few tens of nanoseconds) compared to the one preceding it in the chain; setting this parameter allows to subtract an integer number of clock pulses in order to roughly adjust the timestamps for the selected board compared to the one that precedes in the chain. Fine adjustment can then be applied by means of the TimeShift parameter.

uint16_t TimeShift

This parameter introduces a further delay to the timestamp clock in order to fine-tune the skew with sub-nanosecond resolution. Incrementing by one the TimeShift parameter corresponds to delaying the clock by about 40 ps; a complete calibration procedure can therefore be established, effectively nulling the board-to-board skew.

Return values:

The call ID if the operation succeeds, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Acquisition control

These functions are used to control the acquisition state of each system.

isRunning_system

Query the acquisition state for a board (acquisition in progress or board idle). Asynchronous call.

C++ Declaration:

```
uint32_t isRunning_system ( const char* identifier,
                            uint16_t Board )
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def pyisRunning_system ( string(identifier),
                         uint16(Board) ):

return uint32(call_id)
```

Parameters:

string(identifier)

A string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

start

Start a single spectrum for a specified amount of time or in free-running mode. Asynchronous call.

C++ Declaration:

```
uint32_t start ( const char* identifier,
                 const double time,
                 const uint16_t spect_depth )
```

Parameters:

const char* identifier

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

const double time

The amount of time the acquisition should run, expressed in seconds. The system precision is up to 0.001 secs (1 msec) for measurements ranging from 1 msec up to 1 hour. The system stops automatically after the specified time elapses, but the function stop needs to be called anyway before starting another acquisition. Use 0 to start a free-running acquisition (i.e. the system only stops when the user calls the stop function).

const uint16_t spect_depth

The number of bins used for the histogram. This value can be only 1024, 2048 or 4096. The setting affects also the amount data returned by the getData() function.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def pystart ( string(identifier),
              double(time),
              uint16(spect_depth)):

return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

double(time)

The amount of time the acquisition should run, expressed in seconds. The system precision is up to 0.001 secs (1 msec) for measurements ranging from 1 msec up to 1 hour. The system stops automatically after the specified time elapses, but the function stop needs to be called anyway before starting another acquisition. Use 0 to start a free-running acquisition (i.e. the system only stops when the user calls the stop function).

uint16(spect_depth)

The number of bins used for the histogram. This value can be only 1024, 2048 or 4096. The setting affects also the amount data returned by the getData() function.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

stop

Stops a running acquisition. Asynchronous call.

C++ Declaration:

```
uint32_t stop ( const char* identifier )
```

Parameters:

const char* identifier

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

Please note that if the system is not in free-running mode, the system stops automatically after the specified time elapses, but the function stop needs to be called anyway before starting another acquisition.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def pystop ( string(identifier) ):

return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

Please note that if the system is not in free-running mode, the system stops automatically after the specified time elapses, but the function stop needs to be called anyway before starting another measure.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

clear_chain

Clears acquisition data, run time and statistical data for an entire chain.

C++ Declaration:

```
bool clear_chain ( const char* identifier )
```

Parameters:

const char* identifier

A null-terminated string with the identifiers of the system to query.

Return values:

A boolean true if the data is correctly cleared, false otherwise.

Python Declaration:

```
def pyclear_chain ( string(identifier) ):

return bool(ret_val)
```

Parameters:

String(identifier)

A null-terminated string with the identifiers of the system to query.

Return values:

bool(ret_val)

A boolean true if the data is correctly cleared, false otherwise.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

clear_board

Clears acquisition data, run time and statistical data for a single board.

C++ Declaration:

```
bool clear_board ( const char* identifier, uint16_t Board )
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Return values:

A boolean true if the data is correctly cleared, false otherwise.

Python Declaration:

```
def pyclear_board ( string(identifier), uint16(Board) )

return bool(ret_val)
```

Parameters:

String(identifier)

A null-terminated string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Return values:

A boolean true if the data is correctly cleared, false otherwise.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

start_waveform

Start the acquisition in waveform acquisition mode, for a specified amount of time and with additional acquisition settings (trigger, trigger level, decimation ratio, length). The function stop must be called before starting another acquisition.

Asynchronous call.

C++ Declaration:

```
uint32_t start_waveform (     const char* identifier,
                              uint16_t mode,
                              uint16_t dec_ratio,
                              uint32_t trig_mask,
                              uint32_t trig_level,
                              const double time,
                              uint16_t length )
```

Parameters:

`const char* identifier`

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

`const uint16_t mode`

The waveform acquisition mode (0 for analog mode, 1 for digital mode). Only analog mode is currently supported, so always set *mode* to 0.

`uint16_t dec_ratio`

An integer indicating the decimation of the acquired samples, from 1 to 32. If set to 1, a new sample is acquired every 16 ns; if set to N, one sample every (N·16 ns) is acquired.

`uint32_t trig_mask`

Specify the trigger mode of the waveform acquisition.

Each bit of the integer specifies whether the corresponding trigger mode is enabled (1) or disabled (0). Implemented trigger modes are:

- Bit 0: Enable instant trigger
- Bit 1: Enable rising slope crossing of trigger level
- Bit 2: Enable falling slope crossing of trigger level

If a triggering event does not occur until the time specified within the *time* variable is elapsed, the acquisition will be automatically triggered by an internal trigger.

`uint32_t trig_level`

Trigger level of the acquisition Expressed in ADC bin unit (range 0 to 2^16-1).

`const double time`

The amount of time the system waits for a triggering event, expressed in seconds.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

uint16_t length

Desired length of the waveform expressed in units of 16384 samples. For instance, if *length* is set to 1, 16384 samples are acquired; if set to 2, 32768 sample are acquired; and so on.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def  pystart_waveform (     string   (identifier),
                            uint16   (mode),
                            uint16_t (dec_ratio),
                            uint32_t (trig_mask),
                            uint32_t (trig_level),
                            double   (time),
                            uint16_t (length) ):


return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

uint16(mode)

The waveform acquisition mode (0 for analog mode, 1 for digital mode). Only analog mode is currently supported, so always set *mode* to 0.

uint16(dec_ratio)

An integer indicating the decimation of the acquired samples, from 1 to 32. If set to 1, a new sample is acquired every 16 ns; if set to N, one sample every (N·16 ns) is acquired.

uint32(trig_mask)

Specify the trigger mode of the waveform acquisition.

Each bit of the integer specifies whether the corresponding trigger mode is enabled (1) or disabled (0). Implemented trigger modes are:

- Bit 0: Enable instant trigger
- Bit 1: Enable rising slope crossing of trigger level
- Bit 2: Enable falling slope crossing of trigger level

If a triggering event does not occur until the time specified within the *time* variable is elapsed, the acquisition will be automatically triggered by an internal trigger.

uint32(trig_level)

Trigger level of the acquisition Expressed in ADC bin unit (range 0 to 2^16-1).

double(time)

The amount of time the system waits for a triggering event, expressed in seconds.

uint16(length)

Desired length of the waveform expressed in units of 16384 samples. For instance, if *length* is set to 1, 16384 samples are acquired; if set to 2, 32768 sample are acquired; and so on.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

start_list

Start a list mode acquisition for a specified amount of time or in free-running mode. The function stop must be called before another acquisition is started.

Asynchronous call.

C++ Declaration:

```
uint32_t start_list ( const char* identifier,
                      const double time )
```

Parameters:

const char* identifier

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

const double time

The amount of time the acquisition should run, expressed in seconds. The system precision is up to 0.001 secs (1 msec) for measurements from 1 ms to 48 hours. The system stops automatically after the specified time elapses, but the function stop must be called anyway before a new measure is started. Use 0 if a free-running acquisition is required.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def  pystart_list (   string(identifier),
                      double(time) ):

return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

double(time)

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

The amount of time the acquisition should run, expressed in seconds. The system precision is up to 0.001 secs (1 msec) for measurements from 1 ms to 48 hours. The system stops automatically after the specified time elapses, but the function stop must be called anyway before a new measure is started. Use 0 if a free-running acquisition is required.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

start_listwave

Start the acquisition in list-wave acquisition mode, for a specified amount of time and with additional acquisition settings (decimation ratio, length, offset). The function stop must be called before another acquisition is started.

Asynchronous call.

C++ Declaration:

```
uint32_t start_listwave (  const char* identifier,
                           const double time,
                           uint16_t dec_ratio,
                           uint16_t length,
                           uint16_t offset )
```

Parameters:

const char* identifier

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

const double time

The amount of time the acquisition should run, expressed in seconds. The system precision is up to 0.001 secs (1 msec) for measurements from 1 msec to 48 hours. The system stops automatically after the specified time elapses, but the function stop must be called anyway before a new acquisition is started.Use 0 to start a free-running acquisition.

uint16_t dec_ratio

An integer indicating the decimation of the acquired samples, from 1 to 32. If set to 1, a new sample is acquired every 16 ns; if set to N, a new sample is acquired every N·16 ns. Currently only a dec_ratio of 1 is supported.

uint16_t length

This parameter specifies the desired length of the waveform and is expressed in 16ns samples unit. Allowed range is from 8 to 800 samples.

uint16_t offset

This parameter adjusts the time offset of the acquisition. The digitalized waveform is continuously stored into a circular buffer. If offset is set to '0', the waveform acquisition will be triggered by the detection of the event and the downloaded waveform will not show samples immediately before the triggering event. By applying an offset, instead, it is possible to shift the waveform starting point towards earlier times, thus showing also samples before the triggering event and to reconstruct the full signal rising- or falling-edge.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
ACCREDIA

- confidential -

It is expressed in 16 ns·dec_ratio samples unit. Allowed range is between 0 and 300 samples.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def    pystart_listwave (  string(identifier),
                           double(time),
                           uint16(dec_ratio),
                           uint16(length),
                           uint16(offset) ):

return uint32(call_id)
```

Parameters:

string(identifier)

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

double(time)

The amount of time the acquisition should run, expressed in seconds. The system precision is up to 0.001 secs (1 msec) for measurements from 1 msec to 48 hours. The system stops automatically after the specified time elapses, but the function stop must be called anyway before a new acquisition is started.Use 0 to start a free-running acquisition.

uint16(dec_ratio)

An integer indicating the decimation of the acquired samples, from 1 to 32. If set to 1, a new sample is acquired every 16 ns; if set to N, a new sample is acquired every N·16 ns. Currently only a dec_ratio of 1 is supported.

uint16(length)

This parameter specifies the desired length of the waveform and is expressed in 16ns samples unit. Allowed range is from 8 to 800 samples.

uint16(offset)

This parameter adjusts the time offset of the acquisition. The digitalized waveform is continuously stored into a circular buffer. If offset is set to '0', the waveform acquisition will be triggered by the detection of the event and the downloaded waveform will not show samples immediately before the triggering event. By applying an offset, instead, it is possible

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

to shift the waveform starting point towards earlier times, thus showing also samples before the triggering event and to reconstruct the full signal rising- or falling-edge.

It is expressed in 16 ns·dec_ratio samples unit. Allowed range is between 0 and 300 samples.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

start_map

Start a map acquisition (multiple spectra) for a specified number of points. Behavior of the acquisition depends on the gating/triggering configuration.

Asynchronous call.

C++ Declaration:

```
uint32_t start_map (  const char* identifier,
                      const uint32_t sp_time,
                      const uint32_t points,
                      const uint16_t spect_depth )
```

Parameters:

`const char* identifier`

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

`const uint32_t sp_time`

Duration of each spectrum expressed in milliseconds.

- free-running mode configured: sp_time sets the spectrum time of each point.
- gating/trigger mode configured: sp_time only used internally to avoid spectrum saturation. Default value 100ms.

`const uint32_t points`

The number of spectra to acquire. If it is set to 0, the DPP continues to acquire indefinitely, until a stop is issued by the user (free-running map). There is no deadtime between one spectrum and the following, so the total acquisition time in ms will be sp_time·points (free-running mode).

`const uint16_t spect_depth`

The bin number for each spectrum. This value can be only 4096. No other values are allowed. This setting affects also the amount data returned by the getData function.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
Def pystart_map (  string(identifier),
                   uint32(sp_time),
                   uint32(points),
                   uint16(spect_depth)):
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

```
return uint32(call_id)
```

Parameters:

String(identifier)

A null-terminated string with the identifier(s) of the system to query. If more than one identifier is provided (e.g. because multiple USB or Ethernet connections are provided to a single chain to increase transfer throughput), identifiers must be separated by the special character "|" using the format: *ID1|ID2|…IDn\0*

uint32(sp_time)

Duration of each spectrum expressed in milliseconds.

- free-running mode configured: sp_time sets the spectrum time of each point.
- gating/trigger mode configured: sp_time only used internally to avoid spectrum saturation. Default value 100ms.

uint32(points)

The number of spectra to acquire. If it is set to 0, the DPP continues to acquire indefinitely, until a stop is issued by the user (free-running map). There is no deadtime between one spectrum and the following, so the total acquisition time in ms will be sp_time·points (free-running mode).

uint16(spect_depth)

The bin number for each spectrum. This value can be 1024, 2048 or 4096. No other values are allowed. This setting affects also the amount data returned by the getData function.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

disableBoard

Disable board of the chain which will not generate any payload during the acquisition. In case of multichannel systems, disabling an unused board, allows to maximize the frame-rate from the other boards. By default, all boards of the chain are active.

Asynchronous call.

C++ Declaration:

```
uint32_t disableBoard    ( char* identifier,
                           uint16_t Board,
                           bool disable)
```

Parameters:

char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

bool disable

True for disabling the board.

Return values:

The call ID if the parameters are filled with correct values, 0 otherwise.

Asynchronous answer data:

The function stores a 1 in the data array of the callback function (or in the answers queue) if the operation succeeded; otherwise, if the call was not successful, the type field will be 0.

Python Declaration:

```
def pydisableBoard   ( string(identifier),
                       uint16(Board),
                       bool(disable))
```

Parameters:

string(identifier)

A null-terminated string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

bool(disable)

True for disabling the board.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Retrieving acquired data

getAvailableData

Get the number of available spectra in map acquisition mode, or number of events in list mode. In the latter case, the function returns both the number of events and the waveforms. In any case, please note that this function does not retrieve any data; instead, it should be called before calling the appropriate readout functions in order to preallocate the correct amount of space.

C++ Declaration:

```
bool getAvailableData ( const char* identifier,
                        uint16_t Board
                        uint32_t& data_number)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint32_t& data_number

The number of available spectra in mapping mode or the number of events available for readout while in list mode. In case of list-wave acquisition, the data_number value is automatically increased by 1 every four waveform samples.

Return values:

A boolean true, if the operation succeeded, false otherwise.

Python Declaration:

```
def pygetAvailableData ( string(identifier),
                         uint16(Board):

return tuple(bool(ret_val), uint32(data_number))
```

Parameters:

string(identifier)

A string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint32(data_number)

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

The number of available spectra in mapping mode or the number of events available for readout while in list mode. In case of list-wave acquisition, the data_number value is automatically increased by 1 every four waveform samples and returned as an output.

Return values:

bool(ret_val)

A boolean true, if the operation succeeded, false otherwise.

uint32(data_number)

The number of available spectra in mapping mode or the number of events available for readout while in list mode. In case of list-wave acquisition, the data_number value is automatically increased by 1 every four waveform samples.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

isLastDataReceived

Check if the library has received the last data of the current acquisition from the board.

C++ Declaration:

```
bool isLastDataReceived(const char* identifier, uint16_t Board,
                                                    bool& LastDataReceived)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

bool& LastDataReceived

True if the library has received the last data of the current acquisition.

Return values:

A Boolean true if the returned value is correct, false if a problem occurs.

Python Declaration:

```
def pyisLastDataReceived(const char* identifier, uint16_t Board)
return tuple(bool(ret_val), bool(LastDataReceived))
```

Parameters:

string(identifier)

A string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Return values:

bool(ret_val)

Returns true if the returned value is correct, false if a problem occurs.

bool(LastDataReceived)

True if the library has received the last data of the current acquisition.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

getData

Get the acquired data. It should be used for:

- Single spectrum acquisition
- list mode acquisition.

It can be used also during a map acquisition, but it will return only the last complete spectrum acquired (use getAllData to retrieve all the spectra available). Other acquisition modes have specialized functions to retrieve data, described below.

C++ Declaration:

```
bool getData ( const char* identifier,
               uint16_t Board,
               uint64_t* values,
               uint32_t& id,
               statistics& stats,
               uint32_t& spectra_size)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

The meaning of the remaining parameters depends on the acquisition mode selected:

uint64_t* values

Single spectrum: array of uint64_t (64 bits wide) containing the values of each bin of the spectrum (the spectrum length is programmable and can be 1024, 2048 or 4096).

List mode: array of uint64_t (64 bits wide) containing information about each single event recorded by the DPP. Each 64 bit value corresponds to an event and is organized in this way:

Bits 63 to 62	bits 61 to 18	bits 17 to 16	bits 15 to 0
Reserved, for internal use	Timestamp of the X-Ray arrival time (8 ns samples)	Reserved, for internal use	Energy of the acquired X-Ray.

uint32_t& id

Unique progressive identifier associated to each spectrum acquired. The id is reset only after a power cycle.

statistics& stats

A structure containing the statistics associated to the acquisition. Memory for this structure must be pre-allocated and maintained by the caller.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

The structure is defined with these fields:

```
struct statistics {
        // Basic statistics.
        uint64_t real_time;
        uint64_t live_time;
        double   ICR;
        double   OCR;
        // Advanced statistics.
        uint64_t last_timestamp;      // last timestamp
        uint32_t detected;            // detected events
        uint32_t measured;            // measured and processed events
        uint32_t edge_dt;             // internal use
        uint32_t filt1_dt;            // dead-time of energy filter
        uint32_t zerocounts;          // number of zero counts (artifact)
        uint32_t baselines_value;     // number of calculated baselines
        uint32_t pup_value;           // pile-up detected by fast filter
        uint32_t pup_f1_value;        // pile-up detected by energy filter
        uint32_t pup_notf1_value;     // internal use
        uint32_t reset_counter_value; // number of detected resets
};
```

In this structure both some basic and advanced statistics are reported, the former including real time (expressed in microseconds), live time (expressed in microseconds), input and output count rates (expressed in kcps) of the acquisition. The advanced statistics, instead, include information related to pileup rejection and various other aspects. For more information about them, please contact the vendor.

uint32_t& spectra_size

Single spectrum: spectrum size. It must be 1024, 2048 or 4096.

List mode: number of events the caller wants to read. They must be equal to or less than the available stored events. Use the function **getAvailableData** in order to know the number of counts available.

Return values:

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

Remarks:

In normal acquisition mode (single spectrum), the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

Python Declaration:

```
def  pygetData ( string(identifier),
                 uint16(Board),
                 uint32(spectra_size)):

return tuple(bool(ret_val), uint64(values), uint32(id), statistics(stats))
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

Parameters:

String(identifier)

A null-terminated string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint32_t(spectra_size)

Single spectrum: spectrum size. It must be 1024, 2048 or 4096.

List mode: number of events the caller wants to read. They must be equal to or less than the available stored events. Use the function **getAvailableData** in order to know the number of counts available.

Return values:

bool(ret_val)

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

uint64_t* values

Single spectrum: array of uint64_t (64 bits wide) containing the values of each bin of the spectrum (the spectrum length is programmable and can be 1024, 2048 or 4096).

List mode: array of uint64_t (64 bits wide) containing information about each single event recorded by the DPP. Each 64 bit value corresponds to an event and is organized in this way:

Bits 63 to 62	bits 61 to 18	bits 17 to 16	bits 15 to 0
Reserved, for internal use	Timestamp of the X-Ray arrival time (8 ns samples)	Reserved, for internal use	Energy of the acquired X-Ray.

uint32_t& id

Unique progressive identifier associated to each spectrum acquired. The id is reset only after a power cycle.

statistics& stats

A structure containing the statistics associated to the acquisition. Memory for this structure must be pre-allocated and maintained by the caller.

The structure is defined with these fields:

```
class statistics:
        // Basic statistics.
        uint64(real_time
        uint64(live_time
        double(ICR
        double(OCR
        // Advanced statistics.
        uint64(last_timestam)          // last timestamp
        uint32(detected)               // detected events
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
ACCREDIA

- confidential -

```
uint32(measured)              // measured and processed events
uint32(edge_dt)               // internal use
uint32(filt1_dt)              // dead-time of energy filter
uint32(zerocounts)            // number of zero counts (artifact)
uint32(baselines_value)       // number of calculated baselines
uint32(pup_value)             // pile-up detected by fast filter
uint32(pup_f1_value)          // pile-up detected by energy filter
uint32(pup_notf1_value)       // internal use
uint32(reset_counter_value)   // number of detected resets
```

In this structure both some basic and advanced statistics are reported, the former including real time (expressed in microseconds), live time (expressed in microseconds), input and output count rates (expressed in kcps) of the acquisition. The advanced statistics, instead, include information related to pileup rejection and various other aspects. For more information about them, please contact the vendor.

Remarks:

In normal acquisition mode (single spectrum), the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
ACCREDIA

- confidential -

getAllData

Get a user-defined number of spectra during a mapping mode acquisition. This function must not be used while in other acquisition modes: use **getData** instead.

C++ Declaration:

```
bool getAllData ( const char* identifier,
                  uint16_t Board,
                  uint16_t* values,
                  uint32_t* id,
                  double* stats,
                  uint64_t* advstats,
                  uint32_t& spectra_size,
                  uint32_t& data_number)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint16_t* values

Array of uint16_t (16 bits wide) containing the counts of each bin (the spectrum length is programmable and can be 1024, 2048 or 4096) of each spectrum. The spectra are one next to each other, starting from the first acquired. Therefore, for instance, if 10 spectra are committed, the resulting values length will be 10·spectrum_length.

uint32_t* id

Array of unique progressive identifiers of each spectrum. The first id of the array corresponds to the first spectrum acquired (as for the parameter values).

double* stats

Array of basic statistics for each spectrum. Size of the vector will be 4·data_number.

uint64_t* advstats

Array of advanced statistics of each spectrum. Size of the vector will be 22·data_number

uint32_t& spectra_size

The size of the spectrum. It must be 1024, 2048 or 4096.

uint32_t& data_number

The number of spectra requested. It must be equal to or less than the number of available spectra. The user should call the **getAvailableData** function first in order to query the number of available spectra and consequently preallocate enough space for spectra and statistics.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Return values:

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

Remarks:

In mapping acquisition mode, the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

Python Declaration:

```
def  pygetAllData ( string(identifier),
                    uint16(Board),
                    uint32(spectra_size),
                    uint32(data_number)):

return

tuple(bool(ret_val),uint16(values),uint32(id),double(stats),uint64(adv_stats))
```

Parameters:

`string(identifier)`

A null-terminated string with the identifier of the system to query.

`uint16(Board)`

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

`uint32(spectra_size)`

The size of the spectrum. It must be 1024, 2048 or 4096.

`uint32(data_number)`

The number of spectra requested. It must be equal to or less than the number of available spectra. The user should call the **`getAvailableData`** function first in order to query the number of available spectra and consequently preallocate enough space for spectra and statistics.

Return values:

`bool(ret_val)`

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

`uint16(values)`

Array of uint16_t (16 bits wide) containing the counts of each bin (the spectrum length is programmable and can be 1024, 2048 or 4096) of each spectrum. The spectra are one next to

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

each other, starting from the first acquired. Therefore, for instance, if 10 spectra are committed, the resulting values length will be 10·spectrum_length.

uint32(id)

Array of unique progressive identifiers of each spectrum. The first id of the array corresponds to the first spectrum acquired (as for the parameter values).

Double(stats)

Array of basic statistics for each spectrum. Size of the vector will be 4·data_number.

uint64(advstats)

Array of advanced statistics of each spectrum. Size of the vector will be 22·data_number

Remarks:

In mapping acquisition mode, the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

Return Values Description

stats

Array of basic statistics for each spectrum. Size of the vector is 4·data_number. The statistics are described in the following table, where *i* is the collected spectrum index:

Field	Description
stats[i * 4]	Real time (us)
stats[i * 4 +1]	Live time (us)
stats[i * 4 + 2]	ICR (Kcps)
stats[i * 4 + 5]	OCR (kcps)

advstats

Array of advanced statistics for each spectrum. Size of the vector is 22·data_number. The statistics are described in the following table, where *i* is the collected spectrum index:

Field	Description
advstats[i * 22]:	Last_timestamp
advstats[i * 22 + 1]:	detected
advstats[i * 22 + 2]:	measured
advstats[i * 22 + 5]:	zerocounts
advstats[i * 22 + 14]	Gate/trigg rising
advstats[i * 22 + 15]	Gate/trigg falling
advstats[i * 22 + 16]	Gate high
advstats[i * 22 + 17]	Gate low

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

getLiveDataMap

Gets the acquired data on live mapping acquisition.

C++ Declaration:

```
bool getLiveDataMap ( const char* identifier,
                      uint16_t Board,
                      uint16_t* values,
                      uint32_t& id,
                      statistics& stats,
                      uint32_t& spectra_size)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

Uint16_t* values

Array of uint64_t (64 bits wide) containing the values of each bin of the spectrum (the spectrum length is programmable and can be 1024, 2048 or 4096).

uint32_t& id

Each spectrum acquired has a unique progressive identifier returned with this parameter.

statistics& stats

A structure containing the statistics associated to the acquisition. Memory for this structure must be allocated and maintained by the caller.

The structure is defined with these fields:

```
struct statistics {
        // Basic statistics.
        uint64_t real_time;
        uint64_t live_time;
        double ICR;
        double OCR;
        // Advanced statistics.
        uint64_t last_timestamp;      // last timestamp
        uint32_t detected;            // detected events
        uint32_t measured;            // measured and processed events
        uint32_t edge_dt;             // internal use
        uint32_t filt1_dt;            // dead-time of energy filter
        uint32_t zerocounts;          // number of zero counts (artifact)
        uint32_t baselines_value;     // number of calculated baselines
        uint32_t pup_value;           // pile-up detected by fast filter
        uint32_t pup_f1_value;        // pile-up detected by energy filter
        uint32_t pup_notf1_value;     // internal use
        uint32_t reset_counter_value; // number of detected resets
};
```

In this structure both some basic and advanced statistics are reported, the former including real time (expressed in microseconds), live time (expressed in microseconds), input and output

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

count rates (expressed in kcps) of the acquisition. The advanced statistics, instead, include information related to pileup rejection and various other aspects. For more information about them, please contact the vendor.

uint32_t& spectra_size

The size of the spectrum. It must be 1024, 2048 or 4096.

Return values:

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

Remarks:

In mapping acquisition mode, the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

Python Declaration:

```
bool pygetLiveDataMap ( const char* identifier,
                        uint16_t Board,
                        uint32_t& spectra_size)

return

tuple(bool(ret_val),uint16(values),uint32(id),double(stats))
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint32_t& spectra_size

The size of the spectrum. It must be 1024, 2048 or 4096.

Return values:

bool(ret_val)

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

Uint16_t values

Array of uint64_t (64 bits wide) containing the values of each bin of the spectrum (the spectrum length is programmable and can be 1024, 2048 or 4096).

uint32_t id

Each spectrum acquired has a unique progressive identifier returned with this parameter.

statistics& stats

A structure containing the statistics associated to the acquisition. Memory for this structure must be allocated and maintained by the caller.

The structure is defined with these fields:

```
struct statistics {
        // Basic statistics.
        uint64_t real_time;
        uint64_t live_time;
        double ICR;
        double OCR;
        // Advanced statistics.
        uint64_t last_timestamp;      // last timestamp
        uint32_t detected;            // detected events
        uint32_t measured;            // measured and processed events
        uint32_t edge_dt;             // internal use
        uint32_t filtl_dt;            // dead-time of energy filter
        uint32_t zerocounts;          // number of zero counts (artifact)
        uint32_t baselines_value;     // number of calculated baselines
        uint32_t pup_value;           // pile-up detected by fast filter
        uint32_t pup_f1_value;        // pile-up detected by energy filter
        uint32_t pup_notf1_value;     // internal use
        uint32_t reset_counter_value; // number of detected resets
};
```

In this structure both some basic and advanced statistics are reported, the former including real time (expressed in microseconds), live time (expressed in microseconds), input and output count rates (expressed in kcps) of the acquisition. The advanced statistics, instead, include information related to pileup rejection and various other aspects. For more information about them, please contact the vendor.

Remarks:

In mapping acquisition mode, the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

getListWaveData

Gets the acquired data. It should be used for list-wave DPP acquisitions.

C++ Declaration:

```
bool getListWaveData ( const char* identifier,
                       uint16_t Board,
                       uint64_t* values,
                       uint64_t* wave_values,
                       uint32_t& id,
                       statistics& stats,
                       uint32_t& spectra_size)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint64_t* values

Array of uint64_t (64 bits wide) containing information about each single event recorded and the acquired waveform. More information is provided below.

uint64_t* wave_values

Currently not used.

uint32_t& id

Each spectrum acquired has a unique progressive identifier, returned with this parameter.

statistics& stats

A structure containing the statistics associated to the acquisition. Memory for this structure must be allocated and maintained by the caller.

The structure is defined with these fields:

```
struct statistics {
        // Basic statistics.
        uint64_t real_time;
        uint64_t live_time;
        double ICR;
        double OCR;
        // Advanced statistics.
        uint64_t last_timestamp;      // last timestamp
        uint32_t detected;            // detected events
        uint32_t measured;            // measured and processed events
        uint32_t edge_dt;             // internal use
        uint32_t filtl_dt;            // dead-time of energy filter
        uint32_t zerocounts;  // number of zero counts (artifact)
        uint32_t baselines_value;     // number of calculated baselines
        uint32_t pup_value;   // pile-up detected by fast filter
        uint32_t pup_f1_value;        // pile-up detected by energy filter
        uint32_t pup_notf1_value; // internal use
        uint32_t reset_counter_value; // number of detected resets
```


XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification
ACCREDIA

- confidential -

```
};
```

In this structure both some basic and advanced statistics are reported, the former including real time (expressed in microseconds), live time (expressed in microseconds), input and output count rates (expressed in kcps) of the acquisition. The advanced statistics, instead, include information related to pileup rejection and various other aspects. For more information about them, please contact the vendor.

uint32_t& spectra_size

Number of elements to be retrieved from the queue of energy – timestamp – waveform. See "parameter description" section to check how data is organized into the "values" vector.

Return values:

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

Remarks:

In list wave acquisition mode, the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

Python Declaration:

```
def pygetListWaveData ( string(identifier),
                        uint16(Board),
                        uint32(spectra_size)):
return

tuple(bool(ret_val), uint16(values), uint32(id), statistics(stats))
```

Parameters:

string(identifier)

A null-terminated string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint32(spectra_size)

Number of elements to be retrieved from the queue of energy – timestamp -waveform. See "parameter description" section to check how data is organized into the "values" vector.

Return values:

bool(ret_val)

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

uint64(values)

Array of uint64_t (64 bits wide) containing information about each single event recorded and the acquired waveform. More information is provided below.

uint32_t(id)

Each spectrum acquired has a unique progressive identifier returned with this parameter.

Statistics(stats)

A structure containing the statistics associated to the acquisition. Memory for this structure must be allocated and maintained by the caller.

The structure is defined with these fields:

```
class statistics:
        // Basic statistics.
        uint64(real_time
        uint64(live_time
        double(ICR)
        double(OCR)
        // Advanced statistics.
        uint64(last_timestam)          // last timestamp
        uint32(detected)               // detected events
        uint32(measured)               // measured and processed events
        uint32(edge_dt)                // internal use
        uint32(filt1_dt)               // dead-time of energy filter
        uint32(zerocounts)             // number of zero counts (artifact)
        uint32(baselines_value)        // number of calculated baselines
        uint32(pup_value)              // pile-up detected by fast filter
        uint32(pup_f1_value)           // pile-up detected by energy filter
        uint32(pup_notf1_value)        // internal use
        uint32(reset_counter_value)    // number of detected resets
```

In this structure both some basic and advanced statistics are reported, the former including real time (expressed in microseconds), live time (expressed in microseconds), input and output count rates (expressed in kcps) of the acquisition. The advanced statistics, instead, include information related to pileup rejection and various other aspects. For more information about them, please contact the vendor.

Remarks:

In list wave acquisition mode, the first bin of the spectrum also includes events with negative measured energy (caused for instance by noise); in the same way, events with measured energy exceeding the spectrum range (i.e. larger than the last bin) are included in the last bin of the spectrum.

Parameter Description

uint64_t* values

Array of uint64_t (64 bits wide) containing information about each single event recorded by the DPP. Each event word is followed by its related waveform using the following structure:

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

Energy1 & Timestamp1
Sample1&Sample2&Sample3&Sample4
Sample5&Sample6&Sample7&Sample8
…
Energy2 & Timestamp2

Each 64-bit Energy&Timestamp value corresponds to an event and is organized in this way:

Bits 63 to 62	bits 61 to 18	bits 17 to 16	bits 15 to 0
Internal use	Timestamp of the X-Ray arrival time (8 ns samples)	Internal use	Energy of the acquired X-Ray.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

getWaveData

Gets the waveform data. It must be used in waveform mode only.

C++ Declaration:

```
bool getWaveData ( const char* identifier,
                   uint16_t Board,
                   uint16_t* values,
                   uint32_t& data_size)
```

Parameters:

const char* identifier

A null-terminated string with the identifier of the system to query.

uint16_t Board

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint16_t* values

Array of ADC samples (ranging between 0 and 65535), starting from the first acquired sample up to spectra_size acquired samples. The sample acquisition frequency is 62.5 MHz / decimation ratio.

uint32_t& data_size

The number of samples the caller wants to read. They must be equal or less than the actual waveform length.

Return values:

Returns true if the values array is filled with correct values, false otherwise.

Python Declaration:

```
def pygetWaveData (  string(identifier),
                   uint16(Board),
                   uint32(data_size)):
return

tuple(bool(ret_val),uint16(values),uint32(id))
```

Parameters:

String(identifier)

A null-terminated string with the identifier of the system to query.

uint16(Board)

It contains the progressive number of the board to query, starting from '0' for the MASTER of the chain.

uint32_(data_size)

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

The number of samples the caller wants to read. They must be equal or less than the actual waveform length.

Return values:

bool(ret_val)

Returns true if the values array is filled with correct values, false otherwise. The function returns false also if there is no data available to be read.

uint16(values)

Array of ADC samples (ranging between 0 and 65535), starting from the first acquired sample up to spectra_size acquired samples. The sample acquisition frequency is 62.5 MHz / decimation ratio.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Getting Asynchronous Calls' Answers

register_callback

Register the callback function pointer to get the answers of asynchronous calls. Available only in the Callback release.

C++ Declaration:

```
bool register_callback ( void (*callback)( uint16_t type,
                                           uint32_t call_id,
                                           uint32_t length,
                                           uint32_t* data))
```

Parameters:

void (*callback)(uint16_t type, uint32_t call_id, uint32_t length, uint32_t* data)

A pointer to a function with this signature:

uint16_t type

The type of operation to which the answer refers to. It can have three values: 1 if it refers to a read operation, 2 if it refers to a write operation, 0 if an error occurred.

uint32_t call_id

It holds the call ID of the corresponding request, so that the user can combine answers with requests.

uint32_t length

The number of elements contained in the *data* array.

uint32_t* data

Array that contains the read data in case of a read command, or a 1 in case of a successful write command.

Return values:

A boolean true, if the registering succeeded, false otherwise.

Python Declaration:

```
bool pyregister_callback ( void (*callback)) :
return

bool(ret_val)
```

Parameters:

void (*callback)(uint16_t type, uint32_t call_id, uint32_t length, uint32_t* data)

A pointer to a function with this signature:

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -

`uint16_t type`

The type of operation to which the answer refers to. It can have three values: 1 if it refers to a read operation, 2 if it refers to a write operation, 0 if an error occurred.

`uint32_t call_id`

It holds the call ID of the corresponding request, so that the user can combine answers with requests.

`uint32_t length`

The number of elements contained in the *data* array.

`uint32_t* data`

Array that contains the read data in case of a read command, or a 1 in case of a successful write command.

Return values:

A boolean true, if the registering succeeded, false otherwise.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

GetAnswersDataLength

Get how many elements there are in the answers queue. Available only in the Polling release.

C++ Declaration:

```
bool getAnswersDataLength ( uint32_t& length )
```

Parameters:

uint32_t& length

The number of elements in the answers queue.

Return values:

A boolean true, if the returned number is correct, false otherwise.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

GetAnswersData

Get a variable number of elements from the answers queue. Available only in the Polling release.

C++ Declaration:

```
bool GetAnswerData ( uint32_t length,
                     uint32_t* data )
```

Parameters:

uint32_t length

The number to be extracted from the answers queue.

uint32_t* data

An array containing the element retrieved from the answers queue.

Return values:

A boolean true, if the returned data is valid, false otherwise.

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

ISO 9001 BUREAU VERITAS Certification

- confidential -

Manual revisions

- **2.3**: first manual release for DPP-4553.
- **3.0**: manually partially rewritten.
- **3.1**: aligned with library version 3.4; added methods *load_new_firmware*, *configure_timestamp_delay*; added Python support to missing methods.
- **3.2:** added specifications of single-spectrum, mapping, list-mode.
- **3.3**: fix load_new_firmware method.
- **3.4**: small fixes
- **3.5**: added disableBoard()
- **3.6**: fix limits for gain of dpp
- **3.7**: added isLastDataReceived method.
- **3**.8: added autoScanSlaves recommandation.
- **4**.0: flush_local_eth_conn()
- **4**.1: gain limit in table parameter description

XGLab S.R.L.
Via Conte Rosso 23, I-20134 Milano (Italy)

- confidential -